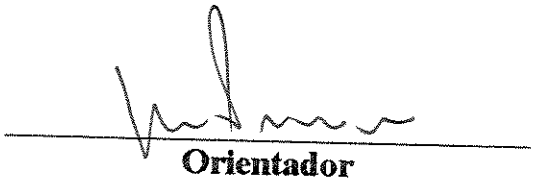
ANÁLISE DE TEMPOS DE ESTABELECIMENTO E ORDEM DE MANUFATURA NO SEQÜENCIAMENTO DE TAREFAS EM PROCESSOS BATELADA

Esta versão corresponde à redação final da Tese de Mestrado defendida pelo Engenheiro Edilson de Jesus Santos, e aprovada pela Comissão julgadora em 24/03/94.



Orientador

UNIVERSIDADE ESTADUAL DE CAMPINAS

FACULDADE DE ENGENHARIA QUÍMICA

DEPARTAMENTO DE ENGENHARIA DE SISTEMAS QUÍMICOS

ANÁLISE DE TEMPOS DE ESTABELECIMENTO E ORDEM DE MANUFATURA NO SEQÜENCIAMENTO DE TAREFAS EM PROCESSOS BATELADA

Por Edilson de Jesus Santos
Orientador: Dr. João Alexandre Ferreira Rocha Pereira

Tese apresentada à Faculdade de Engenharia Química FEQ - UNICAMP como um dos requisitos exigidos para obtenção de título MESTRE EM ENGENHARIA

(Março / 94)
Campinas - SP
Brasil

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA CENTRAL -UNICAMP

Santos, Edilson de Jesus

Sa59a

Análise de tempos de estabelecimento e ordem de manufatura no seqüenciamento de tarefas em processos batelada / Edilson de Jesus Santos, -- Campinas, SP : [s.n.], 1994.

Orientador: João Alexandre F. Rocha Pereira.
Dissertação(mestrado) - Universidade Estadual de Campinas
Faculdade de Engenharia Química.

1. Administração da Produção. 2. Produção-Seqüenciamento. 3. Processos de fabricação. I. Pereira, João Alexandre F. Rocha. II. Universidade Estadual de Campinas, Faculdade de Engenharia Química. III. Título.

20. CDD - 658.5 -658.53
-670.42

Índices para catálogo sistemático

1. Administração da produção	658.5
2. Produção- Seqüenciamento	658.53
3. Processos de fabricação	670.42

Tese defendida e aprovada em 24 de Março de 1994, pela banca examinadora constituída pelos professores:

Prof. Dr. Rafael Santos Mendes

Prof ª. Dr ª. Maria Tereza M. Rodrigues

Prof. Dr. João Alexandre F. R. Pereira
(Orientador)

Dedico este trabalho a minha mãe Maria Anatilde de Jesus Santos e a todos os meus irmãos.

Agradeço

a Deus pela minha objetividade

a minha mãe, Maria Anatilde de Jesus Santos, pela coragem e segurança transmitidas durante todos estes anos.

a toda minha Família pelas oportunidades oferecidas , as quais contribuíram indiscutivelmente para que eu chegasse neste estágio

ao Prof. Dr. João Alexandre R. F. Pereira pela orientação oferecida durante o desenvolvimento deste trabalho

a Profa. Dr. Sandra Cruz e ao Prof. Sérgio Persio Ravagnani pela atenção e apoio oferecidos ao longo desta formação.

a todos os professores do Departamento de Engenharia de Sistemas Químicos da UNICAMP pela atenção dada

aos Professores do Departamento de Engenharia Química da Universidade Federal de Sergipe(UFS) pelo incentivo dado.

a Sra. Rosa Maria Moretti pela atenção e compreensão dadas ao longo deste tempo.

a todos que contribuíram de uma forma ou de outra para o término desta Tese.

RESUMO

A programação de produção deve ser vista como um conjunto de funções para a qual convergem informações que serão transformadas convenientemente em instruções para os diversos departamentos de uma indústria. A interação entre o setor de programação e os setores produtivos deve ser bastante dinâmica, intensificando o fluxo de informações dentro da organização e fazendo com que os setores de produção desenvolvam com mais eficiência os objetivos pré-estabelecidos.

O objetivo de qualquer empresa é fornecer aos seus clientes o melhor produto possível, e com menor custo. Para tanto, faz-se necessário adotar uma política de conduta organizacional para obtenção das metas estabelecidas. Neste ponto, o Planejamento e Programação de Produção deve determinar "o que fazer", "onde fazer, "como fazer" e "quando fazer", assim que os recursos necessários e os projetos dos produtos a serem manufaturados estejam definidos.

A programação de produção é o instrumento que possibilita a conduta racional, e não aleatória, dos procedimentos ou operações que vão conduzir a matéria-prima ao produto final. É inegável que sem a coordenação das tarefas e sem a busca de um roteiro de fabricação ótimo, que venha minimizar custos e aumento da produtividade (minimização de tempo de execução do plano de produção) é praticamente impossível alcançar as metas estabelecidas.

A proposta da presente tese enfoca o escalonamento de produção, mais precisamente o seqüenciamento de tarefas, dentro dos setores produtivos em plantas batelada multiproduto.

Dado um conjunto de N tarefas e os diversos fatores de processamento que influenciam na execução das operações, o setor responsável pela programação de produção deve fornecer aos setores produtivos qual o melhor roteiro de manufatura dos produtos que venha a minimizar uma determinada função objetivo conveniente com as metas determinadas pela organização.

Em nossa abordagem para a solução dos problemas envolvendo o seqüenciamento de tarefas utilizaremos a metodologia de busca controlada em árvore , mais conhecida como metodologia "Branch and Bound"("BAB"), tendo em vista a sua eficiência na busca de um roteiro ótimo, principalmente quando o valor do menor limitante da função objetivo tomada como critério para o seqüenciamento das tarefas é calculado com garantia.

ÍNDICE

CAPÍTULO 1 - INTRODUÇÃO

O projeto e a operação de processos contínuos vêm se tornando objetivos constantes a serem atingidos no campo da engenharia química. No entanto, processos que se realizam em pequena escala (pequeno volume de produção), ou que exigem elevados tempos de residência, ou procedimentos complexos para a síntese ou separação de produtos, com rígidos padrões de controle de qualidade, são realizáveis com melhor desempenho e eficácia em processos batelada.

Em geral, numa planta batelada são produzidos diversos produtos utilizando-se diversas unidades de processamento, podendo um mesmo equipamento ser utilizado para o processamento de diversos produtos, cada um podendo ter tempos diferentes de manufatura. É neste fato que reside a principal característica deste tipo de estrutura, que é a flexibilidade de produção por meio do compartilhamento dos processadores e coordenação de trabalhos.

Para produzir com eficiência, não é mais possível simplesmente comunicar aos setores da indústria, mais precisamente aos setores de produção , a necessidade de um determinado produto. Torna-se imperativo que estes setores tenham um plano de trabalho otimizado, fruto da análise dos diversos fatores que influenciam no desenvolvimento das tarefas, assim como: disponibilidade de equipamento, armazenagem intermediária, estocagem de produtos, mão-de-obra, utilidades ,etc.

A necessidade do planejamento de produção é vital em qualquer indústria, pois geralmente as quantidades necessárias e frequências de pedidos dos produtos, situação bem característica nas indústrias que fabricam por encomenda, não coincidem com os valores ótimos de operação da planta e nem com a disponibilidade de recursos existentes.

Na literatura consultada, nota-se que para diminuir o grau de complexidade no tratamento da função planejamento, algumas aproximacões são feitas. Reklaitis (1982) refere-se ao tratamento do planejamento global através de aproximações. Uma destas aproximações decompõe o planejamento geral em dois subproblemas (dois níveis) interligados: um subproblema de Planejamento a longo e médio prazo e outro subproblema de planejamento a curto prazo (escalonamento).

Tipicamente, o problema de escalonamento de produção envolve um conjunto de tarefas a serem programadas e um critério de desempenho que deve ser otimizado. O caso mais simples de escalonamento é o seqüenciamento de tarefas, onde a ordem de fabricação já determina o programa de produção num horizonte especificado.

A proposta da presente tese objetiva determinar um roteiro de fabricação ótimo, sendo dado um conjunto de N tarefas e M processadores aos quais serão associados, numa etapa posterior, tempos de estabelecimento.e ordem de manufatura.

Neste presente trabalho, procura-se resolver os seguinte problemas numa planta multiproduto:

- O seqüenciamento de tarefas com tempos de estabelecimento(setup times) e de transferência negligenciáveis(transfer times), tendo o tempo de execução total das tarefas ("makespan") como função de desempenho.
- O seqüenciamento de tarefas para a minimização do tempo total de execução das tarefas com tempos de estabelecimento não negligenciáveis.
- O seqüenciamento de tarefas com ordem prioritária de manufatura de produtos.

CAPÍTULO 2 - PLANTAS BATELADA: CARACTERIZAÇÃO E ESCALONAMENTO DE PRODUÇÃO

2.1 - Introdução

Durante muitos anos, o uso de processamento contínuo tem se tornado o modo de operação bastante almejado por diversos setores industriais. Parakrama (1985) mostra que somente 6% dos processos batelada foram substituídos por processos contínuos, concluindo-se que os primeiros ainda têm sua importância nos processos de produção. O modo de operação a ser praticado é inerente às características e ao número de produtos que se pretende fabricar. A tabela 2.1 mostra as principais características que distinguem estes dois tipos de processamento: contínuo e não-contínuo.

Tabela 2.1 - Diferenciação dos tipos de Processamento

Tipo contínuo	Tipo não-contínuo (batelada e semi-contínuo)
Alto volume de produção e baixa flexibilidade de fabricação	Pequeno volume e produção e alta flexibilidade de fabricação
Pouca variedade de produtos	Grande variedade de produtos
Baixa armazenagem intermediária	Alta armazenagem irtermediária
Baixo tempo de residência	Alto tempo de residência
Produto com baixo valor agregado	Produto com alto valor agregado

A utilização de operações batelada são economicamente desejáveis, principalmente, quando o valor agregado aos produtos é alto, mas em contrapartida o número de atividades para desenvolvimento das tarefas é bastante volumoso. Além disso, existem operações que por características próprias encontram-se impossibilitadas de serem desenvolvidas continuamente, como por exemplo: um processo contínuo é impraticável no setor siderúrgico, onde algumas operações exigem altas temperaturas.

2.2 - Estrutura de Produção

2.2.1 - Introdução

Plantas batelada são classificadas em duas categorias: multiproduto e multipropósito. A coordenação dos trabalhos nas plantas batelada é de fundamental importância para que se encontre um plano de produção ótimo. Ku et al. (1987) relatam que a produtividade total e eficácia econômica de uma planta dependem de maneira

crítica do roteiro de produção. Desta forma, é necessário que os recursos disponíveis sejam utilizados racionalmente durante a manufatura das diversas tarefas.

2.2.2 - Planta Multiproduto

Em plantas multiproduto, as tarefas são manufaturadas sucessivamente em uma seqüência de campanhas. Para cada tarefa somente uma rota é permitida. Uma alternativa de produção é projetar uma estrutura com duas ou mais plantas multiproduto operando em paralelo. Esta estratégia tem a vantagem de aumentar a oferta de produtos, assim que o mercado exigir, utilizando-se da manufatura de produtos com tempos similares.

A principal característica das plantas multiproduto é a sua relação de precedência linear entre as unidades, como mostra a figura 2.1. Uma idealização feita para a solução de problemas relativos às plantas com esta configuração é considerar que somente tanques batelada constituem a linha de processamento, mas como se sabe em uma situação real equipamentos semi-contínuos são muitas vezes intercalados entre os equipamentos batelada. Quando estes equipamentos não são negligenciados em problemas "flowshops" (problemas relativos às plantas batelada multiproduto) algumas aproximações são feitas para a solução do problema. Uma destas aproximaxões é considerar que todos os equipamentos pertencentes à série de unidades semi-contínuas ("semicontinuous subtrain") operam no mesmo período de tempo e têm a mesma velocidade de processamento. A figura 2.2 esquematiza uma planta multiproduto com vários conjuntos de equipamento semi-contínuo presentes entre os equipamentos batelada.

Quando a planta batelada multiproduto é caracterizada pela ausência de semi-contínuos a planta recebe o nome de "Linha pura de processamento" ou "flowshop puro", onde somente tanques batelada são considerados.

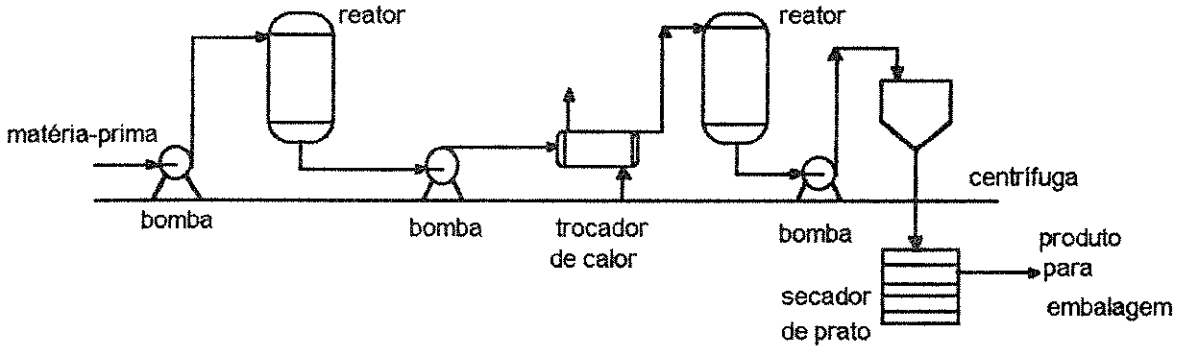


figura 2.1 - Planta batelada multiproduto

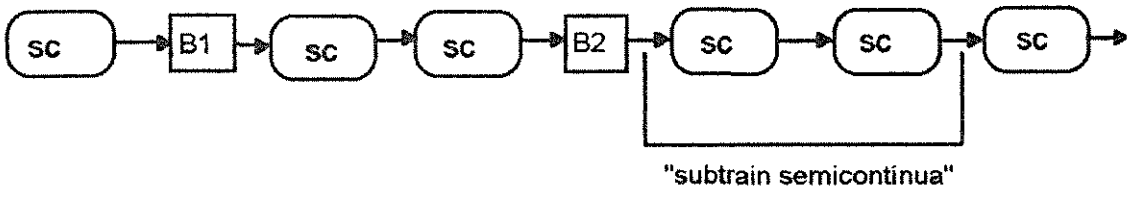


figura 2.2 - Modos de operação dos processadores

A estrutura de uma linha pura de processamento esquematizada na figura 2.3 contem M processadores onde N tarefas são desenvolvidas. Cada tarefa, em cada processador, exige uma única operação . O fluxo de atividades é unidirecional seguindo a ordem natural das unidades. Os processadores são numerados de j = 1 a M; e as operações da tarefa i são correspondentemente enumeradas a cada equipamento, sendo cada tarefa, portanto, constituída por M operações.

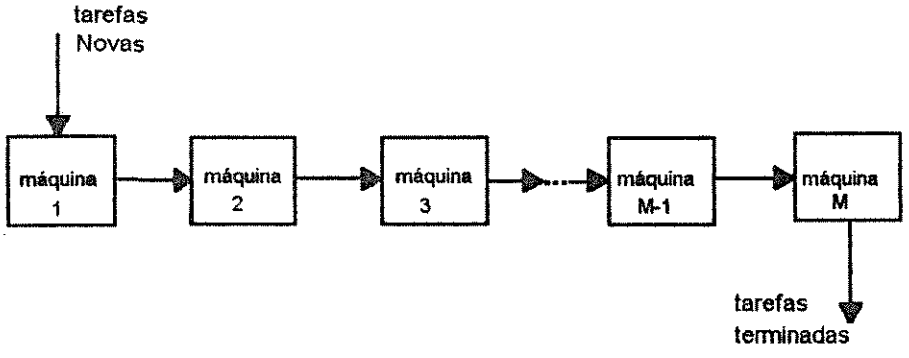


figura 2.3 - Linha "Pura"de Processamento

2.2.3 - Planta Multipropósito

Em plantas multipropósito um conjunto de diferentes produtos pode estar presente na planta no mesmo tempo e o mesmo produto pode ser processado por rotas diferentes em tempos dissimilares. Estas alternativas podem ou não ser pré-definidas e a distribuição de tempo de produção para diferentes produtos é realizada por campanhas multiproduto ou pelo seqüenciamento de bateladas individuais.

Em plantas multiproduto, determinar um programa de produção detalhado envolve principalmente uma distribuição de tempo, desde que não exista liberdade na escolha de equipamentos. Em plantas multipropósito esta situação não ocorre, pois uma determinada tarefa pode ser executada em diferentes unidades, consequentemente a

escolha da tarefa e da unidade a ser utilizada deve ser feita previamente. Egli e Rippin (1986) desenvolveram um programa "SRSBP" ("Short-Range Scheduling of Batch Plants) para escalonamento a curto-prazo em plantas multiproduto/multipropósito. O código determina a seqüência e o programa detalhado de um conjunto de bateladas, encontrando uma demanda específica que minimiza custos de estocagem, custos de mudanças de batelada de tarefas e custos de utilidades, tendo como entrada básica: o número de equipamentos disponíveis, distribuição de datas de produção, equipamentos necessários para cada tarefa. As figuras 2.4 e 2.5 mostram uma planta batelada multipropósito típica e o desenvolvimento de tarefas neste tipo de planta.

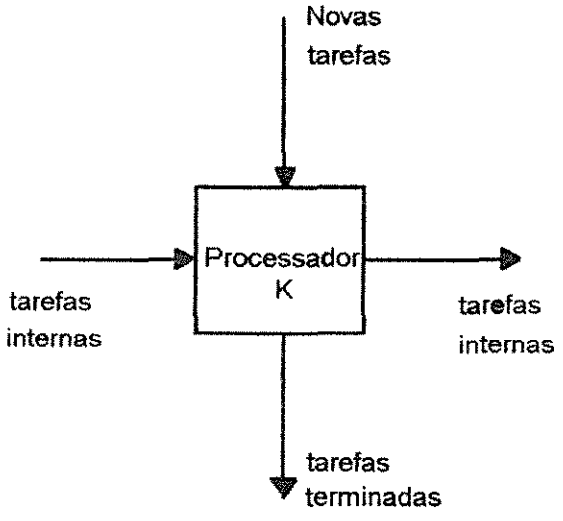


figura 2.4 - Planta Multipropósito Típica

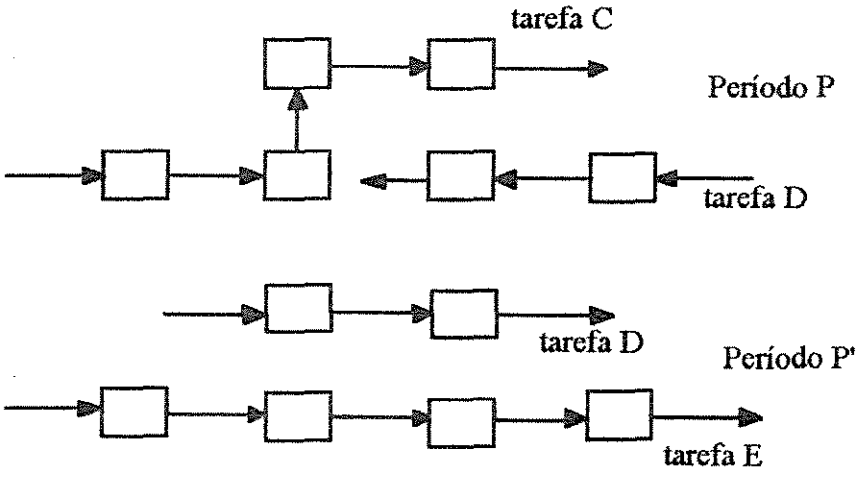


figura 2.5 - Tarefas desenvolvidas em uma Planta Multipropósito

A escolha da estrutura de produção depende de muitos fatores, sendo, principalmente, influenciada pelas características das tarefas a serem desenvolvidas e de suas quantidades.

2.3 - Caracterização das Operações batelada

2.3.1 - Definições Gerais

Grande parte dos processos batelada envolve mais de um estágio, cada um deste pode consistir de mais de um equipamento.

ESTÁGIO - O termo estágio está relacionado com a etapa ou fase corrente do processo.

Como mostram os autores Shah e Pantelides (1991) a estrutura de produção por estágios pode ser vista de forma macro e micro-estrutural. Na visão macro-estrutural somente são considerados as fases do processo ou estágio, como mostra figura 2.6a, enquanto que na visão micro-estrutural os equipamentos que compõem cada fase são detalhados, figura 2.6b.

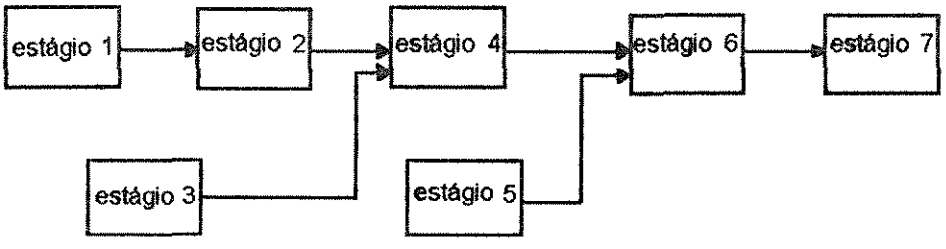


figura 2.6a - Representação Macro de uma Planta Batelada

Estágio 3

figura 2.6b - Representação Micro de uma Planta Batelada (Estágio 3)

No modelo mais simples de planta multiproduto, que é a que possui um equipamento em cada fase, cada um dos equipamentos constitui o mesmo estágio para todas as tarefas, tendo em vista que a seqüência de processamento é a mesma para todas as tarefas a serem desenvolvidas. Este resultado não é válido para plantas batelada multipropósito, onde produtos podem ser manufaturados por diferentes rotas.

No que tange à produção, cada equipamento está disposto em dimensões discretas.

DIMENSÃO DO EQUIPAMENTO - Na fase de Planejamento, a dimensão dos de cada equipamento é fixada, enquanto que na fase de Projeto esta dimensão é escolhida entre valores existentes.

A tabela 2.2 mostra algumas diferenças entre os equipamentos utilizados em plantas contínua e itermitente.

Tabela 2.2 - Diferenças entre equipamentos contínuos e itermitentes

Equipamento contínuo	Equipamento itermitente
- equipamento especialmente projetado	- Do tipo universal
- Poucos ajustes no equipamento	- Ajustes muito frequentes
- Necessidade de grande manutenção preventiva	- pouca manutencão
- Carga de trabalho de máquina é uniforme	- Carga de trabalho de máquina variada

Uma característica de cada equipamento é a sua dimensão S_{ik} ("size factor"), que especifica a produção em volume de uma unidade de massa da tarefa i no

equipamento k. Para a produção de uma batelada B_i(quantidade em massa da tarefa i), no equipamento k, é necessário uma dimensão de equipamento dada pela expressão 2.1.

$$V_{ik} = B_i S_{ik} \qquad (2.1)$$

2.3.2 - Processamento com equipamentos em paralelo

2.3.2.1 - Introdução

Um determinado estágio de uma planta batelada pode possuir mais de um equipamento, como mostrado na figura 2.6b, onde os equipamentos estão dispostos em série. Os equipamentos de um determinado estágio também podem estar dispostos de forma paralela, como mostra a figura 2.7.

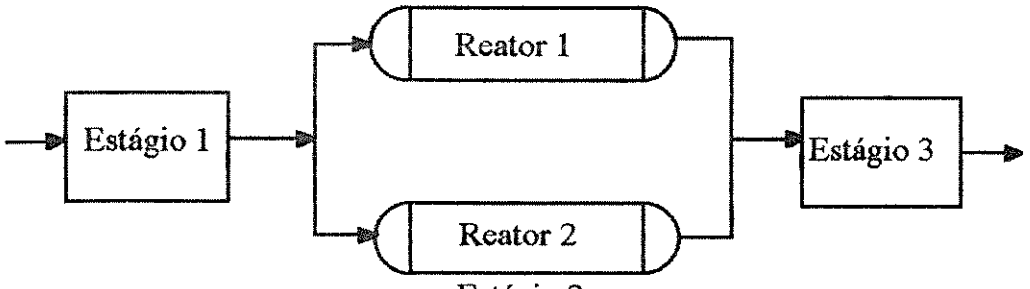


figura 2.7 - Estrutura Multiproduto com processsamento paralelo no estágio 2

O processamento com processadores em paralelo é recomendado quando a produção de uma determinada tarefa é limitada pelo maior equipamento disponível em planta já existente ou pelo maior equipamento que pode ser adquirido, na fase de projeto de uma nova planta. Um caminho econômico para contornar esta situação é a instalação de equipamentos operando em paralelo nos estágios críticos (gargalos). Neste caso, o material proveniente da operação anterior é dividido entre os equipamentos em disposição paralela, enquanto as suas saídas são combinadas e processadas em etapas posteriores. Processadores que operam em paralelo são classificados de acordo com seu modo de operação:

a) Processadores idênticos: neste caso as razões de processamento das tarefas são iguais;
b) Processadores uniformes: No caso de processadores operando de modo uniforme, a cada um dos processadores associa-se uma razão de processamento para todos os produtos, com isso o tempo de processamento t_{ik} da tarefa i no processador k é dado por:

$$t_{ik} = \frac{B_i}{R_{ik}} \qquad (2.2)$$

onde:

B_i = batelada necessária da tarefa i;
R_{ik} = razão de processamento da tarefa i no processador k

c) Processadores sem-relação: Nos processadores não-relacionados as unidades são dissimilares, deste modo, elas possuem razões de processamento diferentes para as tarefas diferentes.

Todas os casos acima são de interesse para os setores industriais, contudo, o caso com unidades idênticas é menos provável acontecer.

2.3.2.2 - Influência da Disposição dos Processadores nos Estágios

A influência da disposição dos equipamentos no processamento batelada é focalizada nesta secção, complementando a discussão feita anteriormente. Para ilustração desta influência, considere uma linha de processamento com os seguintes dados:

a) 3 estágios ou etapas de processamento;
b) cada estágio é composto por um único processador;
c) tempos de limpeza de 0,5 u.t, independentes do estágio
d) os tempos de processamento para as tarefas são dados na tabela 2.3.

Tabela 2.3 - tempos de processamento

Tarefa	Processadores		
	1	2	3
i	1.5	7.5	3.0

A análise desta influência é iniciada com o modo mais simples de operar uma planta, que é processar um produto em um determinado tempo, pois exige menos

trabalho em tempos similares (alternativa 1). A figura 2.8a mostra uma planta operando em modo sem-sobreposição de atividades ou "No-overlapping", neste caso, só é possível iniciar uma nova batelada do produto i depois que a batelada corrente termine seu processamento. O tempo limitante de ciclo ou "LCT" e o tempo de residência("flowtime") para este modo são iguais a 12 u.t.

"LIMITING CYCLE TIME" ou "LCT" - tempo gasto entre duas bateladas consecutivas de produtos.

"FLOWTIME" - tempo gasto para que uma batelada de produto seja processada em todos os estágios, avaliado a partir do início do processamento.

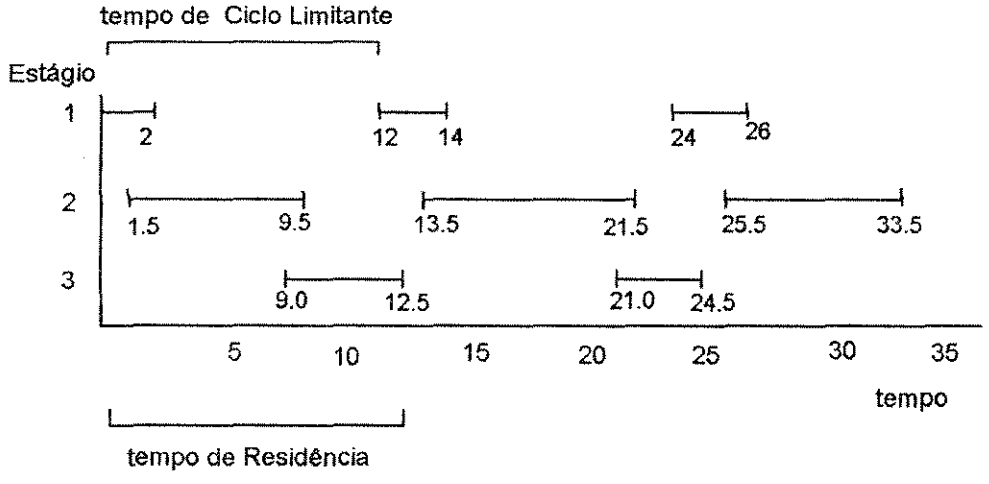


figura 2.8a - Planta Batelada operando sem sobreposição de trabalhos

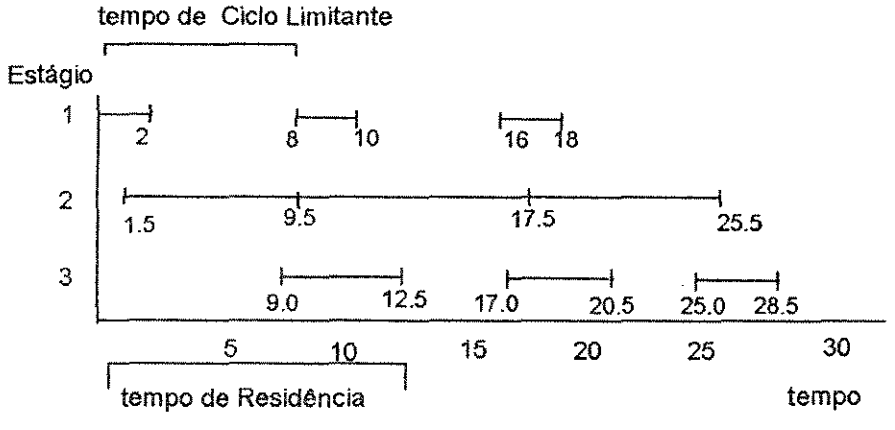


figura 2.8b - Planta Batelada operando com sobreposição de trabalhos

A figura 2.8b mostra a mesma planta operando com-sobreposição de atividades ou "overlapping" (alternativa 2). O tempo de ciclo limitante é condicionado pelo processamento no estágio 2 (estágio com maior tempo de processamento para o produto i). No modo com-sobreposição uma nova batelada do produto i é iniciada no estágio 1 em um tempo factível que coincida com o término da batelada no processador 2, com esta sobreposição de atividades, o "LCT" é reduzido para 8 u.t, no entanto, o tempo de residência permanece inalterado.

Com a incorporação de um novo processador no estágio crítico (estágio 2) operando em paralelo com o processador existente (alternativa 3) é possível reduzir o "LCT" para 4 u.t. Neste modo de operação, a saída do estágio 1 pode ser processada alternativamente por dois processadores diminuindo o intervalo de início de novas bateladas. A figura 2.8c representa esta configuração.

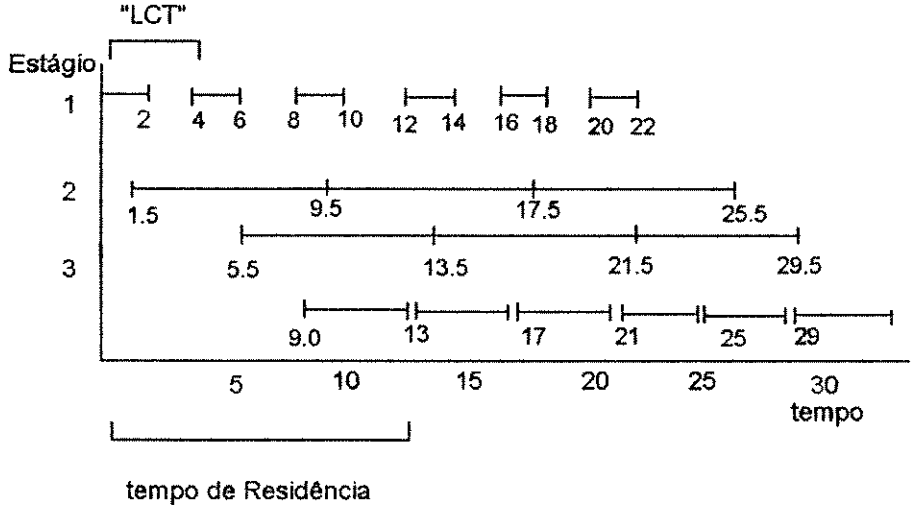


figura 2.8c - Planta Batelada com processamento paralelo no estágio 2

Pela análise realizada acima, o tempo de residência da tarefa i permaneceu inalterado nos três casos, no entanto, a utilização de uma das alternativas apresentadas é determinada pela quantidade desejada do produto(tarefa). A alternativa 3 apresenta o menor "LCT" , ou seja, um número maior de bateladas de i é produzido com a adoção do processamento em paralelo.

2.4 - Análise da influência da armazenagem intermediária em Plantas Batelada Multiproduto

Em plantas multiestágio, a armazenagem intermediária é um dos recursos que mais influencia na determinação de um programa de produção ótimo. Mah (1990) afirma que o escalonamento/seqüenciamento de produção é intensamente afetado pela natureza da armazenagem intermediária. É possível, numa planta de processamento, distinguir quatro tipos de armazenagem interestágios.

a) Armazenagem ilimitada ("Unlimited intermediate storage, UIS"): Esta política de armazenagem é a mais simples de ser tratada e a mais estudada na literatura existente. Parece ser este modo de armazenagem o passo inicial para a solução de outros problemas de maior complexidade em plantas batelada. Sobre a política "UIS" considera-se que um número ilimitado de tanques de armazenamento estejam disponíveis no período de execução das tarefas. Este modo de armazenagem é representado na figura 2.9a.
b) Sem armazenagem intermediária ("No intermediate storage, NIS"): Neste modo (figura 2.9b) não é permitido que haja armazenagem entre os diversos estágios, mas um determinado equipamento pode ser utilizado como tanque de armazenagem, retendo uma determinada batelada até que o próximo equipamento esteja pronto para receber a batelada de material.
c) Armazenagem limitada ("Finite intermediate storage, FIS") : Este modo de armazenagem (figura 2.9c) é intermediário entre o "UIS" e o modo "NIS", pois comporta-se como "UIS" , uma vez que bateladas de material podem ser armazenadas entre os equipamentos, mas só que existe limitação nesta armazenagem.

d) Tempo de armazenagem zero ("Zero wait, ZW"): O modo "ZW" é comum em indústrias de materiais instáveis, onde a batelada de material deve ser imediatamente processada. Este modo de armazenagem é mostrada na figua 2.9d.

Segundo Ku et al. (1987) e Mah (1990) a melhor política que descreve plantas batelada é a combinação destes modos de armazenágem. Uma linha de processamento operando sob esta combinação de modos armazenagens diz-se ter uma política de armazenagem mista, denominada na literatura de "MIS" ("Mixed intermediate storage"). Rajagopalan e Karimi (1989) têm dado atenção a esta política de armazenagem. Dada uma seqüência de tarefas numa linha de processamento "MIS" os autores mostram o cálculo dos tempos de execução de cada tarefa nos processadores.

A política de armazenagem adotada influencia quantitativamente na função objetivo, tendo em vista que as funções de desempenho são funções do tempo de execução das tarefas nos diversos equipamentos. Para exemplificação desta influência, são construídas Cartas de Gantt, para o problema apresentado na tabela 2.4, mostrando como o tempo total de execução é influenciado pela política de armazenagem adotada na planta.

Tabela 2.4 - tempos de Processamento do "flowshop"

tarefas	Processadores			
	1	2	3	4
1	10	20	5	30
2	15	8	12	10
3	20	7	9	5
4	13	7	17	10

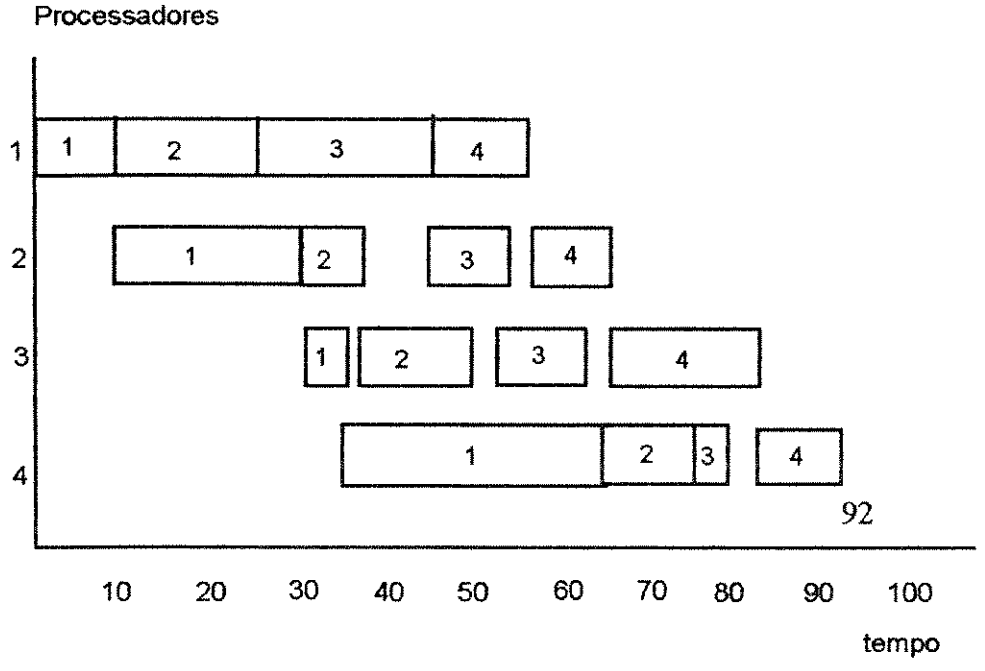


figura 2.9a - Armazenagem Itermediária Ilimitada

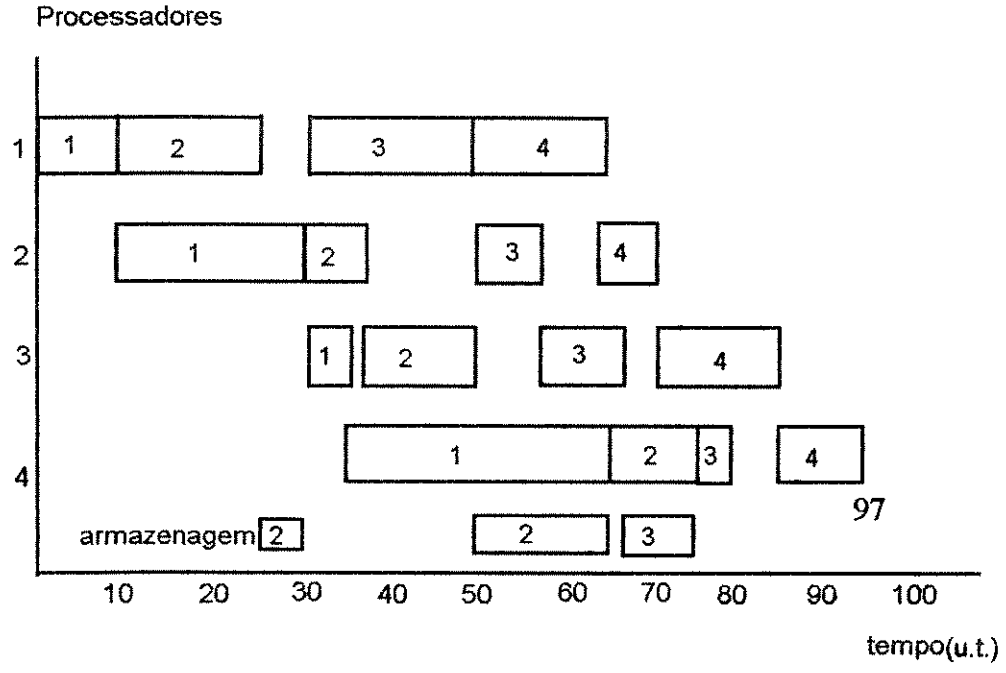


figura 2.9b - Armazenagem Intermediária Finita

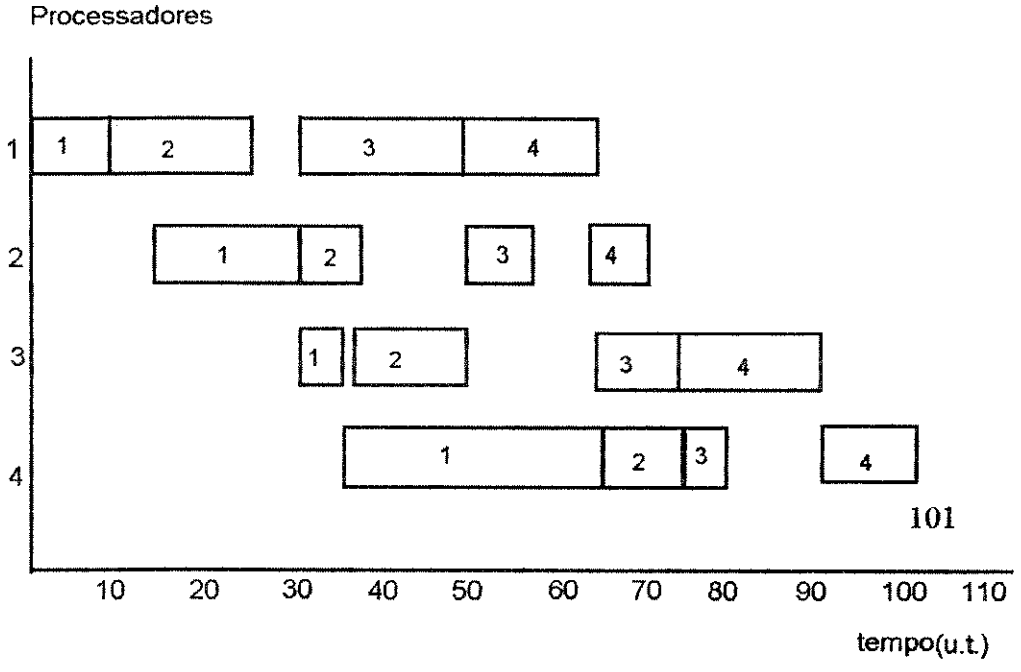


figura 2.9c - Sem Armazenagem Intermediária

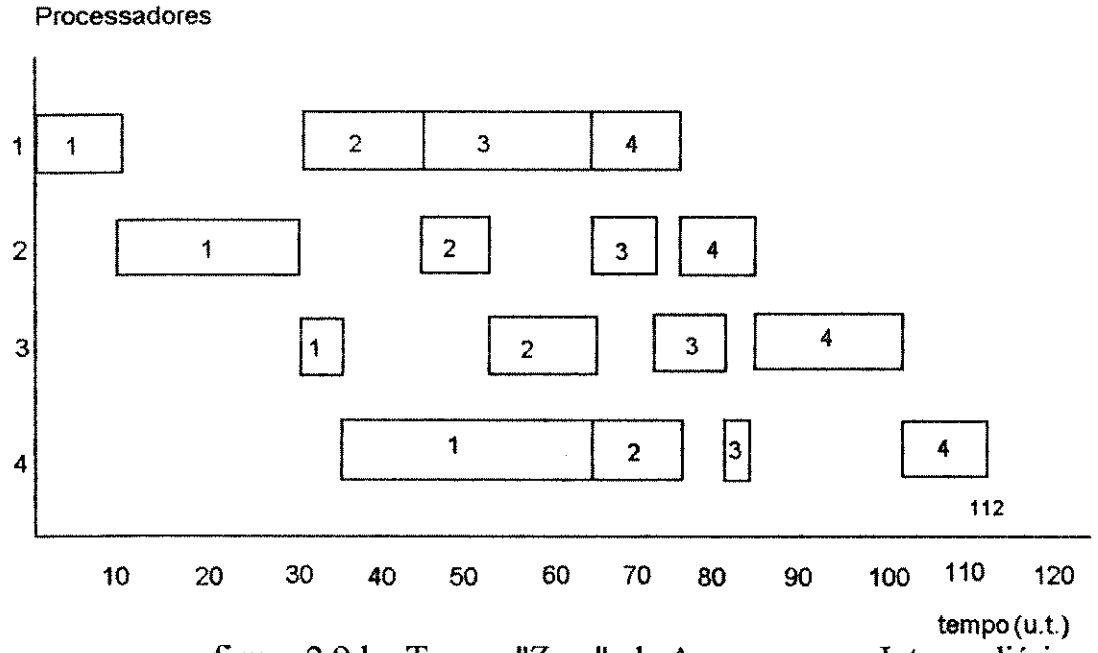


figura 2.9d - Tempo "Zero" de Armazenagem Intermediária

A capacidade de armazenagem intermediária é avaliada em termos de número de tanques disponíveis para armazenar as bateladas de material. Esta consideração encontra-se relacionada de certo modo com a decisão de se tomar temporariamente, na política "NIS", um equipamento como tanque de armazenagem.

2.5 - Planejamento de Produção em Plantas Batelada

2.5.1 - Introdução

O Planejamento de produção é direcionado à distribuição de recursos, assim como: tempo, equipamento, mão-de-obra, etc, de modo a atender as demandas de mercado de forma eficiente, no horizonte de produção estipulado.

A necessidade de planejar existe em todas os setores de produção que desejam melhorias no desempenho da organização como um todo. Esta necessidade relaciona-se com o fato de que as quantidades exigidas pelo mercado não correspondem com as taxas de processamento ótimo da planta e nem com os recursos disponíveis existentes. Esta crucialidade cresce ainda mais quando a planta opera sujeita às variações dinâmicas nas demandas das tarefas (demandas dos produtos variam durante o escalonamento), sendo crucial neste caso a flexibilidade das operações envolvidas.

2.5.2 - Planejamento Hierárquico de Produção

Reklaitis (1982) nota que o Planejamento total de produção pode ser tratado por duas abordadagens: uma é tratá-lo como um único plano(um só nível), onde são considerados simultaneamente o planejamento e o detalhamento do escalonamento. Na outra abordagem, um conjunto de subproblemas interligados é originado da decomposição do Planejamento. Na literatura consultada esta decomposição dá-se em dois planos. O primeiro nível é formado por um problema de longo ou médio prazo("upper level") e o segundo nível por um subproblema de curto prazo("lower level"). Este último é concernido aos detalhes da produção de "chão de fábrica", que requer agilidade na solução do problema. No planejamento de curto prazo, os parâmetros de produção são normalmente determinísticos, ou seja, estes parâmetros são invariáveis durante o processo.

2.5.2.1 - Planejamento de Produção a Longo e Médio Prazo

Apesar de sua importância em processos batelada este tipo de planejamento não vêm recebendo atenção suficiente na literatura existente (Ku, et al.,1987).

Rippin and Mauderli (1979 e 1980) apresentam um programa "Batchman" para o planejamento de produção de médio prazo (8 semanas a 1 ano) para plantas multipropósito. Tendo-se como dados: o número de equipamentos, suas capacidades, interconecções entre os equipamentos, demandas e receitas dos produtos. O programa determina quando e em que quantidades os produtos podem ser manufaturados.

2.5.2.2 - Planejamento de Produção a Curto Prazo

Em contraste com os outros tipos de planejamento, problemas envolvendo o planejamento a curto prazo está recebendo considerável atenção, isto porque o escalonamento em plantas batelada requer programas detalhados, em curtos intervalos de tempo. De um modo geral, os programas devem indicar a ordem de manufatura dos produtos e como os recursos tempo e equipamentos devem ser distribuídos nas diversas batelada. Neste tipo de problema estão envolvidos:

a) Um conjunto de N produtos ou bateladas de produtos a manufaturar;
b) M processadores disponíveis;
c) Uma seqüência de operações para cada produto

2.6 - Escalonamento de Produção em Processos Batelada

2.6.1 - Introdução

A importância do escalonamento de produção é evidenciada, principalmente ,em situações onde estudos na disponibilidade de recursos e equipamento já se encontram fixados. O planejamento e escalonamento em setores produtivos são direcionados à distribuição de recursos sobre o horizonte de produção para um conjunto de tarefas. Na prática, estas duas funções não são independentes. O planejamento estuda e determina as tarefas que devem ser desenvolvidas, assim como avalia as limitações de recursos, enquanto a função escalonamento designa a ordem de tarefas, procurando distribuí-las da melhor forma para atender uma derterminada função objetivo.

A necessidade de escalonar trabalhos não surge do tipo de processamento (contínuo ou não-contínuo), ou melhor, o escalonamento não é determinado pelo modo de operação dos processadores, pelas propriedades dos materiais a serem processados e pela estrutura da planta. A programação de produção surge, sim, pela necessidade de diminuição de custos na manufatura de multiprodutos (Reklaitis,1982).

2.6.2 - Escalonamento de Produção versus Síntese de Processos

Algumas decisões antecedem o problema de escalonamento de tarefas em plantas industriais, tais decisões estão relacionadas à fase de projeto de plantas. O projeto de plantas batelada nasce devido à necessidade de se desenvolver várias tarefas em uma única estrutura. As perguntas que surgem nesta fase são:

a) Que produtos serão manufaturados?
b) Qual a escala de produção?
c) Como os produtos serão manufaturados? Determinando assim as operações envolvidas.
d) Que estrutura de planta será adotada?

Sendo determinado o número e tipos de equipamentos a serem utilizados, preços de produtos, demandas, custos de materiais, mão-de-obra necessária, etc., parte-se para a determinação do roteiro ou programa ótimo de produção. Cabe ao escalonamento de produção responder as sequintes questões:

2.6 - Escalonamento de Produção em Processos Batelada

2.6.1 - Introdução

A importância do escalonamento de produção é evidenciada, principalmente ,em situações onde estudos na disponibilidade de recursos e equipamento já se encontram fixados. O planejamento e escalonamento em setores produtivos são direcionados à distribuição de recursos sobre o horizonte de produção para um conjunto de tarefas. Na prática, estas duas funções não são independentes. O planejamento estuda e determina as tarefas que devem ser desenvolvidas, assim como avalia as limitações de recursos, enquanto a função escalonamento designa a ordem de tarefas, procurando distribuí-las da melhor forma para atender uma derterminada função objetivo.

A necessidade de escalonar trabalhos não surge do tipo de processamento (contínuo ou não-contínuo), ou melhor, o escalonamento não é determinado pelo modo de operação dos processadores, pelas propriedades dos materiais a serem processados e pela estrutura da planta. A programação de produção surge, sim, pela necessidade de diminuição de custos na manufatura de multiprodutos (Reklaitis,1982).

2.6.2 - Escalonamento de Produção versus Síntese de Processos

Algumas decisões antecedem o problema de escalonamento de tarefas em plantas industriais, tais decisões estão relacionadas à fase de projeto de plantas. O projeto de plantas batelada nasce devido à necessidade de se desenvolver várias tarefas em uma única estrutura. As perguntas que surgem nesta fase são:

a) Que produtos serão manufaturados?
b) Qual a escala de produção?
c) Como os produtos serão manufaturados? Determinando assim as operações envolvidas.
d) Que estrutura de planta será adotada?

Sendo determinado o número e tipos de equipamentos a serem utilizados, preços de produtos, demandas, custos de materiais, mão-de-obra necessária, etc., parte-se para a determinação do roteiro ou programa ótimo de produção. Cabe ao escalonamento de produção responder as sequintes questões:

a) Quais os equipamentos a serem utilizados na manufatura de uma determinada tarefa? Neste caso, quando existe liberdade na escolha de utilização de equipamentos.
b) Qual a ordem de manufatura das terefas?
c) Quando cada estágio de processamento será utilizado?

2.6.3 - Colocação do Problema de Escalonamento de Produção

Segundo Graves (1981), três informações básicas devem ser consideradas na classificação de problemas de escalonamento de produção:

a) Necessidade de produção
b) Complexidade de processamento
c) Critério para escalonamento das tarefas

A necessidade de Produção é determinada diretamente de ordens de pedido ou indiretamente por decisões de reabastacimento de estoque . A depender do modo que os pedidos surgem tem-se os seguintes problemas de escalonamento:

a) Produção direta ("Open shop") - onde todas as ordens de produção são diretas do mercado;
b) Produção estocada ("Closed shop") - as ordens são indiretas, frutos da necessidade de manutenção de estoques.

Em um problema de produção direta, em sua forma mais simples , resulta um problema de seqüenciamento de tarefas, onde as ordens de produção são seqüenciadas nos equipamentos. No problema "closed shop" estão envolvidas decisões de seqüenciamento assim como decisões de estoque de produção ("lot-sizing") para o reabastecimento do estoque.

A complexidade de processamento está associada ao número de processadores envolvidos no processo, assim como a estrutura da planta. Esta complexidade está relacionada à dificuldadade de solução de problemas de escalonamento/seqüenciamento em plantas batelada.

O problema de escalonamento de produção é parte essencial para a solução do problema do planejamento total dentro de setores industriais. Este problema em plantas multiproduto, basicamente, apresenta a seguinte estrutura, estando a ordem de operações determinada:

a) Um conjunto de N tarefas ou produtos a serem processados;
b) Um conjunto de M processadores disponíveis para processar as tarefas;
c) tempos de processamento das tarefas em cada processador;
d) Política de armazenagem da planta;
e) Critério de desempenho ou função de custo.

Em alguns problemas a este conjunto de dados incorpora-se:
a) tempos de estabelecimento e de transferência;
b) Datas de entrega de produtos;
c) Restrições tecnológicas e de precedência de produtos;
d) Restrições de utilidades, de mão-de-obra, etc.

A complexidade de solução do problema depende de sua colocação, e pode-se ter casos em que encontrar um programa ótimo é praticamente impossível, buscando neste caso soluções aproximadas.

A classificação de problemas de escalonamento encontra-se ligada ao critério utilizado para avaliar o desempenho do roteiro de fabricação, pela estrutura de processamento e pelas regras de produção que gorvernam o processo (Reklaitis, 1982). A função de custo são geralmente de dois tipos: Critério baseado em custo real e Critério baseado em desempenho do sistema(custo indireto). O primeiro caso é comum em estruturas com um único processador, onde custos reais de produção podem ser isolados. Dentre os critérios de desempenho de sistema estão:
a) Minimização do tempo total de execução de todas as tarefas ou "makespan";
b) Minimização do máximo tempo de residência ou "flowtime";
c) Minimização do tempo de residência médio;
d) Minimização do máximo tempo de atraso ou "maximum tardiness";
e) Minimização do médio tempo de atraso ou "mean tardiness";
f) Minimização de custo sobre-mudança de produtos;

2.6.4 -Definições das variáveis e funções de desempenho mais utilizadas

A análise das principais váriáveis e funções de desempenho básicas no tratamento de problemas de escalonamento/seqüenicamento é vista em Baker (1974). Um problema de escalonamento tem como entrada principal :

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

b) r_i = instante em que a tarefa i está pronta para ser processada("ready time")

c) d_i = instante estabelecido para a entrega da tarefa i("due date")

A saída é basicamente o instante de término da tarefa i em um processador j, representado por Cij.

Os critérios de desempenho dafinidos a seguir são funções destas váriáveis.

Tempo de residência ou "Flow time" ($F_i = C_{iM} - r_i$): avalia o tempo em uma tarefa i permanece no processo;

"Lateness" ($L_i = C_i - d_i$): avalia o tempo que excede a data de entrega estabelecida para a tarefa i;

"Tardiness" (T_i): definido em função do "Lateness". Se $L_i>0$ ($T_i = L_i$) , tem-se um atraso positivo. Para $L_i \leq 0$ não existe atraso ($T_i = 0$).

Tabela 2.5 - Representação das funções objetivos

Funções de Desempenho	Representação
tempo de execução total das tarefas	C_{NM} N = última tarefa prosessada M = último processador
tempo de residência médio ou "Flowtime"	$F_{med} = \frac{1}{N}\sum_{i=1}^{N} F_i$
"tardiness" médio	$T_{med} = \frac{1}{N}\sum_{i=1}^{N} T_i$
tempo de residência máximo	$F_{max} = \{\max\{F_i\} \ / 1 \leq i \leq N\}$
"tardiness" máximo	$T_{max} = \{\max\{T_i\} \ / 1 \leq i \leq N\}$
Número de trabalhos atrasados	$N_T = \sum_{i=1}^{N} \delta(T_i)$
	$\delta(T_i)\begin{cases} = 1, & \text{se } T_i \geq 0 \\ = 0, & \text{caso contrário} \end{cases}$

A tabela 2.5 mostra as principais funções utilizadas na otimização de produção de um processo batelada Estas funções de desempenho são adotadas a depender das características do problema apresentado. Na vasta literatura existente a função "makespan" é um critério bastante utilizado. No caso de problemas envolvendo datas de entrega de produtos, a função objetivo "Tardiness" é conveniente para evitar penalidades para a indústria devido ao atraso de pedidos.

2.6.5 - Funções Regulares de Avaliação

As funções analisadas na secção 2.2.4 são todas dependentes do tempo de execução das tarefas nos processadores. Tomando-se F como sendo uma função de desempenho, então:

$$F = f(c_1, c_2, c_3, \ldots c_N) \tag{2.3}$$

onde Ci (i = 1,2,...N) é o tempo de execução da tarefa i no último processador.

Estas avaliações pertencem a uma classe de funções objetivo que são denominadas medidas regulares de desempenho. Uma função objetivo é regular se:
a) Durante o escalonamento, o objetivo é minimizar F, e;
b) O aumento de F está condicionado ao aumento do tempo de execução de, pelo menos, uma das tarefas.

Considerando F uma função de avaliação sobre o programa de produção S e $F' = f(c'_1, c'_2, c'_3, \ldots c'_N)$ como sendo a mesma função de desempenho sobre o programa S'. Então, F é dita regular se: o objetivo é minimizar F e se:

$$F' > F \Leftrightarrow C'_i > C_i \qquad \text{para alguma tafefa i} \tag{2.4}$$

O estudo de funções regulares é de extrema importância para o escalonamento de produção, pois estas permitem restringir o espaço de busca do programa ótimo dentro de um conjunto, denominado de conjunto "dominante". Para estabelecimento destes grupos, Baker (1974) mostra os passos que devem ser seguidos.

2.6.6- Escalonamento/Seqüenciamento em Estruturas Simples de Processamento

2.6.6.1 - Problema com um estágio e um único processador

O problema de escalonamento em uma única máquina é tido como o caso mais simples de escalonamento de produção. Graves (1981) cita vários trabalhos envolvendo este modelo, assim como o algortimo por Jackson (1955) para a minimização do tardiness. Baker (1974) também mostra alguns resultados relacionados a esta estrutura de processamento. No capítulo 4, o problema com uma única máquina é abordado no seqüenciamento de tarefas com tempos de estabelecimento não negligenciáveis.

2.6.6.2- Problema com um único estágio e processadores em paralelo

Dentre os procedimentos direcionados à solução de problemas de seqüenciamento nesta estrutura, tem-se:

a) Procedimento de McNaughton

O ponto mais relevante de escalonamento de produção, neste modelo, é a possibilidade de operação sem suspenção("preemption"). Um tipo de operação com corte de tempo de processamento é mostrado na figura 2.10, denominado de ocupação prévia-retomada("preempt-resume") envolvendo o mesmo processador. No caso de processadores paralelos idênticos, não existe o problema de escolha de qual processador utilizar para terminar o processamento da respectiva tarefa., uma vez que os equipamentos são iguais.

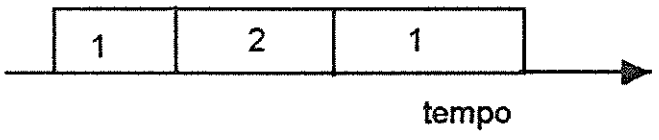


figura 2.10 - Desenvolvimento de tarefas com ocupação prévia-retomada

Um importante resultado para o processamento em paralelo com máquinas idênticas foi obtido por McNaughton (1959), onde o tempo total mínimo de execução das tarefas é dado por:

$$\max[\max(t_i), \frac{1}{M}\sum_{i=1}^{N} t_i] \qquad (2.5)$$

onde t_i é o tempo de processamento da tarefa i no estágio.

Como exemplo de aplicação da regra de McNaughton, considere a tabela de tempos de processamento dados na tabela 2.6. Os tempos de estabelecimento (tempos gastos para preparar a máquina para o processamento de uma nova tarefa) são independentes da seqüência, além de não existir nenhuma restrição de qualquer natureza .

Tabela 2.6 - Tempos de processamento das tarefas(único estágio)

tarefas	1	2	3	4	5	6	7	8
tempo	8	7	6	5	4	3	2	1

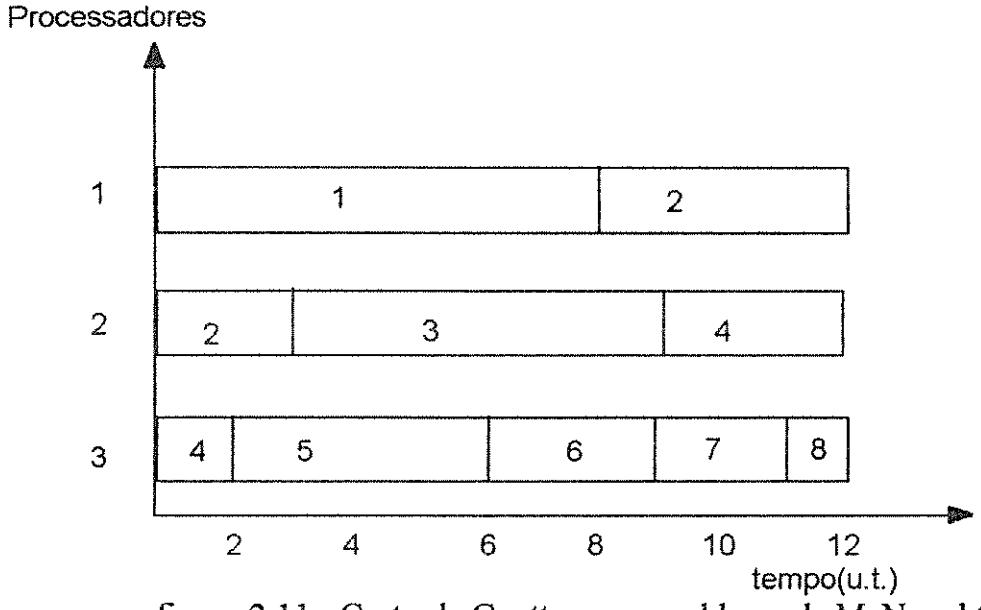


figura 2.11 - Carta de Gantt para o problema de McNaughton

Pela expressão 2.5 o melhor programa de produção tem um "makespan" de 12 u.t. A figura 2.11 mostra a distribuição das tarefas obtida pelo o algoritmo de McNaughton.

b) Heurística tempo de Processamento mais Longo ("Longest Processing Time" ou "LPT").

No caso de operações sem pré-ocupação, os problemas com processadores idênticos podem ser tratados pela heurística "LPT" , em que o produto com maior tempo de manufatura vai sendo designado no processador que apresentar menor tempo total de processamento. Considerando a tabela 2.6, a heurística "LPT" é aplicada visando a minimização do "makespan". A figura 2.12 mostra o programa de manufatura dos produtos obtido por esta heurística.

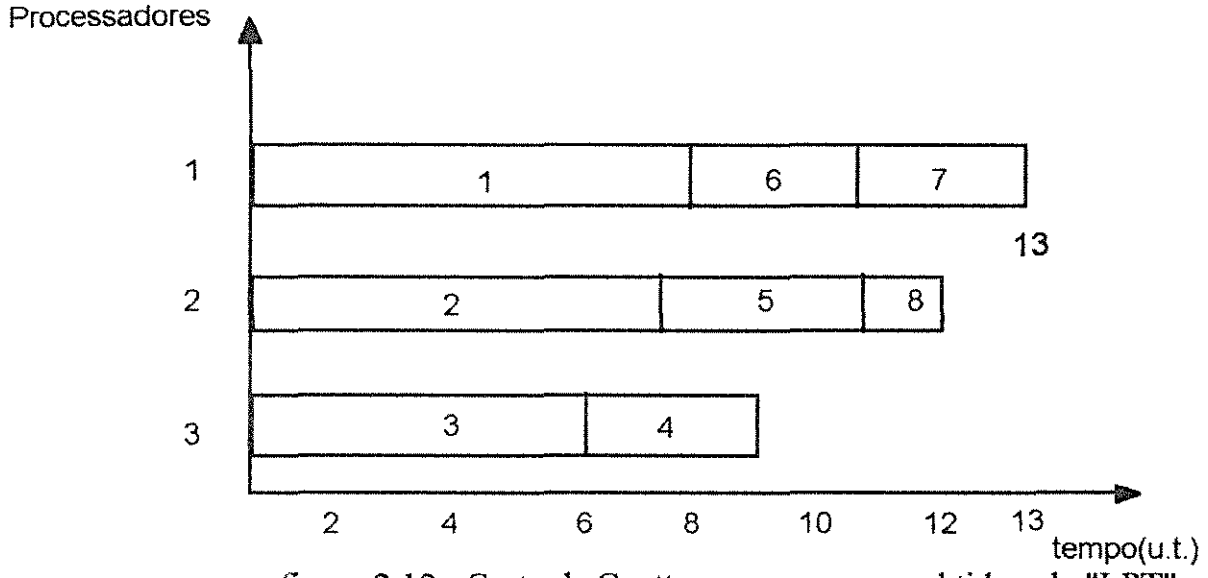


figura 2.12 - Carta de Gantt para o programa obtido pela "LPT"

Nota-se que o tempo total de execução obtido pela regra de McNaughton é inferior ao obtido pela heurística "LPT". Na literatura consultada não há citações que esta situação sempre ocorre.

Uma análise dos piores casos feita por Graham et al. (1974) mostra que o "makespan" obtido pela "LPT" tem um erro limitado pela expressão abaixo:

$$1 \leq \frac{\text{" makespan " (LPT)}}{\text{" makespan " otimo}} \leq \frac{4}{3} - \frac{1}{3M} \qquad (2.6)$$

Para o caso de três processadores em paralelo o "makespan" obtido está no máximo 23% acima do ótimo.

c) Heurística tempo de Processamento mais Curto ("Shortest Processing Time" ou "SPT")

Esta heurística apresenta um raciocínio contrário ao da "LPT" e também pode ser utilizada visando a minimização do tempo de execução total. McNaughton (1959) mostra que este procedimento também é conveniente na minimização do tempo de residência médio. Neste método heurístico, o processadore com menor tempo total de processamento é escolhido para receber a tarefa com menor tempo de manufatura. Considerando os tempos de processamento da tabela 2.6 a heurística "SPT" é aplicada determinando um programa de produção , representado pela figura 2.13.

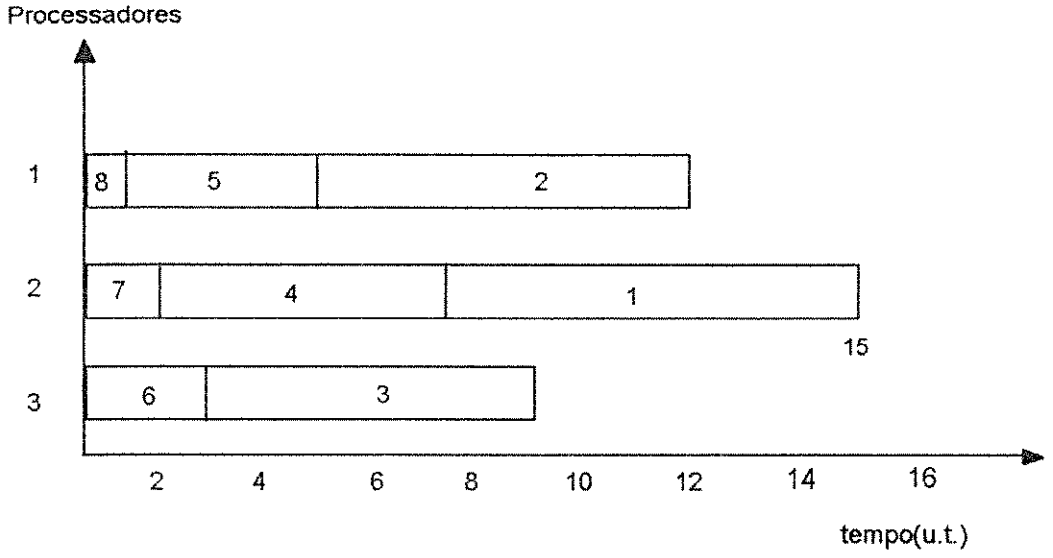


figura 2.13 - Carta de Gantt para o programa obtido pela "STP"

Tabela 2.7 - Trabalhos existentes nas diversas formas de processamento paralelo

tipos de processadores	"Makespan"		"Flowtime" Médio	
	"pre-empt"	"No-pre"	"pre-empt"	"No-pre"
Idêntica	McNaughton (1959)	"NP-complete"	"SPT"	"SPT"
Uniforme	Gonzales-Sahni (1978)	"NP-complete"	Gonzales (1977)	"Extended SPT"
Não relacionada	LP (Lawler & Labetoulle, 1978)	"NP-complete"	"NP-complete"	"Transportation problem (Bruno et al., 1974)

Reklaitis (1982) relaciona as diversas formas de trabalho com processadores em paralelo e que são apresentadas na tabela 2.7.

2.6.7 - Metodologias para resolução de problemas Linha de Processamento

2.6.7.1 - Introdução

Basicamente, existem duas técnicas eficientes para resolver problemas de programação de produção: tratar o problema por meio de uma formulação Linear Inteira-Mista ("Mixed integer linear"), ou adotar a estratégia de enumeração implícita (estratégia de busca controlada "Branch and Bound, BAB").

2.6.7.2 - Enumeração Implícita (Medologia "Branch and Bound" ou "BAB")

A enumeração implicíta compreende uma busca coordenada e heurística (avaliação do menor limitante ou "lower bound") no espaço de todas soluções possíveis, para o problema em estudo. A natureza combinatória dos problemas de linha de processamento inviabiliza a análise do conjunto global de todas as soluções possíveis, tendo-se de fazer uma busca implícita da solução ótima. Nilson (1971) aborda alguns métodos de redução de espaço, dentre eles a pesquisa "BAB" em profundidade ou "Depth first" e a pesquisa em largura ou "Breadth first". Na proposta de tese, sempre que se recorrer à metodologia "BAB", a busca em profundidade é utilizada.

A Metodologia "BAB" apresenta as seguintes etapas:

a) Etapa ou Procedimento "Branching": toma o problema original e então gera um conjunto de subproblemas, que são mutuamente exclusivos e parcialmente solúveis;

b) Procedimento "Bounding": calcula o menor limitante da função de desempenho de cada subproblema.

Representando um problema original por V^0 seus subproblemas são mostrados na figura 2.14. No plano 1 da árvore de busca encontram-se as seqüências originadas pela designação das tarefas para a primeira posição. Logo, para a formação do primeiro plano existem N possibilidades possíveis, gerando assim N nós mestres ($V_1^1, V_2^1 \ldots V_N^1$). O subproblema V_1^1 tem a tarefa 1 designada para a posição 1 no plano 1 da árvore. Uma vez que a primeira posição encontra-se fixada N-1 tarefas devem ainda ser escalonadas. Cada subproblema gerado é novamente decomposto, gerando o segundo

nível. O subproblema V_2^1, por exemplo, é tomado como nó mestre e ramificado originando os nós $V_{21}^1, V_{23}^1, \ldots, V_{2N}^2$. Quando se adota a estratégia de busca em profundidade esta ramificação prossegue até a base da árvore, gerando uma solução completa ou limitante superior. Depois de alcançada a base o incremento de retrocesso ("backtraking") é colocado em prática em busca de subproblemas não solucionados e que são promissores à minimização da função desempenho.

O procedimento "bounding" depende de dois termos: o menor limitante da função desempenho associado à solução de cada subproblema e, no caso de se utilizar uma pesquisa em profundidade, da solução inicial ou solução tentativa. Uma solução tentativa pode ser calculada a partir de uma seqüência de permutação inicial, mas é preferível que se utilize uma heurística de seqüenciamento para se obter este valor. Uma Solução inicial e um menor limitante fortes reduzem a busca da solução ótima, em contrapartida, o tempo computacional cresce.

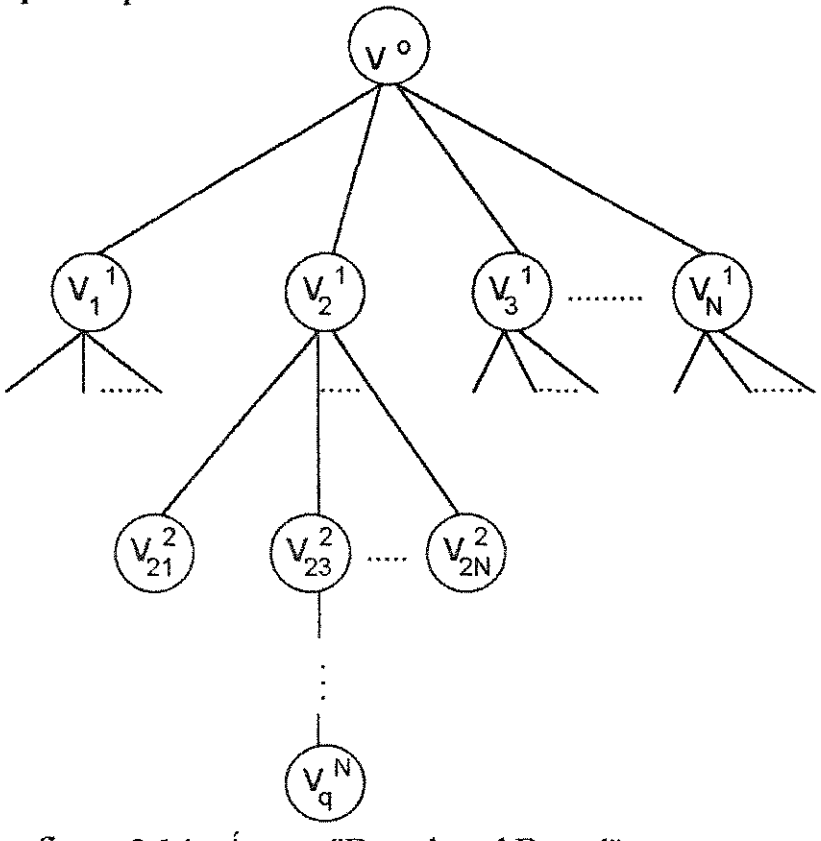


figura 2.14 - Árvore "Branch and Bound"

O procedimento "bounding" é crucial para que a estratégia de enumeração implícita forneça à solução ótima sem ser necessário a geração de todas a sequências, a princípio factíveis. Para a redução ou eliminação (processo de sondagem ou "fathoming") do espaço de busca é necessário que o menor limitante seja estimado com garantia, evitando assim que regiões factíveis não sejam eliminadas durante a busca.

2.6.7.3 - Programação Linear Inteira-Mista(método"MILP")

A programação linear inteira-mista é uma caso específico da programação inteira-mista, em que a função objetivo e restrições são lineares. A metodologia "MILP" para problemas de escalonamento é desenvolvida em cima das fórmulas de recorrência para o cálculo do tempo de execução das tarefas. Tendo-se uma política "UIS" (armazenagem intermediária ilimitada) o problema linha de processamento pode ser estruturado como segue:

$$x_{ij} = \begin{cases} 1, \text{se a tarefa i é escalonada na posição j da seqüência} \\ 0, \text{ caso contrário} \end{cases} \qquad (2.7)$$

Considerando o tempo de execução total das tarefas como critério de desempenho e usando Cij como tempos de execução das seqüências parciais, então a função objetivo é minimizar C_{NM} , que é o tempo de execução da última tarefa no último processador.

$$\text{função Objetivo: Minimizar } C_{NM} \qquad (2.8)$$

O primeiro conjunto de restrições assegura que uma tarefa somente é designada para um processador, ocupando uma posição da seqüência.

$$\sum_{j=1}^{N} x_{ij} = 1 \qquad i = 1, 2, \ldots, N \qquad (2.9)$$

Por último, tem-se as restricões que asseguram que em cada posição só poderá estar escalonada uma tarefa.

$$\sum_{i=1}^{N} x_{ij} = 1 \qquad j = 1, 2, \ldots, N \qquad (2.10)$$

As equações que surgem das relações para uma linha de processamento "UIS" são:

estágio 1:

$$C_{i1} - C_{i-1,1} - \sum_{k=1}^{N} t_{k1} x_{ki} \geq 0 \qquad i = 1, 2, \ldots, N \qquad (2.11)$$

estágios J= 2 a M

$$C_{ij} - C_{i-1,j} - \sum_{k=1}^{N} t_{kj} x_{ki} \geq 0 \qquad i = 1, 2, \ldots, N \qquad (2.12)$$

$$C_{ij} - C_{i,j-1} - \sum_{k=1}^{N} t_{kj} x_{ki} \geq 0 \qquad i = 1, 2, \ldots, N \qquad (2.13)$$

$$Cij = 0, \quad i \leq 0 \qquad (2.14)$$

onde Cij é o tempo de execução das seqüências parciais.

A formulação dada pelas expressões 2.7 a 2.14 constitui o método "MILP" para um problema linha de processamento com armazenagem ilimitada, que pode ser solucionado por pacotes comerciais disponíveis. A limitação da dimensão do problema é de NXM ≤ 30 (Ku et. al.,1987) e Mah (1990).

2.6.7.4 - Algoritmo de Johnson

No seqüenciamento de tarefas envolvendo 2 processadores com armazenagem ilimitada, cujo o critério de desempenho é o tempo total de execução das tarefas, utiliza-se o algoritmo desenvolvido por Johnson (Ku et al.,1987). Este procedimento é de ordem polinomial.

Pela regra de Johnson uma seqüência é ótima se:

$$\min\{t_{i1}, t_{j2}\} \leq \min\{t_{i2}, t_{j1}\} \qquad (2.15)$$

Considerando um conjunto de N tarefas, o algoritmo de Johnson pode ser aplicado para seqüenciá -las, encontrando desta forma um roteito ótimo de manufatura. Este algoritmo é constituído dos seguintes passos:

a) Determine tij = min{tij, i=1,2,...N};

onde:

j = processador 1 e 2;
i = tarefa escalonada;
tij = tempo de processamento da tarefa i no processador j;

b) Se j =1, a tarefa é escalonada o mais cedo possível, retirando-a da fila de espera;
c) Se j = 2, a tarefa é designada para o final da fila, e em seguida é retirada do conjunto U.

Como exemplo de aplicação do procedimento de Jonhson, considere tabela 2.8 com os tempos de processamento das tarefas.

Tabela 2.8 - Tempos de processamento das tarefas

tarefa	Processador	
	1	2
1	3	2
2	5	6
3	1	2
4	6	6
5	7	5

Estágio	U	min tij	Sequência parcial
1	{1,2,3,4,5}	t31	3 X X X X
2	{1,2,4,5}	t22	3 X X X 2
3	{1,4,5}	t11	3 1 X X 2
4	{4,5}	t52	3 1 X 5 2
5	{4}	t41=t42	3 1 4 5 2 *

*seqüência ótima: 3-1-4-5-2
tempo de execução = 27 u.t.

A grande importância deste algoritmo reside no fato que mesmo sendo os problemas de escalonamento de produção de natureza combinatorial, este procedimento contorna o problema de maneira polinomial, ou seja, o tempo de execução é uma função polinomial da constante N do problema.

2.6.7.5 - Regras Heurísticas de Seqüenciamento

Métodos heurísticos são procedimentos aproximados muitas vezes utilizados em problemas de escalonamento/seqüenciamento. Ku et al. (1987) tratam estas regras como sendo métodos ótimos para soluções iniciais utilizadas em métodos mais exatos. Segundo estes autores métodos como as estratégias "BAB" e "MILP" produzem soluções superiores quando a solução inicial é obtida por métodos heurísticos. É importante também acrescentar que a diminuição do tempo de busca da solução quando se utiliza tais métodos pode ser bastante considerável. A maior parte das heurísticas desenvolvidas e apresentadas na literatura incorpora a regra de Johnson, citada anteriormente. Dentre os procedimentos heurísticos para o seqüenciamento de tarefas, tem-se:

a) Heurística CDS

Dado os tempos de processamento dos diversos produtos nos diferentes estágios, este procedimento aplica o algoritmo de Johnson M-1 vezes, gerando pseudo-problemas com 2 processadores. Os tempos fictícios para os estágios são dados por:

estágio 1:

$$t_{i1} = \sum_{k=1}^{j} t_{i,k} \quad j = 1, 2, \ldots, M - 1 \qquad (2.16)$$

$$i = 1,2,\ldots,N$$

estágio 2:

$$t_{i2} = \sum_{k=1}^{j} t_{i,(M-k)+1} \quad j = 1, 2, \ldots, M - 1 \qquad (2.17)$$

$$i = 1,2,\ldots,N$$

onde:
N = número de tarefas;
M = número de estágios;
tij = tempo de processamento da terefa i no processador j;

A escolha da melhor seqüência é feita por comparação direta do tempo de execução das seqüências geradas.

Exemplo de aplicação: Considerando a tabela 2.8, a heurística "CDS" é aplicada para encontrar a melhor seqüência de execução das tarefas.

Tabela 2.9 - tempos de processamento das tarefas

tarefa	estágio			
	1	2	3	4
1	25	36	75	10
2	52	68	44	96
3	18	73	61	7
4	80	11	28	30

subproblema 1

Estágio	tarefa			
	1	2	3	4
1	25	52	18	80
2	10	96	7	30

sequência ótima: 2-4-1-3
tempo de execução: 336 u.t.

subproblema 2

estágio	tarefa			
	1	2	3	4
1	61	120	91	91
2	85	140	68	58

sequência otima: 1-2-3 4
tempo de execução: 337 u.t.

subproblema 3

Estágio	tarefa			
	1	2	3	4
1	136	164	152	119
2	121	208	141	69

sequência otima: 2-3-1-4
tempo de execução: 387 u.t.

melhor sequência via heurística: 2-4-1-3
tempo de execução: 336 u.t.

sequência ótima do problema original: 1-2-4-3
tempo de execução: 322 u.t.

b) Heurística RAES

Esta heurística gera pseudos-problemas com 2 estágios, sendo aplicado posteriormente o algoritimo de Jonhson. Os tempos fictícios para os 2 estágios são gerados com pesos diferentes.

estágio 1:

$$t_{i1} = \sum_{j=1}^{M}((M-1)+1)t_{ij} \qquad (2.18)$$

estágio 2:

$$t_{i2} = \sum_{j=1}^{M} jt_{ij} \qquad (2.19)$$

Uma vez obtida uma seqüência, esta heurística aplica, posteriormente, o algoritmo de troca de pares adjascentes como tentativa de melhorar o roteiro, como mostrado em Baker (1974).

2.6.7.6 - Gráfico de Gantt para acomplanhamento de trabalhos

Este tipo de técnica é as vezes utilizada em setores produtivos para estabelecer e representar a programação de produção. Numa visão mais ampla e concisa o Gráfico de Gantt mostra a distribuição temporal de tarefas e recursos utilizados para o desenvolvimento destas tarefas. Esta metodologia é praticamente inviável em processos onde detalhes de processamento, como por exemplo: manutenção preventiva de equipamento, restrições de recursos, etc., são comuns e inevitáveis durante a produção, pois o tempo para a construção e manipulações destes gráficos cresce à medida que aumenta o número de trabalhos e complexidade na planta. A figura 2.17 a distribuição temporal de utilização de 3 recursos para 3 tarefas.

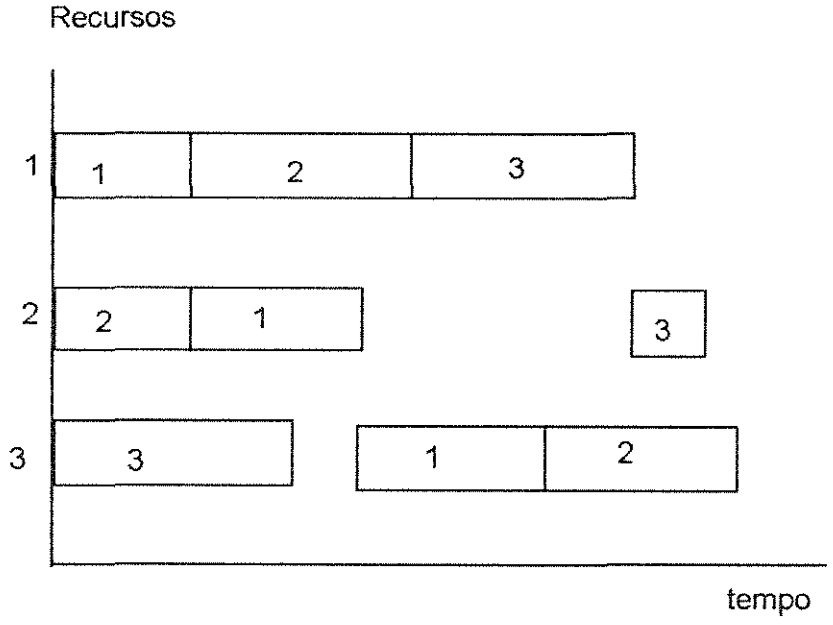


figura 2.15 - Distribuição temporal de recursos

2.7 - Conclusão

Neste capítulo, foram mostradas as principais características presentes no processamento batelada, focalizando, principalmente, o processamento serial de produtos (plantas multiproduto). Devido à sua alta flexibilidade de produção, o escalonamento de tarefas, nesta estrutura, é uma ferramenta indispensável para a coordenação das atividades. Dentre os métodos citados neste capítulo, o método de busca implícita "Branch and Bound" foi escolhido, isto devido à sua constante aplicação na literatura existente, principalmente, quando o número de tarefas envolvidas não é superior a 12. Para a aplicação da metodologia "BAB" é necessário a construção de uma função de custo que

contemple detalhes do processamento, pois a efetividade de redução do espaço de busca está relacionada com a avaliação dos custos dos nós da árvore. Na literatura são citadas várias funções objetivos. A escolha de uma função ou mais funções que irão servir para avaliação do sistema deve está ligada às características do processamento e da indústria em questão. Em nossa abordagem, o "makespan" ou tempo total de execução das tarefas é tomado com função de desempenho, pois quanto maior o tempo para o desenvolvimento das tarefas, o custo de produção aumenta.

CAPÍTULO 3 - SEQÜENCIAMENTO DE TAREFAS EM PLANTAS MULTIPRODUTO

3.1 - Introdução

Em um problema de seqüenciamento, a determinação da ordem em que as tarefas são desenvolvidas já define o plano de produção. Quando, neste problema, somente tempos de processamento são considerados, tem-se o caso mais simples de seqüenciamento de produção.

De acordo com a literatura existente um problema de seqüenciamento apresenta as seguintes características:

a) Não existe problema de distribuição de recursos para o desenvolvimento das tarefas;

b) A política de armazenagem é do tipo "UIS", "ZW" ou "NIS" (ver item 2.4);

c) A oferta de recursos compartilhados é ilimitada

3.2 - Ordem de execução de tarefas

Em um problema de seqüenciamento com N tarefas, um roteiro ótimo de execução destas tarefas pode ser uma seqüência de permutação (seqüência em que a ordem de execução é a mesma em todos os processadores) dos N inteiros representantes das tarefas. O conjunto das N! seqüências é um subconjunto das $N!^M$ seqüências possíveis de serem executadas. Neste trabalho, somente são consideradas seqüências de permutação, limitando com isso o espaço de busca de uma solução.

Um estudo de propriedades "dominantes" para a redução do espaço de busca é tratado por Baker (1974) para avaliações regulares, os resultados obtidos, apresentados a seguir são de extrema importância na teoria de escalonamento/seqüenciamento.

a) Para M = 2 e para qualquer avaliação regular, as seqüências de permutação formam um conjunto dominante, e somente elas devem ser consideradas na busca do roteiro ótimo.

b) Para M = 2 e M = 3 e tendo como função de desempenho o tempo de execução total das tarefas, as seqüências de permutação também formam um conjunto dominante.

A figura 3.1 mostra que nem sempre seqüências de permutação fornecem o melhor roteiro de manufatura. A título de exemplicação considere a tabela 3.1 de tempos de processamento, algumas Cartas de Gantt são construídas mostrando este resultado.

Tabela 3.1- Tempos de Processamento

Tarefas	Processadores			
	1	2	3	4
1	1	5	5	2
2	5	1	1	5

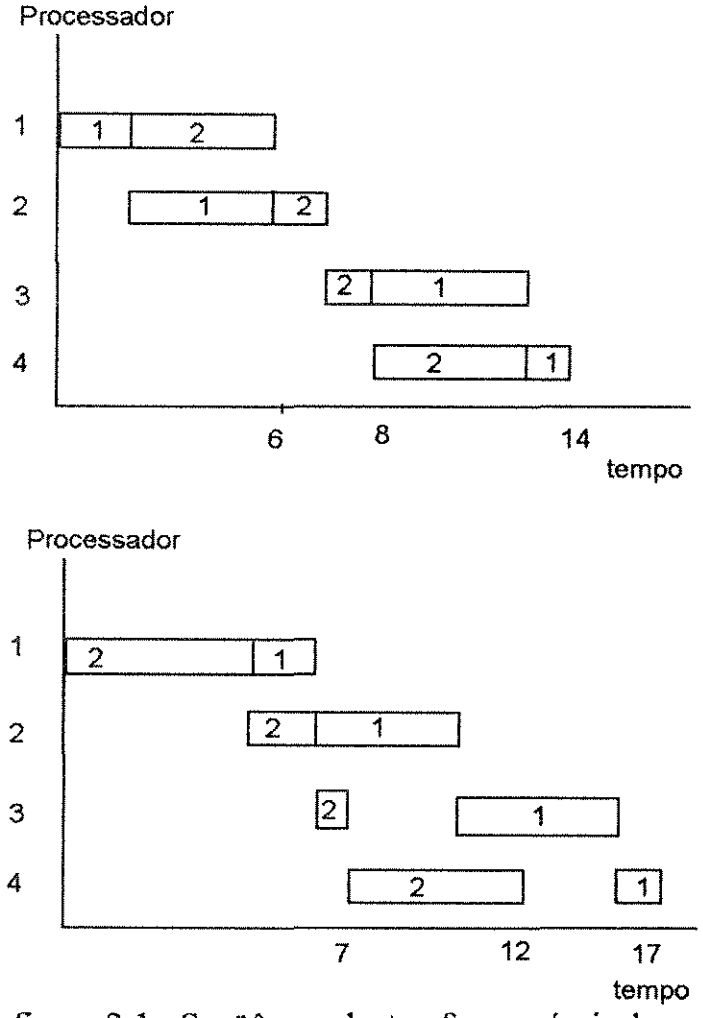


figura 3.1 - Seqüências de tarefas possíveis de serem executadas

3.3 - Análise do tempo de execução de Problemas de Programação de Produção

Mesmo sendo o espaço de busca limitado pelas seqüências de permutação, os problemas de Programação de produção em linhas de processamento são da classe "NP-complete" (Garey et. al. ,1976). O tempo necessário para solucionar estes problemas aumenta exponencialmente com a dimensão N do problema. A tabela 3.2 mostra dados comparativos de tempos necessários para solucionar problemas de escalonamento com diferentes dimensões, utilizando busca explícita (enumeração completa) e algoritmos polinomiais. Os dados mostram a impossibilidade de se pesquisar todas as soluções possíveis, sendo necessário a adoção de procedimentos que diminuam a busca de uma solução ótima.

Tabela 3.2 - tempos comparativos de solução de problemas linhas de processamento (Maria Tereza M. Rodrigues)

N	10	15	20	40
lnN	1s	1.8s	2.6s	6.4
N^2	1s	2.25s	4s	16s
N!	0.1s	10h	2000 anos	-

3.4 - Metodologia "Branch and Bound" aplicada ao seqüenciamento de tarefas

3.4.1 - Seqüenciamento de produção com tempos de estabelecimento independentes da ordem de execução das tarefas.

3.4.1.1 - Introdução

A utilização da metodologia de busca em árvore exige o cálculo de um menor limitante da função de desempenho. Os métodos de busca em árvore diferem em relação ao modo, pelo qual, é tomado o nó a ser expandido e pela heurística utilizada para calcular o menor limitante.

No cálculo do menor limitante estão envolvidas duas parcelas:

a) A primeira corresponde ao cálculo do tempo já gasto na execução das tarefas programadas ou seqüenciadas.
b) A segunda parcela equivale ao cálculo do tempo futuro a ser gasto para a execução das tarefas não-seqüenciadas.

A soma destas duas parcelas resulta no menor limitante (limitante inferior) da função objetivo.

A tabela 3.3 apresenta heurísticas para o cálculo do limitante inferior em tempo de execução total. Sendo:

a) C_{ij} : instante de término da tarefa i no processador j;
b) q : seqüência parcial;
c) LB: limitante inferior

Tabela 3.3 - Expressões heurísticas para o cálculo do limitante inferior em tempo de execução total

1 - "Simple Machine Based Bound"

$$LB_1 = C(q,M) + \sum_{i \in M} t_{iM}$$

2 - "Full Machine Based Bound"

$$bj = C(q,j) + \sum_{i \in U} t_{ij} + \min_{i \in U} \{ \sum_{k=j+1}^{M} t_{ik} \} \qquad (1 \le j \le M-1)$$

$$b_M = C(q,j) + \sum_{i \in U} t_{ij}$$

$$LB_2 = \max_{1 \le j \le M} \{bj\}$$

3 - "Job Based Bound"

$$d_j = C(q,j) + \max_{i \in U} \{ \sum_{k=j}^{M} t_{ik} + \sum_{\substack{l \in U \\ l \neq i}} \min[t_{ij}, t_{lM}] \}$$

$$LB_3 = \max_{1 \le j \le M-1} \{d_j\}$$

4 - "Non Interface Bound"

$$C_i^* = C(k,i) + \sum_{j=1}^{M-1} t_{ij}$$

C^* = instante "potencial" em que a tarefa i pode ser iniciada no processador M. Estes instantes são ordenados em ordem crescente.

$v_i = \max\{C_i^*, v_{i-1}\} + t_{iM}$, com $v_o = 0$

v_i = instante mais certo que a tarefa i poderia ser terminada no processador M

$$LB_4 = \max_{i \in U} \{v_i\}$$

Um estudo comparativo de algoritmos de seqüenciamento em uma Linha de processamento utilizando a metodologia "BAB" é mostrado em Baker (1975). A análise foi feita com base no tempo de resposta do algoritmo para problemas com diferentes dimensões. Para a avaliação do limitante inferior foram utilizadas as heurísticas da tabela 3.3 e algoritmos combinados.

A eficiência da metodologia "BAB" está associada diretamente ao cálculo do limitante inferior da função objetivo. Cada menor limitante calculado está associado a um nó mestre da árvore de busca, indicando que nenhum nó gerado a partir deste nó mestre possui custo menor.

3.4.1.2- Relações de Recorrência para uma linha pura processamento

A discussão tratada, nesta seção, é focalizada em três diferentes suposições:
a) São consideradas somente seqüências de permutação;
b) tempos de estabelecimento e de transferência negligenciáveis;
c) Política de armazenagem ilimitada

O seqüenciamento de tarefas pode ser visto como dois problemas interligados. O primeiro é a determinação da ordem em que as tarefas são executadas e, dado um roteiro de fabicação, o segundo problema é o cálculo dos instantes finais de cada uma das tarefas em cada processador. Para uma linha de processamento com as suposições acima citadas, o instante Cij factível para que uma nova tarefa possa ser iniciada na máquina j é dado pela expresão 3.1.

$$C_{ij} = \max\{C_{i,j-1}; C_{i-1,j}\} + tij \qquad (3.1)$$

$$C_{i0} = C_{0j} = 0 \qquad (3.2)$$

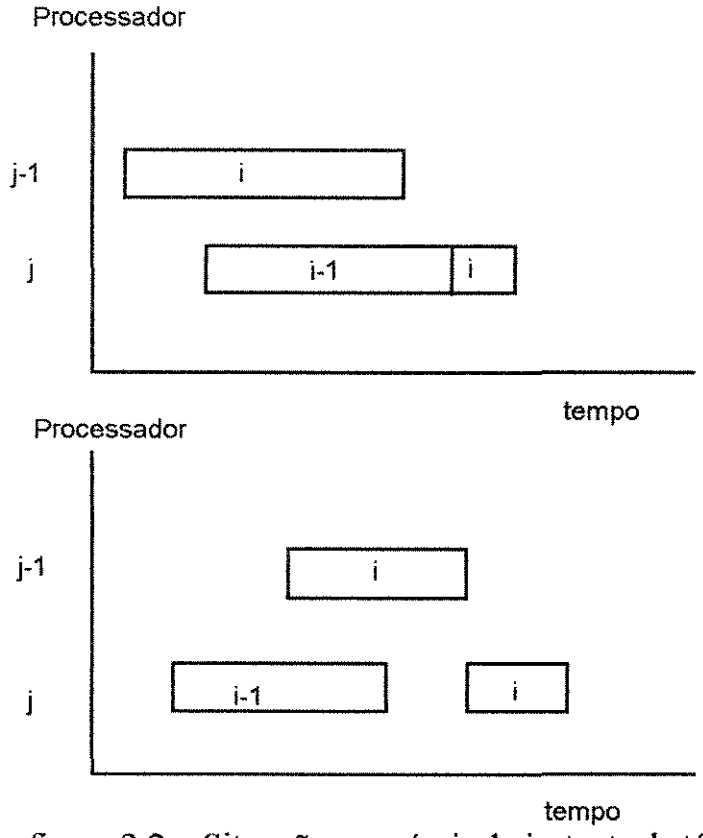


figura 3.2 - Situações possíveis do instante de término de uma tarefa em uma linha de processamento "UIS"

Estas relações de recorrência podem tomar as seguintes formas:

$$C_{(qi,j)} = \max\{C_{(qi,j-1)}; C_{(q,j}\} + tij \qquad (3.3)$$

$$C_{i0} = C_{0j} = 0 \qquad (3.4)$$

A figura 3.2 mostra as situações possíveis de instante de término de uma tarefa em uma Linha de processamento "UIS".

3.4.1.3 - Algoritmo para a minimização do tempo de execução total via metodologia "BAB" em uma linha de processamento "pura"

A chave para a eficiência da técnica "BAB" é desprezar as ramificações que não conduzem a um valor ótimo da função objetivo. Ignall e Scharage (1965) desenvolveram um procedimento para avaliar a produditividade ótima em uma planta

com N tarefas e M processadores utilizando a heurística "Full Machine Based Bound" para avaliação do menor limitante. Esta heurística é composta por três parcelas:

a) Instante menor factível em que se pode começar uma nova tarefa no processador j, Cij;
b) Carga de trabalho restante a ser processada no processador j, Rj

$$Rj = \sum_{i \in U} t_{ij} \qquad (3.5)$$

c) Tempo mínimo de trabalho supondo que o processamento em j já está encerrado, Uj.

$$Uj = \min_{i \in U} \{ \sum_{k=j+1}^{M} t_{ik} \} \qquad (3.6)$$

Então, o tempo mínimo bj para processar todas as tarefas a partir de cada processador j é dado por:

$$b_j = C_{ij} + R_j + U_i \quad \text{para } j = 1,2,...M \qquad (3.7)$$

O limitante inferior para cada seqüência parcial então é dado por:

$$LB = \max_j \{b_j\} \quad \text{para } j=1,2,...M \qquad (3.8)$$

Exemplo de Aplicação:

Considere uma linha de processamento com 3 processadores, 4 tarefas e tempos de processamento dados na tabela 3.4, determinar o roteiro de manufatura ótimo.

Tabela 3.4 - Tempos de Processamento

tarefas	Processadores		
	1	2	3
1	3	4	10
2	11	1	5
3	7	9	13
4	10	12	2

Como ilustração é mostrado o cálculo do limitante inferior da seqüência parcial qi = 1. Primeiramente é necessário o cálculo do instante de término, Cij, desta

seqüência em cada processador. Através das equações 3.3 e 3.4 este instante Cij pode ser calculado como segue:

$$C_{1,1} = \max\{0, 0\} + 3 = 3$$

$$C_{1,2} = \max\{3, 0\} + 4 = 7$$

$$C_{1,3} = \max\{7, 0\} + 10 = 17$$

O cálculo da carga mínima de trabalho, Rj, é dada pela equação 3.5,

$$R_1 = 11 + 7 + 10 = 28$$

$$R_2 = 1 + 9 + 12 = 22$$

$$R_3 = 5 + 13 + 2 = 20$$

Através da equacão 3.6, o tempo mínimo ainda necessário na Linha de processamento levando em conta que o processamento em j já esteja terminado é:

$$U_1 = \min\{6, 22, 14\} = 6$$

$$U_2 = \min\{5, 13, 2\} = 2$$

Calculando-se o tempo mínimo necessário para que o processamento das tarefas termine a partir de um processador j:

$$b_1 = 3 + 28 + 6 = 37$$

$$b_2 = 7 + 22 + 2 = 31$$

$$b_3 = 17 + 20 = 37$$

O limitante inferior para a seqüência parcial qi= 1- é:

$$LB = \max\{37, 31, 37\} = 37$$

Tabela 3.5 - Nós sondados na aplicação do algoritmo

sequência parcial	(C_1,C_2,C_3)	(b_1,b_2,b_3)	LB
1	(3, 7,17)	(37,31,37)	37
2	(11,12,17)	(45,39,42)	45
3	(7, 16, 29)	(37,35,46)	46
4	(10,22,24)	(37,41,52)	52
12	(14,15,22)	(45,38,37)	45
13	(10,19,32)	(37,34,39)	39
14	(13,25,27)	(37,40,45)	45
132	(21,22,37)	(45,36,39)	45
134	(20,32,34)	(37,38,39)	39
1342	(31,33,39)	(31,33,39)	39

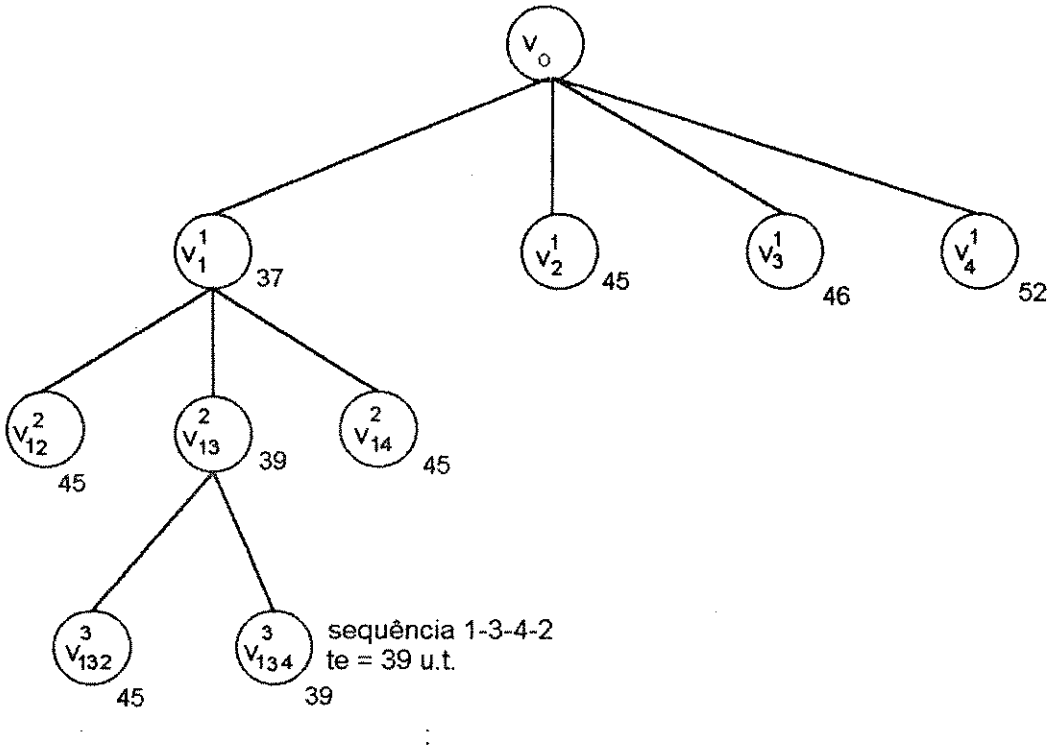


figura 3.3 - Árvore de busca para o problema

3.4.2 - Seqüenciamento com tempos de estabelecimento dependentes da seqüência de produção

3.4.2.1 - Introdução

Em muitos problemas realísticos o tempo para estabelecer ou preparar um processador depende da ordem em que as tarefas são programadas, consequentemente, o tempo de execução total para a produção de todas as tarefas é notoriamente afetado pelo roteiro de fabricação praticado. O seqüenciamento com tempos de estabelecimento ("setup times") é comum em plantas batelada, as quais por natureza manufaturam vários produtos, como por exemplo: indústrias de tintas, detergentes e misturas de óleos, indústria têxtil. Nestes setores de produção os tempos de preparação de máquina são significantes e devem ser considerados no tratamento do problema.

3.4.2.2 - Seqüenciamento envolvendo um único processador

Quando a estrutura da planta possui um único processador, a minimização do tempo total de execução das tarefas equivale a minimização do tempo total de preparação (Baker 1974), uma vez que o somatório dos tempos de processamento das tarefas é constante. Com esta característica, este problema assemelha-se ao problema clássico da otimização: Problema do Caixeiro Viajante ("Traveling Salesman Problem" ("TSP")). Este problema de natureza combinatorial, um caixeiro deve planejar uma viagem de modo que visite cada cidade, somente uma única vez, e retorne para a cidade de partida. O objetivo é determinar o custo de percurso mínimo de viagem. A figura 3.4 representa o problema "TSP".

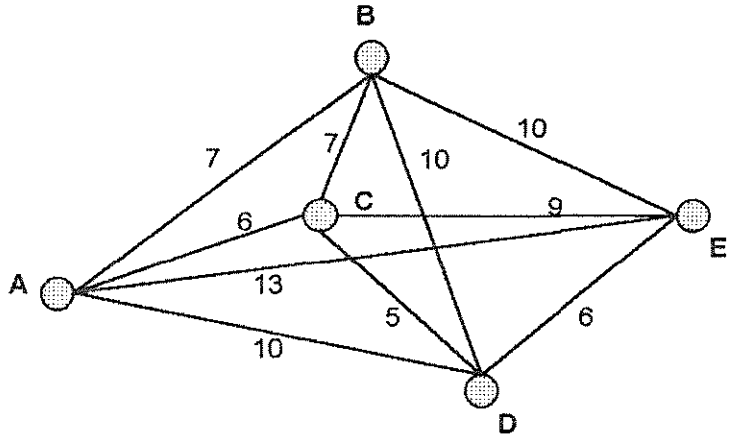


figura 3.4 - Representação do Problema do Caixeiro Viajante

Esta representação também pode ser feita por meio de uma matriz simétrica., como mostra a figura 3.5.

	A	B	C	D	E
A	-	7	6	10	13
B	7	-	7	10	10
C	6	7	-	5	9
D	10	10	5	-	6
E	13	10	9	6	-

figura 3.5 - Matriz correspondente ao problema "TSP"

Um procedimento direto e simples para a solução deste problema com N cidades é verificar todas as rotas possíveis e por comparação direta escolher uma rota mínima. Claramente, existem (N-1)! rotas e a distância total percorrida em cada rota envolve N adições, totalizando N!. Para problemas de grande dimensões, por exemplo se N= 50, o número de adições é de 10^{64}, e considerando um computador capaz de realizar 10^{9} adições por segundo, o tempo total para solucionar o problema é de 10^{45} séculos, somente para executar as adições. Portanto, é inviável tratar problemas de natureza combinatorial sem o desenvolvimento de métodos que limitem a região de busca de uma solução ótima.

Quando tempos de estabelecimento são considerados, a cada tarefa processada é necessário preparar o processador para receber uma nova tarefa, e este tempo gasto depende do par de tarefas envolvido na operação de troca de batelada.

Considerando S a matriz de tempos de preparação de cada par de tarefas , t_i o tempo de processamento da tarefa i e Ci o tempo para execução da tarefa i no único processador, tem-se que:

$$C_1 = S_{0-1} + t_1 \qquad (3.9)$$

$$C_2 = S_{1-2} + t_2 + C_1 \qquad (3.10)$$

$$C_3 = S_{2-3} + t_3 + C_2 \qquad (3.11)$$

para a tarefa N-1, tem-se:

$$C_{N-1} = S_{N-2,N-1} + t_{N-1} + C_{N-2} \qquad (3.12)$$

para a tarefa n, tem-se:

$$C_N = S_{N-1,N} + t_N + C_{N-1} \qquad (3.13)$$

O tempo para terminar a n-ésima tarefa é obtido somando-se as expressões 3.9 a 3.13, resultando a expressão 3.14.

$$C_N = S_{0-1} + S_{1-2} + S_{2-3} + \ldots + S_{N-1,N} + t_1 + t_2 + t_3 + \ldots t_N \qquad (3.14)$$

Considerando o estado final do processo n+1, a expressão 3.14 pode ser escrita na forma:

$$C_{N-1} = S_{N-2,N-1} + t_{N-1} + C_{N-2} \qquad (3.12)$$

para a tarefa n, tem-se:

$$C_N = S_{N-1,N} + t_N + C_{N-1} \qquad (3.13)$$

O tempo para terminar a n-ésima tarefa é obtido somando-se as expressões 3.9 a 3.13, resultando a expressão 3.14.

$$C_N = S_{0-1} + S_{1-2} + S_{2-3} + \ldots + S_{N-1,N} + t_1 + t_2 + t_3 + \ldots t_N \qquad (3.14)$$

Considerando o estado final do processo n+1, a expressão 3.14 pode ser escrita na forma:

$$te = \sum_{i=1}^{N+1} S_{i-1,i} + \sum_{i=1}^{N} t_i \qquad (3.15)$$

onde:

te = tempo de execução total das tarefas

$S_{i-1,i}$= tempo para preparar a máquina para a tarefa i depois que a tarefa anteriormente processada, i-1, termina seu processamento.

O segundo somatório da equação 3.15 é constante, concluindo-se que para se obter um tempo de execução mínimo é necessário minimizar os tempos de preparação total do processador.

Os procedimentos utilizados na minimização do percurso do Problema do Caxeiro Viajante (PCV) são igualmente utilizados para o tratamento do problema de Seqüenciamento Dependente dos Tempos de Estabelecimento (problema "SDTE"). Os procedimentos existentes na literatura para a solução deste problema são:
a) "Branch and Bound aproximada"
b) Programação Dinâmica aproximada
c) Método Heurístico

A - Método "Branch and Bound" - Algoritmo B

A técnica "BAB" aplicada na solução do problema PCV foi estudada por Little et al. (1963). O menor limitante para um determinado par de tarefas são calculados pelo método de redução da matriz de tempos de estabelecimento (S). O método tem como base a análise de dois subproblemas:

a) o primeiro subproblema analisa a presença de um par de tarefas na formação da seqüência ótima,
b) o segundo subproblema (subproblema complementar) analisa a influência da ausência deste par na seqüência.

A.1 - Etapas do Método

a) Etapa de Redução : Pela subtração das linhas do seu valor mínimo , a matriz original S é reduzida. O processo é repetido para as colunas que não apresentarem zeros após a redução das linhas. O menor limitante inicial ,θ_i, é calculado pelo somatório destes valores mínimos. A matriz S_r (matriz reduzida) possui obrigatoriamente um elemento nulo em cada linha e coluna. Estes elementos nulos correspondem aos pares de tarefas igualmente promissores, nesta etapa, a formarem uma seqüência ótima.

b) Etapa "Branching" (ramificação): No processo de ramificação, cada elemento nulo presente em S_r é rotulado por λ, que representa a soma do menor valor da linha e coluna do elemento nulo considerado. O problema original é dividido em dois subproblemas. O primeiro toma o par de tarefas representado pelo elemento nulo de maior rótulo como parte da seqüência ótima, e o outro faz com que este esteja ausente na seqüência . Se a

esta ausência resultar em uma matriz que não possui as características de S_r (pelo menos um elemento nulo em cada linha e coluna) deve-se proceder uma nova redução.

c) Eliminação dos pares infactíveis: Uma vez escolhido um par i-k para fazer parte da seqüência, o par k-i, assim como os pares k-l (l= 1,2...N e $l \neq k$) e os pares l-i (l= 1,2...N e $l \neq i$) são eliminados de S_r.

c) Etapa "Bounding" : O procedimento de redução novamente é colocado em prática. O menor limitante LB^* para os novos subproblemas é calculado pela soma de θ_l do subproblema anterior e o custo da redução corrente θ.

A.2 - Exemplo do método

A tabela 3.6 mostra a matriz de tempos de estabelecimento para um problema Linha de processamento com um único processador, cujo objetivo é minimizar tempo total de processamento. Como mostra a fórmula 3.15, para se conseguir este propósito basta minimizar o tempo total de preparação.

Tabela 3.6 - Tempos de Estabelecimento

Tarefas	Tarefas				
	1	2	3	4	5
1	-	4	8	6	8
2	5	-	7	11	13
3	11	6	-	8	4
4	5	7	2	-	2
5	10	9	7	5	-

Solução:

Etapa 1 -Redução inicial

$$\begin{bmatrix} - & 4 & 8 & 6 & 8 \\ 5 & - & 7 & 11 & 13 \\ 11 & 6 & - & 8 & 4 \\ 5 & 7 & 2 & - & 2 \\ 10 & 9 & 7 & 5 & - \end{bmatrix} \begin{matrix} 4 \\ 5 \\ 4 \\ 2 \\ 5 \end{matrix} \qquad \underline{\text{redução 1}} \qquad \begin{bmatrix} - & 0 & 4 & 2 & 4 \\ 0 & - & 2 & 6 & 8 \\ 7 & 2 & - & 4 & 0 \\ 3 & 5 & 0 & - & 0 \\ 5 & 4 & 2 & 0 & - \end{bmatrix}$$

Cálculo do custo da redução inicial

$$\theta_i = 4+5+4+2+5 = 20 \text{ (limitante inferior)}$$

Etapa 2 (Ramificação): rotular os elementos nulos com a soma do valor mínimo da linha e coluna do elemento nulo.

$$S_{r1} = \begin{vmatrix} - & 0^4 & 4 & 2 & 4 \\ 0^6 & - & 2 & 6 & 8 \\ 7 & 2 & - & 4 & 0^2 \\ 3 & 5 & 0^2 & - & 0 \\ 5 & 4 & 2 & 0^4 & - \end{vmatrix}$$

Formação dos subproblemas

a) subproblema 1: tomar o par de tarefas 2-1(apresenta maior rótulo λ) como par promissor à minimização do tempo total de estabelecimento. Eliminar pares infactíveis e aplicar o processo de redução.

subproblema P_{12}

$$S_{r1} = \begin{vmatrix} - & - & 4 & 2 & 4 \\ - & - & - & - & - \\ - & 2 & - & 4 & 0^2 \\ - & 5 & 0^2 & - & 0 \\ - & 4 & 2 & 0^4 & - \end{vmatrix} \begin{matrix} 2 \\ \\ \\ \\ \\ 2 \end{matrix}$$

<u>reduçao 2</u>

$$S_{r2} = \begin{vmatrix} - & - & 2 & 0^2 & 2 \\ - & - & - & - & - \\ - & 0^2 & - & 4 & 0^0 \\ - & 3 & 0^2 & - & 0^0 \\ - & 2 & 2 & 0^2 & - \end{vmatrix}$$

Calcular o menor limitante para o subproblema: $LB^* = \theta_i + \theta = 20 + 4 = 24$

b) suproblema 2 (subproblema complementar): analisar a ausência do par 1-2 na seqüência.

subproblema $P_{\cdots}$
$_{21}$

$$S_{r1} = \begin{vmatrix} - & 0 & 4 & 2 & 4 \\ - & - & 2 & 6 & 8 \\ 7 & 2 & - & 4 & 0 \\ 3 & 5 & 0 & - & 0 \\ 5 & 4 & 2 & 0^4 & - \\ 3 & & & & \end{vmatrix}^2 \quad \underline{\text{redução}} \quad S_{r2} = \begin{vmatrix} - & 0 & 4 & 2 & 4 \\ - & - & 0 & 4 & 6 \\ 4 & 2 & - & 4 & 0 \\ 0 & 5 & 0 & - & 0 \\ 2 & 4 & 2 & 0^4 & - \end{vmatrix}$$

$LB^* = \theta_i + \theta = 20 + 5 = 25$

A partir da matriz S_{r2} do subproblema P_{12} procede-se uma nova ramificação. Nota-se que neste estágio vários elementos são indicadores de uma seqüência com custo mínimo de estabelecimento. A escolha do par de tarefas, neste ponto, é arbitrária. Designando o par de tarefas 5-4 , chega-se aos seguintes subproblemas:

subproblema $P_{21,54}$

$$S_{r2} = \begin{bmatrix} - & - & 0 & - & 0 \\ - & - & - & - & - \\ - & 0 & - & - & 0^0 \\ - & 3 & 0 & - & 0^0 \\ - & - & - & - & - \end{bmatrix}$$

$LB^* = \theta_i + \theta = 25 + 1 = 26$

subproblema $P_{21,\overline{54}}$

$$S_{r2} = \begin{bmatrix} - & - & 2 & 0 & 2 \\ - & - & - & - & - \\ - & 0 & - & 4 & 0^0 \\ - & 3 & 0 & - & 0^0 \\ - & 0 & - & 0 & - \end{bmatrix}$$

$LB^* = \theta_i + \theta = 25 + 1 = 26$

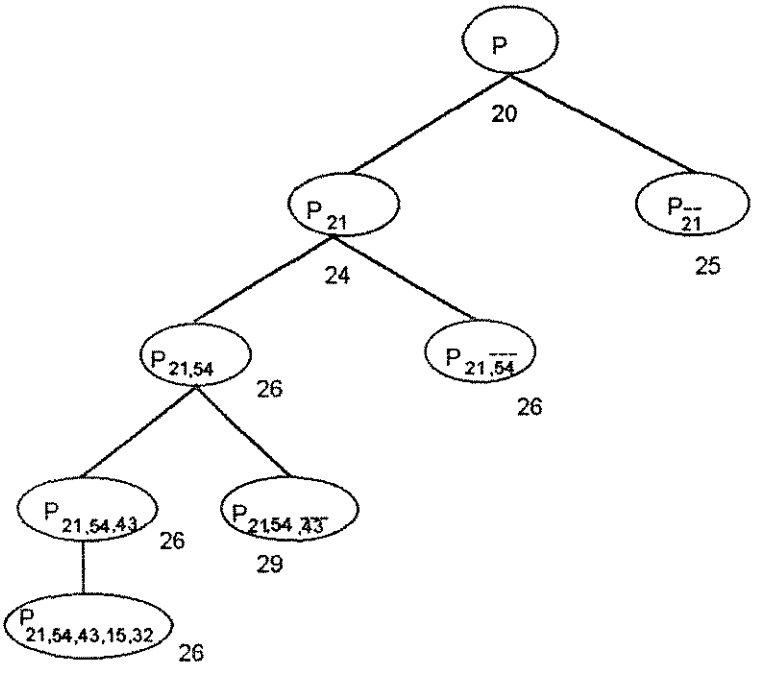


figura 3.5 - Árvore de busca do limitante superior do problema "SDTE"

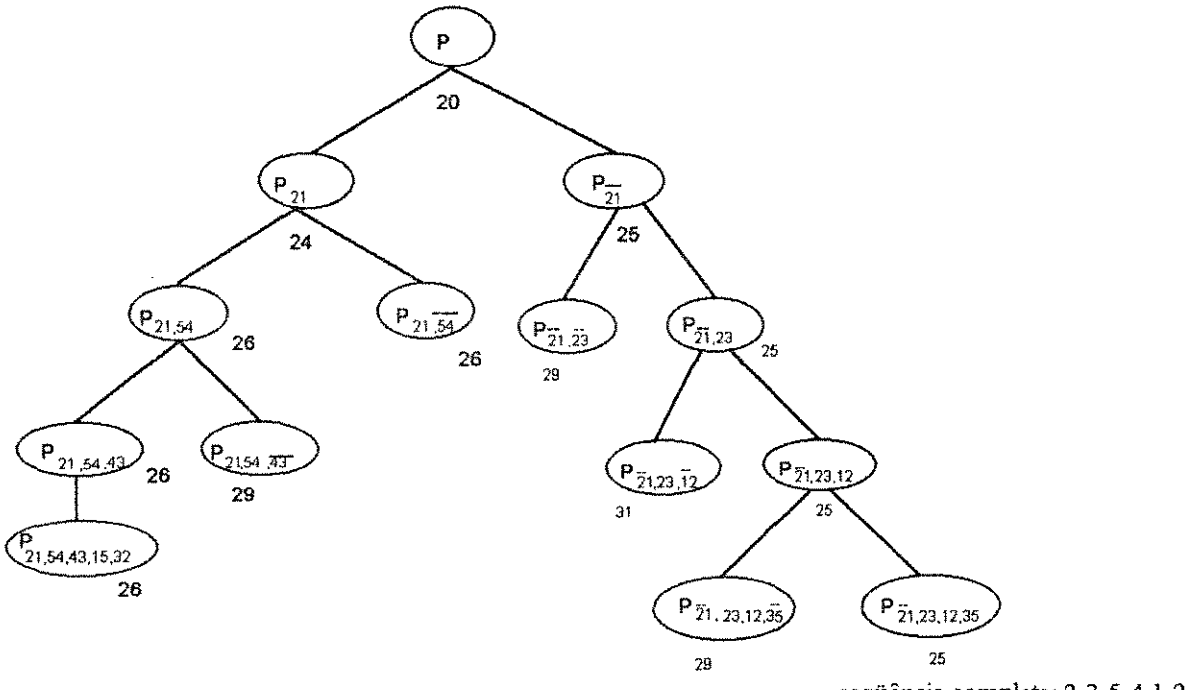


figura 3.6 - Árvore de busca completa da melhor seqüência

A figura 3.5 mostra a primeira solução do problema encontrada. O processo continua até que uma seqüência completa seja encontrada, como mostra a figura 3.6. Em seguida, o incremento de retro-rastreamento("backtraking") é colocado em prática, onde os nós com limitante inferior acima do limitante superior são desprezados (processo "fathomed").

B - Programação Dinâmica

B.1 - Descrição do Método

A Programação Dinâmica também pode ser adaptada ao problema de Seqüenciamento com tempos de estabelecimento, na tentativa de minimização do tempo total de estabelecimento. No exemplo de aplicação que segue é notório a limitação deste método quando estão envolvidas várias tarefas, isto devido ao grande histórico de resultados necessário para a construção de uma seqüência ótima. Na aplicação deste método considere:

a) O ínicio do processamento em i_0;
b) S como um conjunto das tarefas a serem seqüenciadas;
c) K como o conjunto de todas as tarefas menos i_0

A volta ótima, segundo Baker (1974), pode ser vista como sendo $\{i_0\},J,\{i\},S,\{i_0\}$, ou seja, depois do processamento da tarefa i_0 designa-se uma tarefa do conjunto J, em seguida escolhe-se uma tarefa i particular, partindo-se para a designação das tarefas do conjunto S e retornando a tarefa i_0. Os conjuntos J e S não possuem elementos comuns, cosenquentemente o conjunto J possui N - (k+2), se S possuir k tarefas. Tomando-se :

f(i, S) = como sendo o menor tempo de preparação da tarefa i, processando as tarefas do conjunto S e retornando ao processamento de i, então:

$$f(i,S) = \min_{l \in S}[S_{il} + f(l, S - \{l\}] \qquad (3.16)$$

de tal modo que :

$$f(i,\varnothing) = S_{i\, i_0} \tag{3.17}$$

Utilizando-se as espressões 3.16 e 3.17, pode-se chegar a uma seqüência ótima considerando o conjunto S formado por uma, duas até N-1 tarefas, de modo que esteja armazenado em f(i,S) toda a informação de uma seqüência completa.

B.2 - Exemplo do Método

Os dados para aplicação do método estão apresentados na tabela 3.6.

Considerando $i_0 = 1$, tem-se:

estágio 1

$f(2,\varnothing) = 5$
$f(3,\varnothing) = 11$
$f(4,\varnothing) = 5$
$f(5,\varnothing) = 10$

estágio 2

$f(2,\{3\}) = 7 + 11 = 18$
$f(2,\{4\}) = 11 + 5 = 15$
$f(2,\{5\}) = 13 + 10 = 23$

$f(3,\{2\}) = 6 + 5 = 11$
$f(3,\{4\}) = 8 + 5 = 13$
$f(3,\{5\}) = 4 + 10 = 14$

$f(4,\{2\}) = 7 + 5 = 12$
$f(4,\{3\}) = 2 + 11 = 13$
$f(4,\{5\}) = 2 + 10 = 12$

$f(5,\{2\}) = 9+5 = 14$
$f(5,\{3\})=7+11=18$
$f(5,\{4\}) = 5+5 = 10$

estágio 3

$f(2,\{3,4\}) = \min[7+13,11+13] = 20$
$f(2,\{3,5\}) = \min[7+14,13+8] = 21$
$f(2,\{4,5\}) = \min[11+12,13+10] = 23$

$f(3,\{2,4\}) = \min[6+15,8+12] = 20$
$f(3,\{2,5\}) = \min[6+23,4+14] = 18$
$f(3,\{4,5\}) = \min[8+12,4+10] = 14$

$f(4,\{2,3\}) = \min[7+18,2+11] = 13$
$f(4,\{2,5\}) = \min[7+23,2+10] = 12$
$f(4,\{3,5\}) = \min[8+14,2+18] = 16$

$f(5,\{2,3\}) = \min[9+18,7+11] = 18$

$f(5,\{2,4\}) = \min[9+15,5+12] = 17$

$f(5,\{3,4\}) = \min[7+13,5+13] = 18$

estágio 4

$f(2,\{3,4,5\}) = \min[7+14,11+16,13+18] = 21$
$f(3,\{2,4,5\}) = \min[6+23,8+12,4+17] = 20$
$f(4,\{2,3,5\}) = \min[7+21,2+18,2+18] = 20$
$f(5,\{2,3,4\}) = \min[9+20,7+20,5+13] = 18$

estágio 5
$f(1,\{2,3,4,5\}) = \min[4+21,8+20,6+20,8+18] = 25$

A melhor seqüência encontrada via esse método é: seqüência: 1-2-3-5-4-1

C - Métodos Heurísticos

Panwalkar e Iskander (1977) fornecem uma série de procedimentos heurísticos para o seqüenciamento de tarefas. No que se refere a tempos estatelecimento são citados os seguintes procedimentos:

a) Selecionar a tarefa que não necessita de tempo de estabelecimento ;
b) Selecionar a tarefa que necessita de um menor tempo de estabelecimento relativo à tafera processada;
c) Selecionar par de tarefas que possui menor tempo de estabelecimento.

A heurística "Closest Unvisited City", tratada em Baker (1974), é citada como um caminho para a minimização destes tempos. Esta heurística tem como base a designação da tarefa com menor tempo de estabelecimento em relação a tarefa anteriormente processada. Para exemplo deste procedimento considere a tabela 3.6. Iniciando o processamento com a tarefa 1 chega-se a seqüência 1-2-3-5-4-1, com custo de estabelecimento igual a 25.

3.5 - Exemplo de Minimização do tempo total de execução em um só processador

Considere a matriz de tempos de estabelecimento da tabela 3.6 e os tempos de processamento das tarefas na tabela 3.9. O tempo de execução total das tarefas, é calculado pela equação 3.15. Tomando-se a seqüência com tempo total de estabelecimento minimizado, obtida na secção 3.4.2, tem-se:

Tabela 3.7 - Tempos de Processamento

tarefa	1	2	3	4	5
Processador	2	4	4	3	4

seqüência com tempo total de estabelecimento mínimo = 1-2-3-5-4-1

$$te = \sum_{I=1}^{N+1} S_{I-1,I} + \sum_{I=1}^{N} t_I = 25 + 15 = 40 \text{ u.t.}$$

Qualquer seqüência que apresente o tempo total de estabelecimento minimizado, tem tempo total de execução mínimo. Esta conclusão somente é válida quando somente um só processador está envolvido, tendo-se que partir para outros procedimentos quando o número de processadores está acima de 1. A figura 3.7 mostra a Carta de Gantt para o problema acima exemplificado.

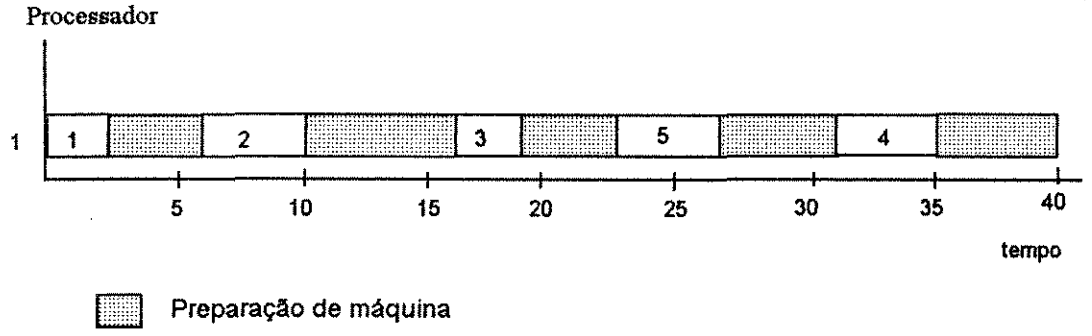


figura 3.7 - Carta de Gantt para o problema

3.6 - Conclusão

Neste Capítulo, o seqüenciamento de tarefas, caso mais simples do escalonamento, é focalizado. A maioria das heurísticas e procedimentos desenvolvidos são direcionados para a solução de problemas de seqüenciamento. Como por exemplo, as heurísticas apresentadas na tabela 3.3 para avaliação do limitante inferior na metodologia "BAB", não incorporam detalhes de processamento , ou seja, só contemplam os tempos de processamento das tarefas, fazendo com que modificações sejam necessárias em suas equações se algum detalhe do processamento não for negligenciável , como por exemplo tempos de transferência das bateladas e de estabelecimento dos processadores. Com isto, é necessário o desenvolvimento de uma função de custo que reflita o processamento em questão.

Quando tempos de estabelecimento são considerados durante o seqüenciamento, um caminho coerente para a otimização do tempo total de execução é a minimização destes tempos improdutivos(tempos mortos). A metodologia "BAB" apresentou-se, novamente, como o método mais promissor para este objetivo.

A semelhança do problema "SDTE" (Seqüenciamento Dependente dos Tempos de Estabelecimento) com o problema do caixeiro viajante fez com que fosse dado ênfase ao estudo da natureza deste último problema e as implicações que surgem quando comparado ao problema "SDTE".

CAPÍTULO 4 - ANÁLISE DO SEQÜENCIAMENTO DE TAREFAS EM UMA LINHA "PURA" DE PROCESSAMENTO

4.1 - Introdução

No presente capítulo são apresentados os algoritmos desenvolvidos neste trabalho, os quais têm o intuito de solucionar alguns problemas linhas de processamento. A metodologia adotada na solução destes problemas é a técnica "Branch and Bound" na versão "Deph First". A abordagem utilizada é semelhante a citada na literatura quando existe um conderável número de tarefas a seqüenciar e quando detalhes da linha de produção são incorporados aos problemas.

Neste capítulo, os seguintes problemas são solucionados:

a) Seqüenciamento sem restrição de qualquer natureza e com tempos de estabelecimento e de transferência negligenciáveis.
b) Seqüenciamento com tempo de estabelecimento envolvendo um único processador.
c) Seqüenciamento com tempos de estabelecimento não-negligenciáveis e independentes do processador.
d) Seqüenciamento de tarefas em dois processadores com tempos de estabelecimento não-negligenciáveis e dependentes da máquina.
e) Seqüenciamento com ordem de manufatura e tempos de estabelecimento e de transferência negligenciáveis.

4.2 - Algoritmo A - Seqüenciamento de tarefas com tempos de estabelecimento independentes da seqüência de produção

4.2.1- Introdução

O seqüenciamento de tarefas neste capítulo será desenvolvido adotando-se como função objetivo o tempo de execução total das tarefas. A heurística "Full Machine Based Bound", apresentada na tabela 3.3, é utilizada para o cálculo do limitante inferior. A busca em profundidade(Pesquisa "Deph First") é adotada no desenvolvimento do Algoritmo.

O algoritmo de seqüenciamento apresentado nesta secção utiliza as fórmulas de recorrências mostradas na secção 3.4.1.2. No desenvolvimento do algoritmo as seguintes simplificações são adotadas:

a) Todas as tarefas estão prontas para o processamento no instante $t = 0$;
b) Um determinado processador nunca está ocioso tendo tarefa a ser executada.

O algoritmo consta das seguintes etapas básicas:

a) Cálculo de uma solução tentativa (ST);
b) Etapa de ramificação - Geração dos subproblemas ou nós filhos
c) Etapa "Bounding"- cálculo do limitante inferior para os subproblemas;
d) Seleção do nó mestre a ser expandido
e) Etapa de retrocesso ("backtraking")

4.2.2 - Descrição das Etapas do Algoritmo A

a) Cálculo da solução tentativa: Para o cálculo da solução inicial é utilizada uma seqüência de permutação, formada pelos N inteiros representantes das tarefas. De posse desta seqüência foi calculado o seu tempo de execução, constituindo assim a solução tentativa para o problema. original.
b) Etapa de Ramificação e cálculo do limitante inferior: Nesta fase, os nós da árvore de busca são gerados e cada nó é rotulado com seu limitante inferior.
c) Seleção do nó mestre a ser expandido: Em cada nível é tomado um nó mestre que apresenta o menor valor do limitante inferior e que irá gerar nós filhos.
d) Etapa de retrocesso ou "backtraking": Uma vez alcançada a base da árvore (Neste ponto é construído o limitante superior do problema) o procedimento de retrocesso é colocado em prática para sondagem e eliminação de nós com custo maior que a solução corrente.

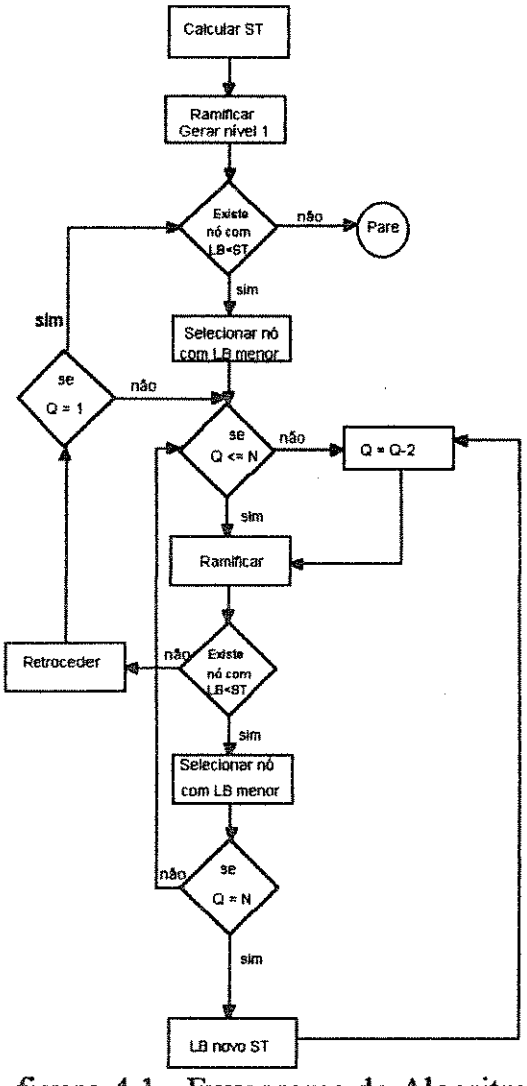


figura 4.1 - Fuxograma do Algoritmo A

A figura 4.1 mostra o fluxograma do Algoritmo A. para o seqüenciamento de tarefas.

4.2.3 - Exemplo de Aplicação

Considere uma linha de Processamento com os dados apresentados na tabela 4.1. Seqüenciar as tarefas de modo a otimizar o tempo de execução total de processamento.

Tabela 4.1 - Tempos de Processamento

Tarefas	Processadores	
	1	2
1	4	5
2	2	3
3	6	3
4	1	4

Neste problema, a tabela 4.2 mostra os nós sondados pelo algoritmo A. No problema exemplo somente uma seqüência completa foi gerada, comprovando, deste modo, a importância da metodologia "BAB".

A seqüência ótima encontrada via Algoritmo A faz parte do subgrupo 4 da tabela 4.3.

Definição de Subgrupo de tarefas: Conjunto de seqüências de tarefas, as quais possuem, por designação, a mesma tarefa na primeira posição .

Esta definição é utlizada na construção dos algoritmos propostos neste trabalho.

Tabela 4.2 - Nós sondados pelo Algoritmo A

Nós sondados	Limitante inferior
A	19
B	17
C	21
D	16
DA	16
DB	16
DC	18
DAB	16
DAC	18
DABC(*)	16

sequência ótima: D-A-B-C ou 4-1-2-3-

Para o problema linha de processamento acima descrito se um algoritmo de enumeração fosse utilizado, N! seqüências teriam de ser analisadas. A tabela 4.3 mostra todas as seqüências possíveis para o problema com os seus tempos de execução. Estas seqüências estão separadas por subgrupos de tarefas.

Tabela 4.3 - Enumeração completa para o problema exemplo

Seqüência	tempo de execução toal (te)
subgrupo 1	
1-2-3-4	19.0
1-2-4-3	19.0
1-3-2-4	20.0
1-3-4-2	20.0
1-4-2-3	19.0
1-4-3-2	19.0
subgrupo 2	
2-1-3-4	19.0
2-1-4-3	18.0
2-3-1-4	21.0
2-3-4-1	20.0
2-4-1-3	17.0
2-4-3-1	18.0
subgrupo 3	
3-1-2-4	22.0
3-1-4-2	22.0
3-2-1-4	21.0
3-2-4-1	21.0
3-4-1-2	21.0
3-4-2-1	21.0
subgrupo 4	
4-1-2-3	16.0
4-1-3-2	17.0
4-2-1-3	16.0
4-2-3-1	18.0
4-3-1-2	19.0
4-3-2-1	18.0

A figura 4.2 mostra a verificação da seqüência ótima através da árvore de busca. O valor que rotula cada nó é o seu limitante inferior.

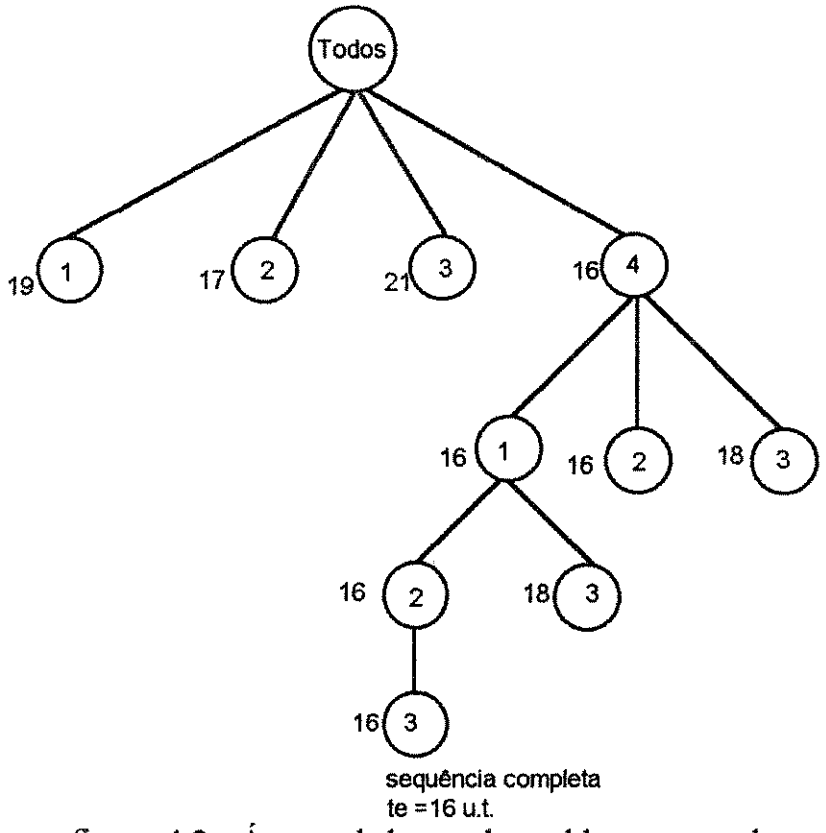


figura 4.2 - Árvore de busca do problema exemplo

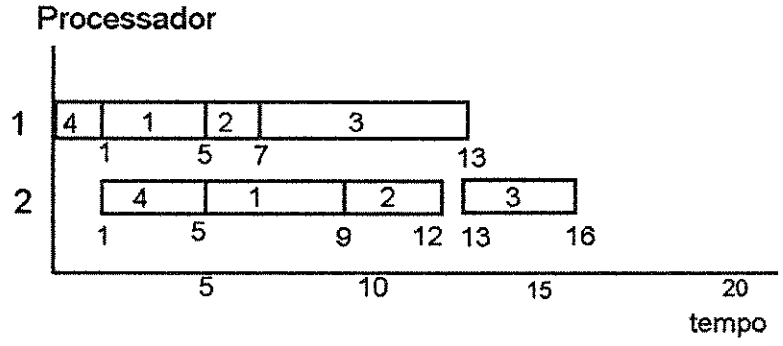


figura 4.3 - Carta de Gantt da seqüência ótima

4.3 - Algoritmo B modificado - Minimização do tempo total de estabelecimento via Metodologia "BAB"

4.3.1 - Introdução

Dentre os métodos discutidos na secção 3.4.2.2, a metodologia de busca em árvore é aplicada. Algumas modificações do procedimento proposto por Little et. al. (1963) foram feitas, de modo a se obter a uma flexibilidade de aplicação futura no seqüenciamento de tarefas com tempos de estabelecimento não-negligenciáveis. O

Algoritmo B é desenvolvido, basicamente, em cima das etapas do método "BAB" (Algoritmo B) , apresentado na secção 3.4.2.2.

4.3.2 - Estratégias utilizadas no desenvolvimento do Algoritmo B modificado

a) O Algoritmo B modificado analisa em um nível corrente da árvore todos os pares de tarefas candidatos, pela regra de seleção (mesmo valor de λ), a minimizarem o tempo total de estabelecimento.

b) Os subproblemas complementares, gerados pela não seleção de um par de tarefas promissor a minimizar o tempo total de estabelecimento, não são analisados.

A escolha arbitrária, quando existe mais de um par promissor à minimização do tempo total de estabelecimento, é evitada pela ramificação ou análise de todos os pares promissores. Na redução 2 do (problema solucionado na secção 3.4.2.2), existem vários pares com $\lambda = 2$, igualmente factíveis de serem expandidos (pares 1-4,3-2,4-3,5-4). No Algortimo B modificado todos estes pares são analisados e árvore resultante no estágio (nível corrente) é mostrada na figura 4.4

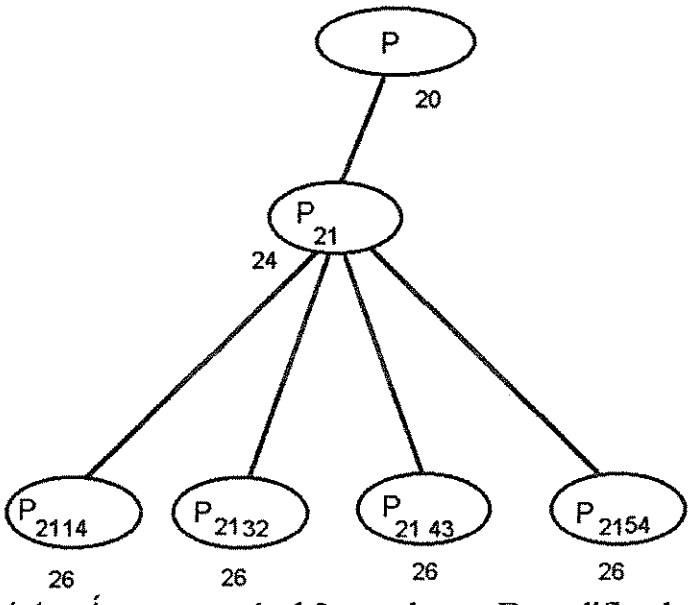


figura 4.4 - Árvore no nível 2 gerada por B modificado

A figura 4.5a e 45b mostra o fluxograma do Algoritmo B modificado.

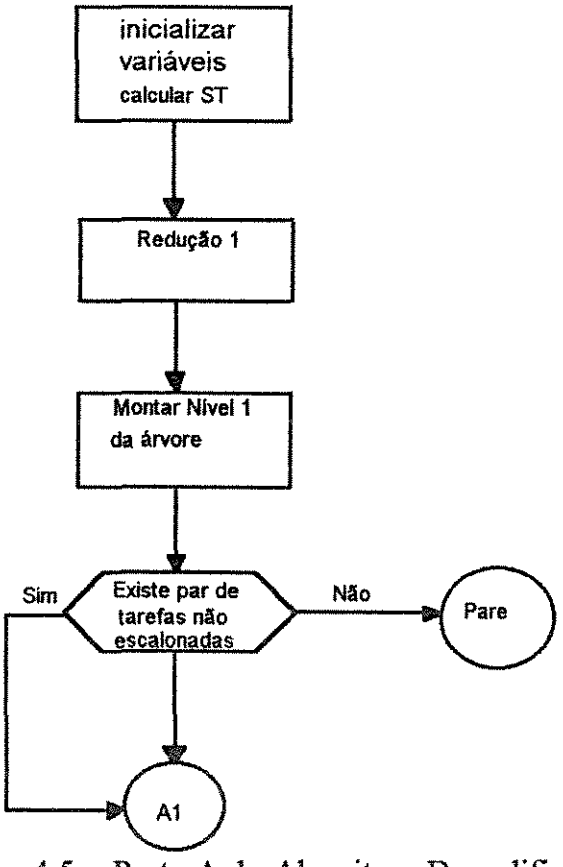


figura 4.5a- Parte A do Algoritmo B modificado

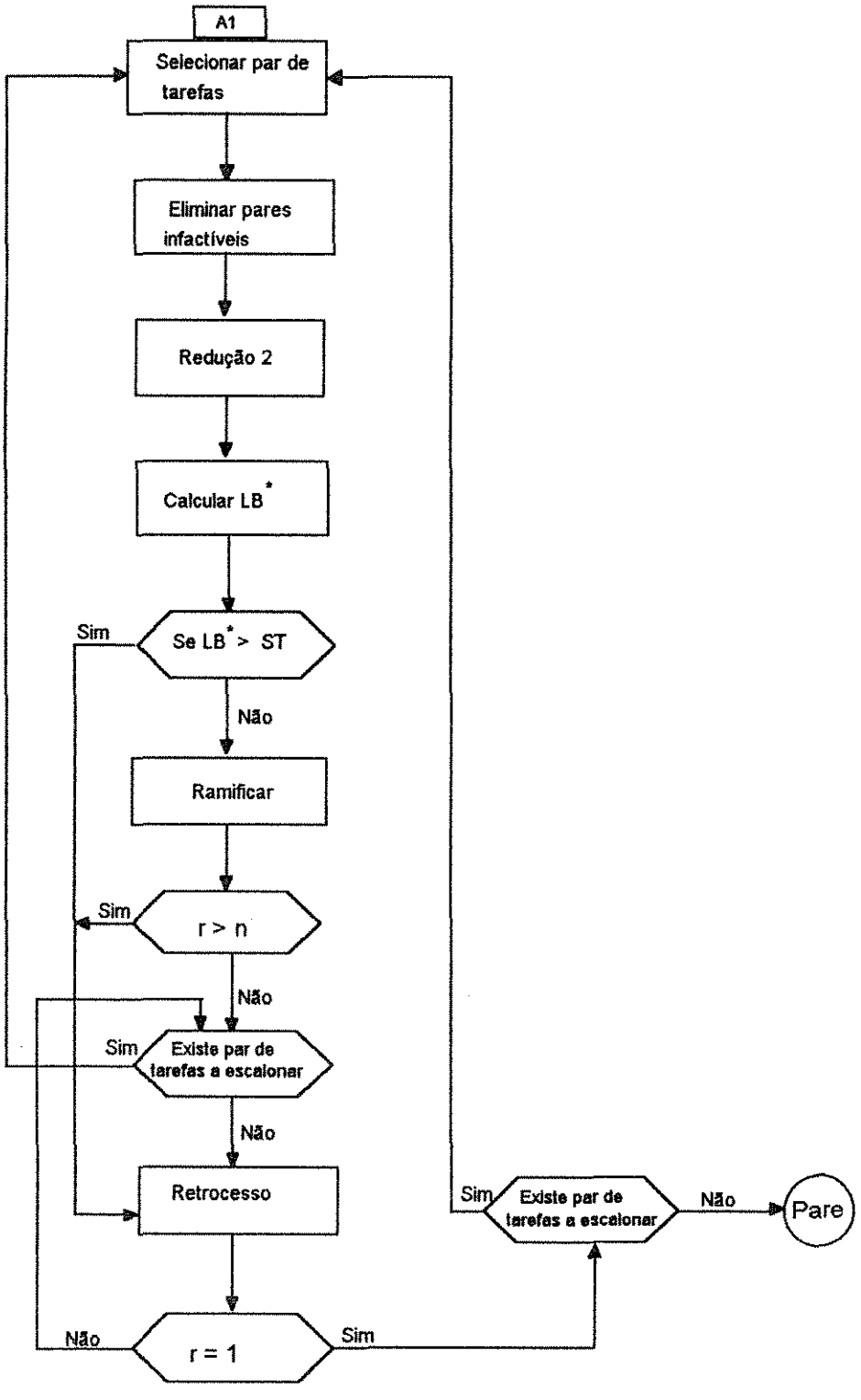


figura 4.5b - Parte A1 do Algoritmo B modificado

4.3.3 -Exemplo 1 de Aplicação do Algoritmo B modificado

Dada a tabela 4.4 representando tempos de estabelecimento para um único processador, o Algoritmo B modificado é aplicado com o objetivo de diminuir os tempos mortos ou tempos improdutivos. Tempos de estabelecimento podem ser vistos como custos de processamento (custo direto: consumo de água para limpeza de máquina; assim custo indireto: tempo que máquina passa sem produzir).

Tabela 4.4 - Tempos de Estabelecimento

Tarefas	Tarefas			
	1	2	3	4
1	-	6	7	3
2	5	-	1	4
3	2	7	-	2
4	8	4	3	-

A tabela 4.5a e 4.5b mostram todas as seqüências possíveis de serem executadas, a cada roteiro está associado o seu tempo total de estabelecimento.

Tabela 4.5a- Enumeração completa para o problema original(subgrupo 1 e 2)

Seqüência	Tempo total de estabelecimento
subgrupo 1	
1-2-3-4-1	17.0
1-2-4-3-1	15.0
1-3-2-4-1	26.0
1-3-4-3-1	18.0
1-4-2-3 -1	10.0
1-4-3-2-1	18.0
subgrupo 2	
2-1-3-4-2	18.0
2-1-4-3-2	18.0
2-3-1-4 -2	10.0
2-3-4-1-2	17.0
2-4-1-3-2	26.0
2-4-3-1-2	15.0

Tabela 4.5b- Enumeração completa para o problema para o problema original(subgrupo 3 e 4)

Seqüência	Tempo total de estabelecimento
subgrupo 3	
3-1-2-4-3	15.0
3-1-4-2-3	10.0
3-2-1-4-3	18.0
3-2-4-1-3	26.0
3-4-1-2-3	17.0
3-4-2-1-3	18.0
subgrupo 4	
4-1-2-3-4	17.0
4-1-3-2-4	26.0
4-2-1-3-4	18.0
4-2-3-1 -4	10.0
4-3-1-2-4	15.0
4-3-2-1-4	18.0

A tabela 4.6 mostra as seqüências analisadas pelo Algoritmo B modificado

Tabela 4.6 - Seqüências Analisadas por B modificado

Seqüências analisadas	Tempo total de estabelecimento
3-1-4-2-	10

Esta seqüência faz para do subgrupo de tarefas 3 da tabela 4.6.

4.3.4 - Exemplo 2 de Aplicação do Algoritmo B modificado

Considere uma linha de processamento com tempos de estabelecimento dados pela 4.7, a tabela 4.8 mostra as seqüências completas analisadas.

Tabela 4.7 - tempos de estabelecimento

Tarefas	Tarefas			
	1	2	3	4
1	-	3	12	8
2	5	-	4	2
3	4	9	-	5
4	2	7	5	-

Tabela 4.8- Enumeração completa para o problema original

Seqüência	Tempo total de estabelecimento
subgrupo 1	
1-2-3-4-1	14.0
1-2-4-3-1	14.0
1-3-2-4-1	25.0
1-3-4-3-1	29.0
1-4-2-3-1	23.0
1-4-3-2-1	27.0
subgrupo 2	
2-1-3-4-2	29.0
2-1-4-3-2	27.0
2-3-1-4-2	23.0
2-3-4-1-2	14.0
2-4-1-3-2	25.0
2-4-3-1-2	14.0
subgrupo 3	
3-1-2-4-3	14.0
3-1-4-2-3	23.0
3-2-1-4-3	27.0
3-2-4-1-3	25.0
3-4-1-2-3	14.0
3-4-2-1-3	29.0
subgrupo 4	
4-1-2-3-4	14.0
4-1-3-2-4	25.0
4-2-1-3-4	29.0
4-2-3-1-4	23.0
4-3-1-2-4	14.0
4-3-2-1-4	27.0

Algumas seqüências geradas por B modificado são mostradas na tabela 4.9. Nota-se que houve repetição de seqüências analisadas , isto devido à análise de todos os pares promissores à minimização.

Tabela 4.9 - Seqüências Analisadas por B modificado

Seqüências analisadas	Custo associado
1-2-3-4-1-	14.0
1-2-4-3-1-	14.0
1-2-4-3-1-	14.0
1-2-3-4-1-	14.0

4.4 - Algoritmo C - Seqüenciamento de tarefas com tempos de estabelecimento dependentes da seqüência de produção e independentes do processador

4.4.1 - Alterações em B modificado (Algoritmo C)

A estratégia de construção do Algoritmo C é baseada em B modificado. A idéia básica é aproveitar as seqüências geradas por B modificado, as quais possuem tempo total de estabelecimento ótimo ou sub-ótimo. Para cada seqüência gerada completamente é calculado o tempo de execução total e por comparação direta com o melhor valor corrente é escolhida a melhor seqüência.

Para atender a estratégia do desenvolvimento do algoritmo C, primeiramente foi realizado um estudo em B modificado. O resultado deste estudo foi que a análise de todos os pares factíveis no nível corrente pode gerar seqüências sub-ótimas em tempo de estabelecimento. Para tanto, não é realizado a etapa de retro-rastreamento ("backtraking") quando limitante inferior de um nó for maior que solução corrente. Testes realizados com esta alteração mostraram que seqüências com custos muito próximos são geradas, e que a priori são candidatas à minimização do tempo de execução. Este resultado é evidenciado quando os tempos de estabelecimento envolvidos são pequenos, como mostrado na tabela 4.10.

Tabela 4.10 - tempos de estabelecimentos

Tarefas	Tarefas						
	1	2	3	4	5	6	7
1	-	.1	.2	.4	.1	.4	.2
2	.2	-	.2	.3	.1	.1	.1
3	.1	.2	-	.1	.2	.1	.1
4	.1	.1	.1	-	.1	.1	.4
5	.3	.3	.4	.3	-	.1	.1
6	.1	.3	.4	.1	.2	-	.1
7	.2	.3	.3	.1	.1	.2	-

Para a tabela 4.10 algumas seqüências completas analisadas pelo Algoritmo C modificado são mostradas na tabela 4.11. Os valores do tempo total de estabelecimento mostram que a análise de todos os pares candidatos aumenta a possibilidade de obtenção de uma seqüência minimizada em tempo total de estabelecimento.

Tabela 4.11 - Seqüências analisadas pelo Algoritmo B modificado

Sequência	Tempo total de estabelecimento
4-3-1-2-5-6-7-4	0.7
4-3-6-1-2-5-7-4	0.7
4-3-1-2-5-6-7-4	0.8
4-3-1-2-5-7-6-4	0.8
4-3-1-2-6-5-7-4	0.8

4.4.2 - Exemplo de Aplicação (Algortimo C)

Dada uma linha de processamento com tempos de processamento representados pela tabela 4.1 e tempos de estabelecimento apresentados na tabela 4.4, otimizar o tempo total de execução.

A tabela 4.12a e 4.12b mostram todas as seqüências possíveis para o problema, assim como o tempo de execução total e o tempo total de estabelecimento associado a cada roteiro de produção.

Tabela 4.12a - Enumeração completa para o problema para o problema original (subgrupo: 1 e 2)

Seqüência	Tempo total de estabelecimento	te (u.t.)
subgrupo 1		
1-2-3-4-1	17.0	36
1-2-4-3-1	15.0	34
1-3-2-4-1	26.0	46
1-3-4-3-1	18.0	38
1-4-2-3-1	10.0	29
1-4-3-2-1	18.0	37
subgrupo 2		
2-1-3-4-2	18.0	37
2-1-4-3-2	18.0	36
2-3-1-4-2	10.0	31
2-3-4-1-2	17.0	37
2-4-1-3-2	26.0	43
2-4-3-1-2	15.0	33

Tabela 4.12b - Enumeração completa para o problema para o problema original (subgrupo: 3 e 4)

Seqüência	Tempo total de estabelecimento	te (u.t.)
subgrupo 3		
3-1-2-4-3	15.0	37
3-1-4-2-3	10.0	32
3-2-1-4-3	18.0	39
3-2-4-1-3	26.0	47
3-4-1-2-3	17.0	38
3-4-2-1-3	18.0	39
subgrupo 4		
4-1-2-3-4	17.0	33
4-1-3-2-4	26.0	43
4-2-1-3-4	18.0	34
4-2-3-1-4	10.0	28
4-3-1-2-4	15.0	34
4-3-2-1-4	18.0	36

A busca da seqüência ótima via Algoritmo C para o problema gerou as seqüências apresentadas na tabela 4.13a e 4.13b. Nota-se que o Algoritmo tendeu à enumeração completa para este problema.

Tabela 4.13a - Seqüências Analisadas do subgrupos 1 e 2 pelo Algoritmo C

Seqüência	Tempo total de estabelecimento	te (u.t.)
subgrupo 1		
1-2-4-3-1	15.0	36
1-2-3-4-1	17.0	34
1-3-4-2-1	18.0	46
1-3-2-4-1	26.0	38
1-4-3-2-1	18.0	29
1-4-2-3-1	10.0	37
subgrupo 2		
2-1-4-3-2	18.0	37
2-1-3-4-2	18.0	36
2-3-4-1-2	17.0	31
2-3-1-4-2	10.0	37
2-4-3-1-2	15.0	43
2-4-1-3-2	26.0	33

Tabela 4.13b - Seqüências Analisadas: subgrupos 3 e 4 pelo Algoritmo C

Seqüência	Custo total de estabelecimento	te(u.t.)
subgrupo 3		
3-1-4-2-3	10.0	37
3-1-2-4-3	15.0	32
3-2-4-1-3	26.0	39
3-2-1-4-3	18.0	47
3-4-2-1-3	18.0	38
subgrupo 4		
4-1-3-2-4	17.0	33
4-1-2-3-4	26.0	43
4-2-3-1-4	18.0	34
4-2-2-1-4	10.0	28
4-3-2-1-4	15.0	34
4-3-1-2-4	18.0	36

4.5 - Algoritmo D - Seqüenciamento de tarefas com tempos de estabelecimento dependentes da seqüência de produção e dependentes do processador.

4.5.1 - Introdução

O algoritmo D procura resolver este problema combinando,basicamente, o limitante inferior dado pela heurística "Full Machine Based Bound" e o limitante inferior obtido pelo processo de redução, ou seja, é necessário que o cálculo do limitante inferior das seqüências parciais leve em conta a presença dos tempos de estabelecimento.

4.5.2 - Fórmulas de Recorrência para o cálculo do instante de término de uma tarefa k no processador j e do limitante inferior.

A heurística "Full machine Based Bound" utilizada para o cálculo do limitante inferior, no problema de seqüenciamento, deve ser modificada para atender à característica do processo. Para problemas de seqüenciamento dependente da ordem de execução das tarefas, o tempo de término de uma seqüência parcial, qk, em um processador j é dado por:

$$C_{(qk,j)} = \max\{C_{(qk,j-1)}, C_{(q,j)} + s^{j}_{k-1,k}\} + t_{kj} \qquad (4.1)$$

$$C_{(qk,0)} = C_{(0,j)} = 0 \qquad (4.2)$$

onde

k = nova tarefa a entrar na linha de processamento;
q = seqüência parcial já desenvolvida;
$sj_{k-1,k}$= tempo necessário para estabelecer o processador j para a nova tarefa k depois que a tarefa anteriormente processada (k-1) termina o seu processamento;

t_{kj}= tempo de processamento da tarefa k no processador j

Considerando uma seqüência parcial qk o tempo mínimo para que se desenvolva as atividades restantes no processador j é dado por:

$$b_j^* = C_{(q\,k,j)} + (LB^* - sc^j_{q\,k}) + R_j \qquad 1 \le j \le M \tag{4.3}$$

$$R_j = \sum_{l \in U} t_{lj} \tag{4.4}$$

onde:

LB^* = limitante inferior em tempo de estabelecimento para a seqüência qk;

$sc^j_{q\,k}$= tempo estabelecimento já comprometido pela seqüência parcial qk;

R_j = carga de trabalho remanescente no processador j;
U = conjunto de tarefas não seqüenciadas

O limitante inferior para a seqüência qk é dado por :

$$LB^{**}_{qk} = \max\{bj^*\} \tag{4.5}$$

O limitante inferior representante da seqüência parcial q é:

$$LB^{**}_q = \min\{LB^{**}_{qk}\} \tag{4.6}$$

4.5.3 - Estratégias para desenvolvimento dos Algoritmos de minimização do tempo total de execução

Em nossa proposta dois problemas de seqüenciamento são abordados:

a) Caso 1- Minimização do tempo de execução total com tempos estabelecimento não negligenciáveis e independentes do processador;
b) Caso 2- Minimização do tempo de execução total com tempos de estabelecimento não negligenciáveis e dependentes do processador.

A busca do roteiro ótimo dá-se por análise de subgrupo de tarefas. Para cada subgrupo são geradas árvores de busca. A figura 4.6 e 4.7 mostram a formação de dois subgrupos: 1 e 2 respectivamente. A primeira posição da seqüência é fixada com a tarefa N = 1,2,3,...N representante de cada subgrupo.

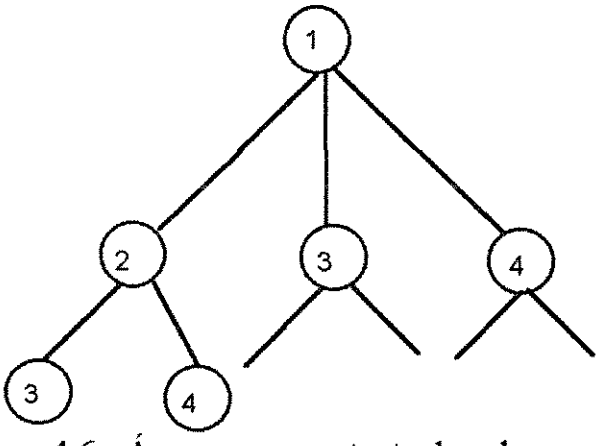


figura 4.6 - Árvore representante do subgrupo 1

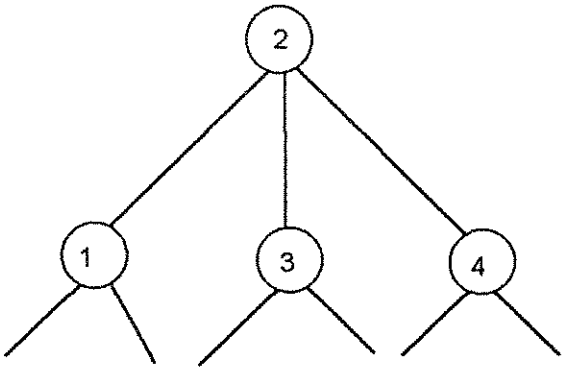


figura 4.7 - Árvore representante do subgrupo 2

A primeira solução encontrada no subrupo 1 é utilizada como limitante superior nos outros subgrupos.

4.5.4 - Aplicação do Algoritmo D - Caso 1(tempos de estabelecimentos independentes do processador)

Os dados para este exemplo são apresentados na tabela 4.14. Neste problema, a busca via metodologia "BAB" é aplicada utilizando as fórmulas propostas para o cálculo do limitante inferior na secção 4.5.2.

Tabela 4.14 - Dados da Linha de Processamento "Pura"

	Processadores		tempos de estabelecimento Processador 1 tarefas				tempos de estabelecimento Processador 2 tarefas			
Tarefas	1	2	1	2	3	4	1	2	3	4
1	4	5	1	-	6	7	3	1	-	6	7	3
2	2	3	2	5	-	1	4	2	5	-	1	4
3	6	3	3	2	7	-	2	3	2	7	-	2
4	1	4	4	8	4	3	-	4	8	4	3	-

As tabelas de 4.15 a 4.18 mostram o cálculo do limitante inferior dos subproblemas de cada subgrupo.Nestas tabelas são mostrados o tempo de execução parcial em cada processador e o limitante inferior das seqüências geradas.

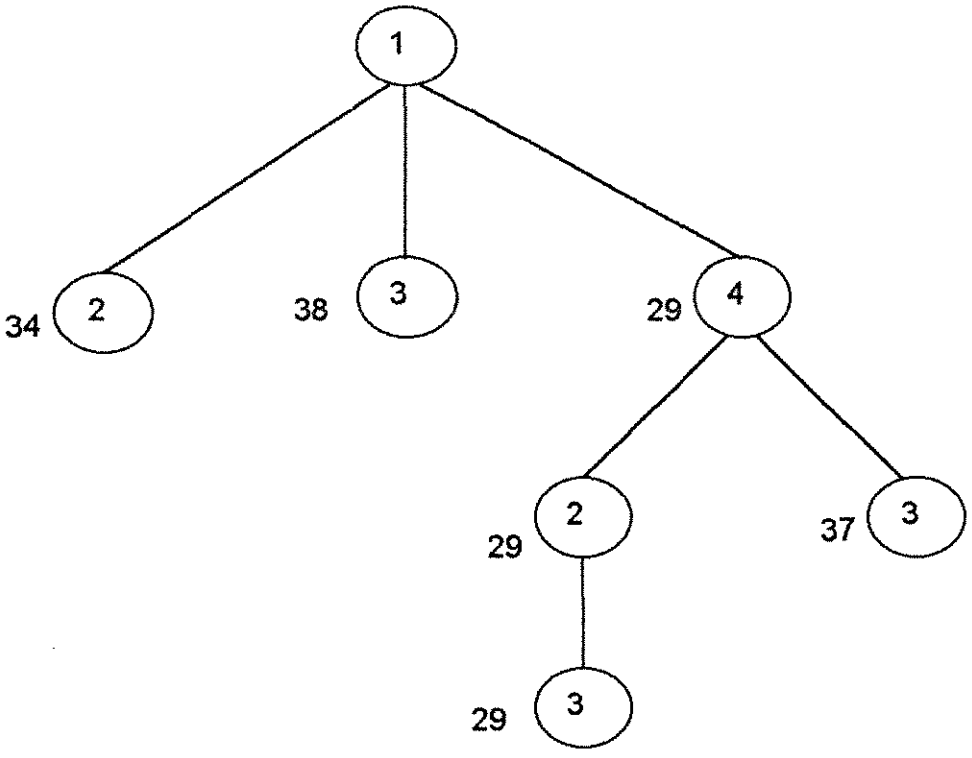
As Árvores de busca para o problema são mostradas nas figuras 4.8 a 4.11.

Tabela 4.15 - Avaliação do limitante inferior dos subproblemas do subgrupo 1

seqüência parcial q	(C1,C2) seqüência q	seqüência parcial qk	(C1,C2) seqüência qk	(b_1^*, b^*_2)	LB_{qk}^{**}	LB_q^{**}
1-2-	(12,18)	1-2-3-	(19,22)	(30,36)	36	
1-2-	(12,18)	1-2-4-	(17,26)	(28,34)	34	34
1-3-	(17,20)	1-3-2-	(26,30)	(39,46)	46	
1-3-	(17,20)	1-3-4-	(20,26)	(31,38)	38	38
1-4-	(8,16)	1-4-2-	(14,23)	(23,29)	29	29
1-4-	(8.16)	1-4-3-	(17,22)	(31,37)	37	
1-4-2-	(14,23)	1-4-2-3-	(21,27)	(23,29)	29	29
1-4-2-3-1*					29	

(*) seqüência completa: 1-4-2-3-1
te = 29 u.t.

O valor do tempo de execução da seqüência 1-4-2-3-1 constitui o limitante superior para o problema.



seqüência completa: 1-4-2-3-1

te = 29 u.t.

figura 4.8 - Árvore de Busca no subgupo 1

Tabela 4.16 - Avaliação do limitante inferior dos subproblemas do subgrupo 2

seqüência parcial q	(C1,C2) seqüência q	seqüência parcial qk	(C1,C2) seqüência qk	(b_1^*, b^*_2)	LB_{qk}^{**}	LB_q^{**}
2-1-	(11,16)	2-1-3-	(24,27)	(31,37)	37	
2-1-	(11,16)	2-1-4-	(15,23)	(31,36)	36	36
2-3-	(9,12)	2-3-1-	(15,20)	(23,31)	31	
2-3-	(9,12)	2-3-4-	(12,18)	(30,37)	37	31
2-4-	(7,13)	2-4-1-	(19,26)	(39,43)	43	32
2-4-	(7.13)	2-4-3-	(16,19)	(28,32)	32	

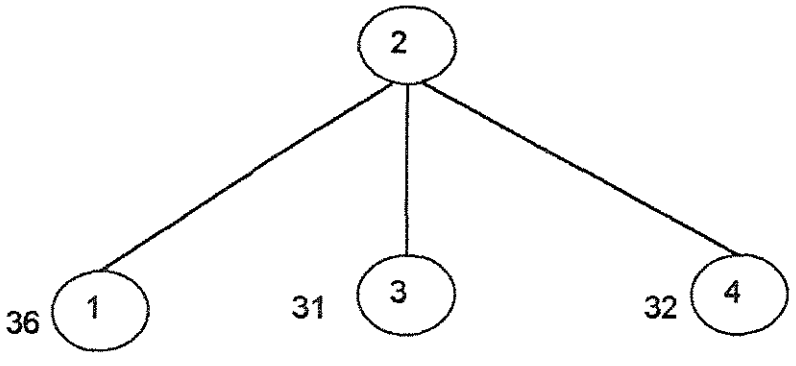


figura 4.9 - Árvore de Busca no subgupo 2

Nota-se que nenhuma seqüência completa do subgrupo 2 foi gerada, pois nenhum limitante inferior dos subproblemas tabela 4.16 está abaixo de 29. Este mesmo resultado pode ser visto na figura 4.10, todos os subproblemas do subgrupo 3 do primeiro nível apresentam limitante inferior acima de 29.

Tabela 4.17 - Avaliação do limitante inferior dos subproblemas do subgrupo 3

seqüência parcial q	(C1,C2) seqüência q	seqüência parcial qk	(C1,C2) seqüência qk	(b_1^*,b^*_2)	LB_{qk}^{**}	LB_q^{**}
3-1-	(12,17)	3-1-2-	(20,26)	(28,37)	37	
3-1-	(12,17)	3-1-4-	(16,24)	(23,32)	32	32
3-2-	(15,19)	3-2-1-	(24,29)	(31,39)	39	
3-2-	(15,19)	3-2-4-	(13,16)	(39,47)	47	39
3-4-	(8,11)	3-4-1-	(21,28)	(30,38)	38	38
3-4-	(8.11)	3-4-2-	(15,22)	(31,39)	39	

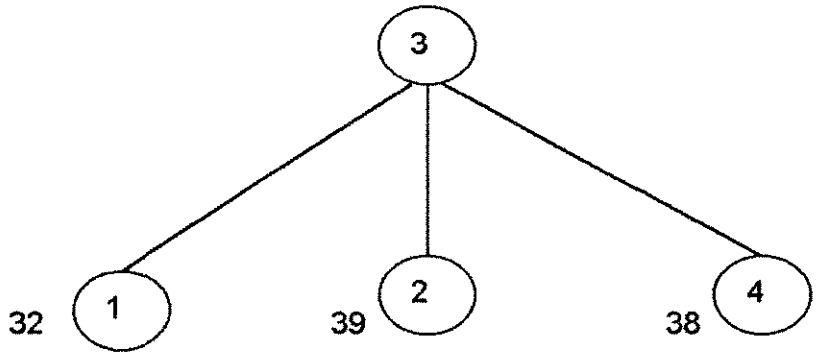


figura 4.10 - Árvore de Busca no subgupo 3

Tabela 4.18 - Avaliação do limitante inferior dos subproblemas do subgrupo 4

seqüência parcial q	(C1,C2) seqüência q	seqüência parcial qk	(C1,C2) seqüência qk	(b_1^*,b^*_2)	LB_{qk}^{**}	LB_q^{**}
4-1-	(13,18)	4-1-2-	(21,27)	(30,33)	33	
4-1-	(13,18)	4-1-3-	(26,29)	(39,43)	43	33
4-2-	(7,12)	4-2-1-	(16,22)	(31,34)	34	
4-2-	(7,12)	4-2-3-	(14,17)	(23,27)	27	27
4-3-	(10,13)	4-3-1-	(16,21)	(28,34)	34	34
4-3-	(10,13)	4-3-2-	(19,23)	(31,36)	36	
4-2-3-	(14,17)	4-2-3-1-	(20,25)	(23,28)	28	28
4-2-3-1-4*	-	-	-	-	28	-

(*) seqüência completa: 4-2-3-1-4

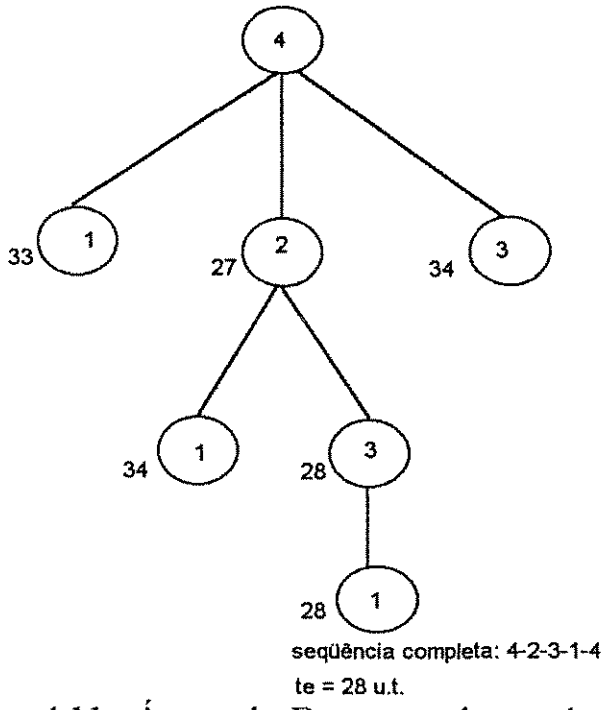


figura 4.11 - Árvore de Busca no subgupo 4

A figura 4.12 mostra a Carta de Gantt da seqüência ótima do problema original.

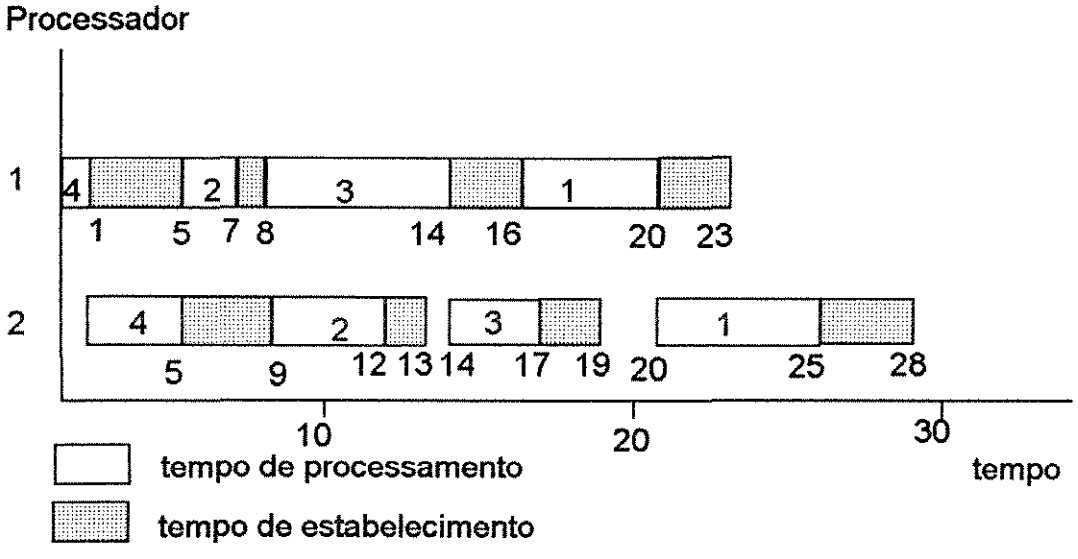


figura 4.12 - Carta de Gantt para o problema exemplo

Por meio do Algoritmo D somente 2 seqüências completas foram analisadas diante das 24 seqüências analisadas pelo Algoritmo C.

4.5.5 - Aplicação do Algoritmo D - Caso 2(tempos de estabelecimento dependentes do processador)

Neste exemplo de aplicação são utilizados os dados da tabela 4.19. A matriz de tempos de estabelecimento do processador é a mesma utilizada nos algortimos anteriores (C e D - Caso1).

Tabela 4.19 - Dados da Linha de Processamento "pura"

Tarefas	Processadores 1	2	tempos de estabelecimento Processador 1 tarefas 1	2	3	4	tempos de estabelecimento Processador 2 tarefas 1	2	3	4
1	4	5	1	-	6	7	3	1	-	3	12	8
2	2	3	2	5	-	1	4	2	5	-	4	2
3	6	3	3	2	7	-	2	3	4	9	-	5
4	1	4	4	8	4	3	-	4	2	7	5	-

Nas tabela 4.20 a 4.23 é mostrado o cálculo do limitante inferior dos subprobleams de cada subgrupo.

Tabela 4.20 - Avaliação do limitante inferior dos subproblemas subgrupo 1

seqüência parcial q	(C1,C2) seqüência q	seqüência parcial qk	(C1,C2) seqüência qk	(b_1^*, b^*_2)	LB''_{qk}	LB_q^{**}
1-2-	(12,15)	1-2-3-	(19,22)	(30,33)	33	
1-2-	(12,15)	1-2-4-	(17,21)	(28,33)	33	33
1-3-	(17,24)	1-3-2-	(26,36)	(39,44)	44	44
1-3-	(17,24)	1-3-4-	(20,33)	(31,48)	48	
1-4-	(8,21)	1-4-2-	(14,31)	(23,42)	42	42
1-4-	(8.21)	1-4-3-	(17,29)	(31,46)	46	
1-2-3-	(19,22)	1-2-3-4	(22,31)	(30,33)	33	33
1-2-4-	(17,21)	1-2-4-3-	(26,29)	(28,33)	33	33
1-2-3-4-1*					33	

(*) seqüência completa: 2-3-4-1-2

te = 33 u.t.

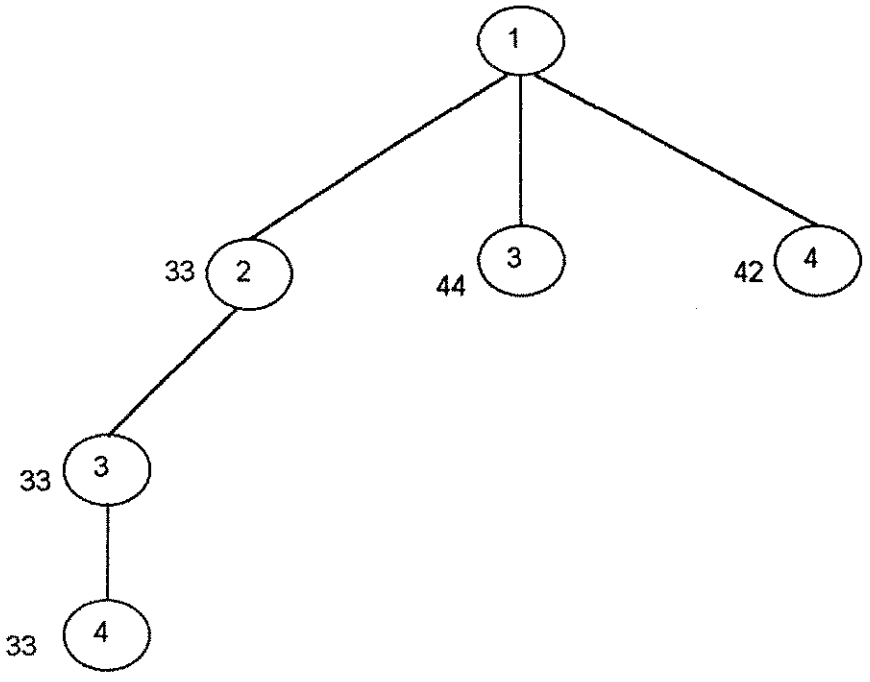


figura 4.13 - Árvore de Busca no subgupo 1

Tabela 4.21 - Avaliação do limitante inferior dos subproblemas do subgrupo 2

seqüência parcial q	(C1,C2) seqüência q	seqüência parcial qk	(C1,C2) seqüência qk	(b_1^*, b^*_2)	LB_{qk}^{**}	LB_q^{**}
2-1-	(11,16)	2-1-3-	(24,31)	(31,47)	47	
2-1-	(11,16)	2-1-4-	(15,28)	(31,45)	45	45
2-3-	(9,12)	2-3-1-	(15,21)	(23,40)	41	
2-3-	(9,12)	2-3-4-	(12,21)	(30,31)	31	31
2-4-	(7,11)	2-4-1-	(19,24)	(39,48)	48	
2-4-	(7.11)	2-4-3-	(16,19)	(28,31)	31	31
2-3-1-	(15,21)	2-3-1-4-	(19,33)	(23,40)	40	40
2-3-4-	(12,21)	2-3-4-1-	(24,29)	(30,32)	32	32
2-3-4-1-2*	-	-	-	-	32	

(*) seqüência completa: 1-2-3-4-1
te = 32 u.t.

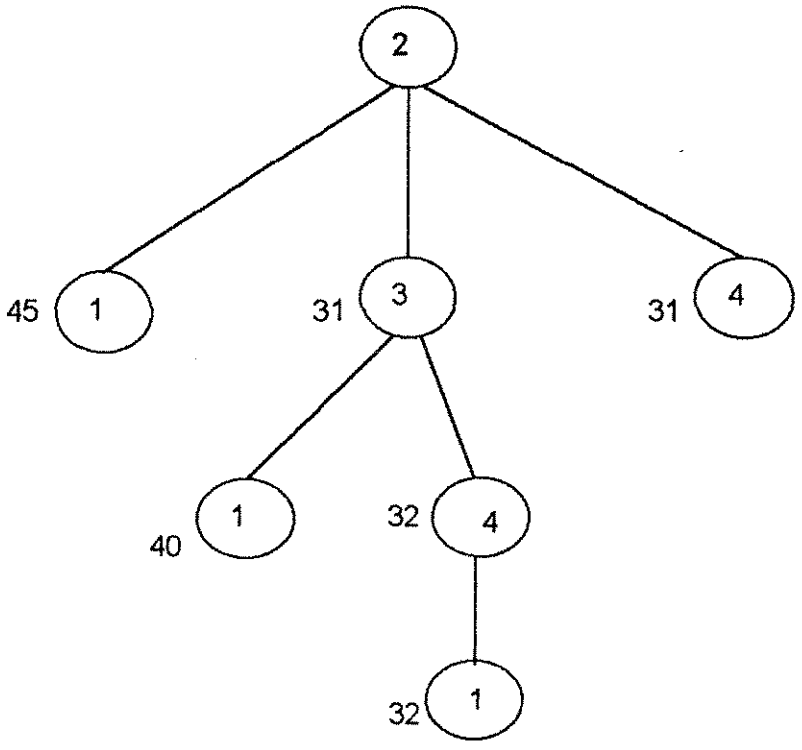


figura 4.14 - Árvore de Busca no subgupo 2

Tabela 4.22 - Avaliação do limitante inferior dos subproblemas do subgrupo 3

seqüenciaparcial q	(C1,C2) seqüência q	seqüência parcial qk	(C1,C2) seqüência qk	$(b1^*,b^*2)$	LB_{qk}^{**}	LB_q^{**}
3-1-	(12,18)	3-1-2-	(20,24)	(28,37)	35	
3-1-	(12,18)	3-1-4-	(16,30)	(23,44)	44	35
3-2-	(15,21)	3-2-1-	(24,31)	(31,48)	48	
3-2-	(15,21	3-2-4-	(20,27)	(39,46)	46	46
3-4-	(9,18)	3-4-1-	(21,26)	(30,36)	36	36
3-4-	(9.18)	3-4-2-	(15,28)	(31,50)	50	

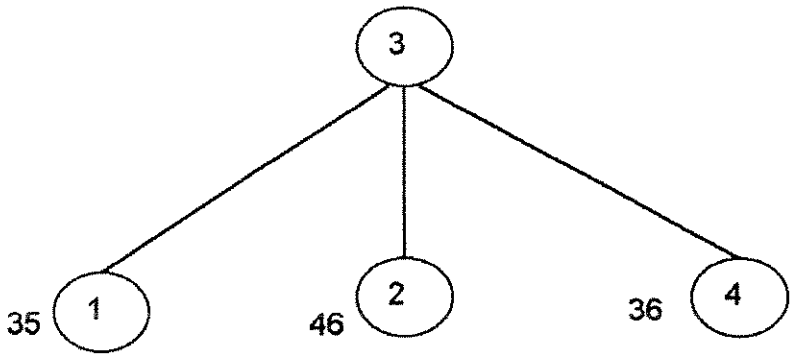


figura 4.15 - Árvore de Busca no subgupo 3

Nenhuma seqüência pertencente ao subgrupo 3 foi completamente analisada.

Tabela 4.23 - Avaliação do limitante inferior dos subproblemas do subgrupo 4

seqüência parcial q	(C1,C2) seqüência q	seqüência parcial qk	(C1,C2) seqüência qk	(b_1^*, b^*_2)	LB_{qk}^{**}	LB_q^{**}
4-1-	(13,18)	4-1-2-	(21,24)	(30,36)	36	
4-1-	(13,18)	4-1-3-	(26,33)	(39,47)	47	36
4-2-	(7,15)	4-2-1-	(16,25)	(31,45)	45	
4-2-	(7,15)	4-2-3-	(14,22)	(23,39)	39	39
4-3-	(10,13)	4-3-1-	(16,22)	(28,30)	30	30
4-3-	(10,13)	4-3-2-	(19,25)	(31,43)	43	
4-3-1-	(16,22)	4-3-1-2-	(24,28)	(28,30)	30	30

(*) seqüência completa: 4-3-1-2-4
te = 30 u.t.

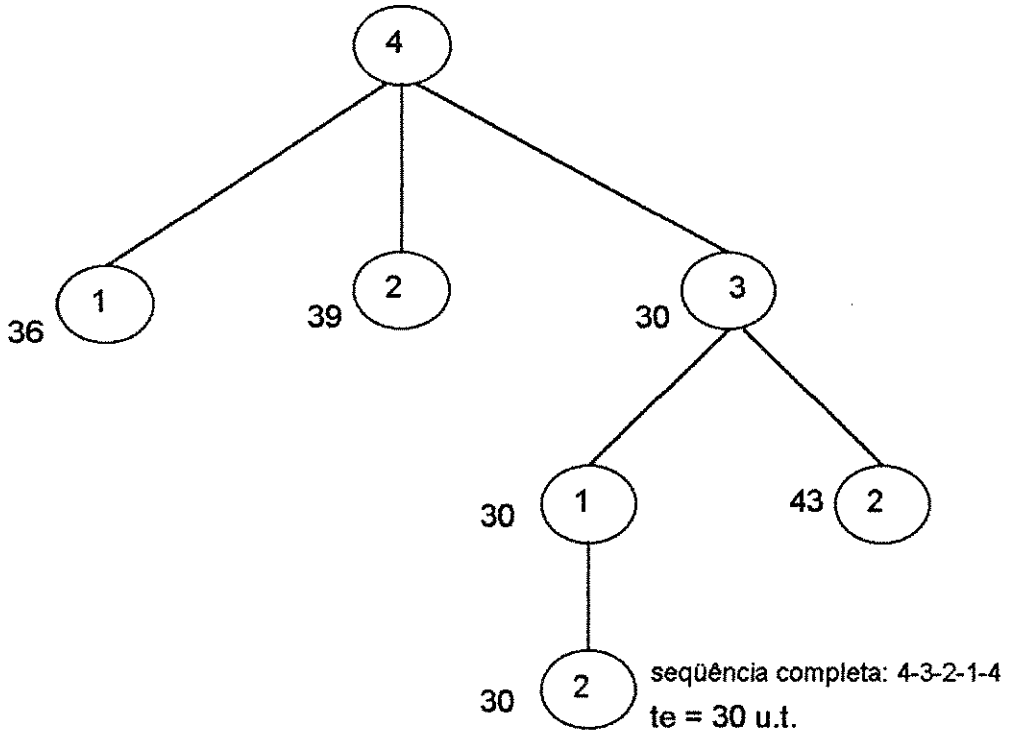


figura 4.16 - Árvore de Busca no subgupo 4

As figuras 4.13 a 4.16 mostram a pesquisa em árvore desenvolvida em cada subgrupo. Nota-se que somente 3 seqüências completas foram geradas. Para efeito de comparação a tabela 2.24a e 4.24b mostram todas as seqüências possíveis para o problema, associando-as a seus tempo de estabelecimento e o tempo de execução.

Tabela 4.24a - Enumeração completa para o problema original (subgrupo: 1 e 2)

Seqüência	Tempo de estabelecimento 1 Processador 1	Tempo de estabelecimento 2 Processador 2	te (u.t.)
subgrupo 1			
1-2-3-4-1	17.0	14.0	33.0
1-2-4-3-1	15.0	14.0	33.0
1-3-2-4-1	26.0	25.0	44.0
1-3-4-3-1	18.0	29.0	48.0
1-4-2-3-1	10.0	23.0	42.0
1-4-3-2-1	18.0	27.0	46.0
subgrupo 2			
2-1-3-4-2	18.0	29.0	47.0
2-1-4-3-2	18.0	27.0	45.0
2-3-1-4-2	10.0	23.0	40.0
2-3-4-1-2	17.0	14.0	32.0
2-4-1-3-2	26.0	25.0	48.0
2-4-3-1-2	15.0	14.0	31.0

Tabela 4.24b - Enumeração completa para o problema original (subgrupo: 3 e 4)

Seqüência	Custo 1 Processador 1	Custo 2 Processador 2	te (u.t.)
subgrupo 3			
3-1-2-4-3	15.0	14.0	35.0
3-1-4-2-3	10.0	23.0	44.0
3-2-1-4-3	18.0	27.0	48.0
3-2-4-1-3	26.0	25.0	49.0
3-4-1-2-3	17.0	14.0	36.0
3-4-2-1-3	18.0	29.0	50.0
subgrupo 4			
4-1-2-3-4	17.0	14.0	36.0
4-1-3-2-4	26.0	25.0	47.0
4-2-1-3-4	18.0	29.0	45.0
4-2-3-1-4	10.0	23.0	39.0
4-3-1-2-4*	15.0	14.0	30.0
4-3-2-1-4	18.0	27.0	43.0

(*) seqüência ótima

A ditribuição temporal das tarefas do melhor roteiro de manufatura é mostrado na figura 4.17.

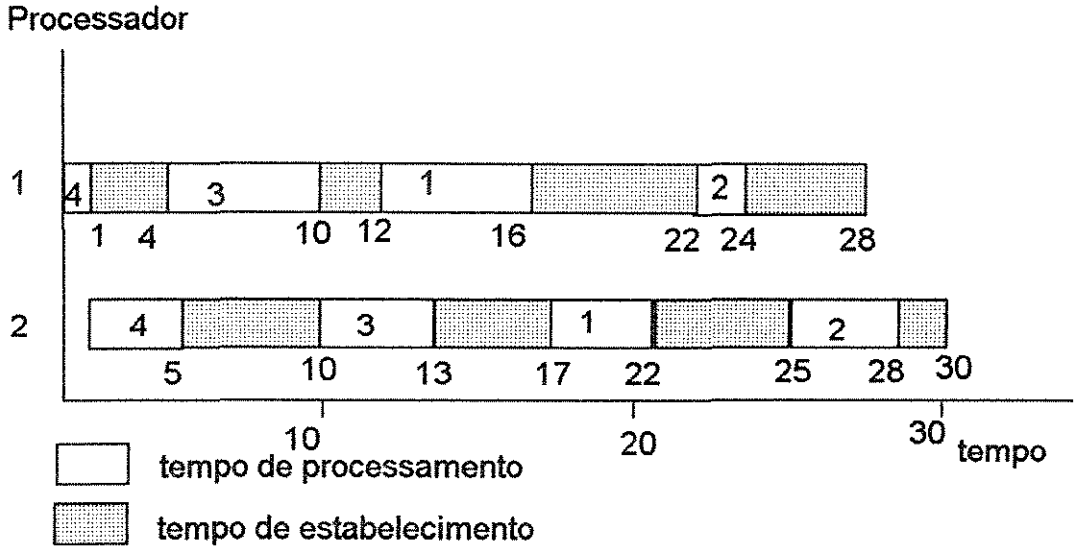


figura 4.17 - Carta de Gantt para o melhor roteiro manufatura

a) O tempo de processamento a ser gasto no processador j, R_j, com a adoção de uma pseudo-tarefa com a característica citada não é afetado;
b) A terceira parcela que compõe o limitante inferior da heurística "Full Machine Bound Based", Uj (ver tabela 3.3), refere-se ao tempo mínimo a ser gasto supondo que o processamento em j já tenha acabado Quando a pseudo-tarefa faz parte do conjunto de tarefas não seqüenciadas, esta pseudo-tarefa é vista como um agregado de tarefas,ou melhor, como uma única tarefa, não afetando assim o cálculo do limitante inferior..

O desenvolvimento do Algoritmo E tem como base estas modificações efetuadas no Algoritmo A., já que a restrição de ordem de manufatura considerada não descaracterizou o problema de seqüenciamento.

4.6.3 - Exemplo de Aplicação

Dada a matriz de tempos de processamento, mostrada na tabela 4.26, de uma linha de processamento, encontrar o roteiro ótimo de manufatura, tendo como ordem prioritária a execução das tarefas 6-1.

Tabela 25 - Tempos de Processamento

Tarefas	Processadores		
	1	2	3
1	3	2	3
2	4	5	3
3	2	2	6
4	4	3	2
5	1	5	1
6	1	4	3

Para este problema a tarefa 7, pseudo-tarefa, tem tempos de processamento dados pelo somatório dos tempos das tarefas presentes na ordem prioritária de tarefas (6 e 1).

O Algoritmo E consta dos principais passos:

A - Construção da matriz pseudo-matriz de tempos processamento

Para o problema esta matriz apresenta as seguintes características:

a) Os tempos de processamento das tarefas pertencentes à ordem de manufatura são nulos;
b) A sétima tarefa (N+1) corresponde a pseudo-tarefa com tempos de processamento sendo a soma dos tempos das tarefas 6 e 1.

A tabela 4.26 mostra a pseudo-matriz para o problema proposto

Tabela 4.26 - Tempos de Processamento

Tarefas	Processadores		
	1	2	3
1	-	-	-
2	4	5	3
3	2	2	6
4	4	3	2
5	1	5	1
6	-	-	-
7	4	6	6

B - Aplicação da Metodologia "BAB" ao Problema

A matriz apresentada na tabela 4.26 é aplicada a metodologia "BAB" tal como no seqüenciameno de tarefas sem restrições, apresentado na secção 4.2.

C - Desmembramento da Pseudo-tarefa

De posse da pseudo-seqüência o roteiro ótimo de manufatura é obtido pelo desmembramento da tarefa 7 (pseudo-tarefa).

pseudo-seqüência: 5-3-7-2-4
seqüência real(roteiro de produção ótimo): 5-3-6-1-2-4
te = 25 u.t.

No exemplo acima, a melhor seqüência encontrada foi: 5-3-7-2-4. A tabela 4.27 mostra as seqüências completas analisadas pelo algoritmo E..

Tabela 4.27 - Seqüências analisadas na pesquisa

Seqüências	te (u.t)
7-2-3-4-5	28
3-7-2-4-5	26
5-3-7-2-4	25

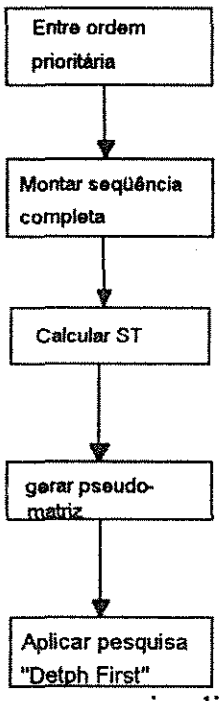


figura 4.18 - Fluxograma simplificado do Algoritmo E

A figura 4.18 mostra o fluxograma do Algoritmo E em sua forma simplificada.

A figura 4.19 mostra a árvore de busaca do roteiro ótimo.

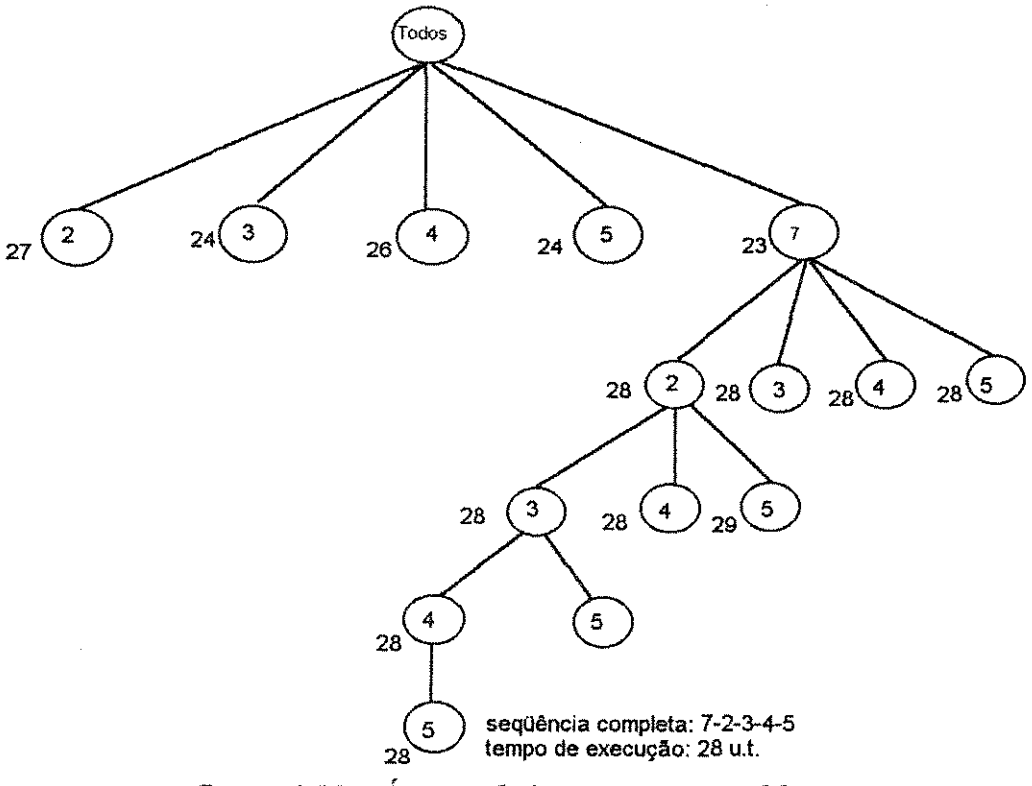


figura 4.19 - Árvore de busca para o problema

4.7- Amostragem Aleatória

No apêndice B é mostrado o gerador de números pseudo-aleatório em Turbo Pascal utilizado no estabelecimento de problemas aleatórios, os quais foram utilizados pelos Algoritmos desenvolvidos. O procedimento pseudo-aleatório foi desenvolvido por George Marsaglia e Zaman (Universidade Estadual Da Flórida, 1987) em Fortran e tem como entrada básica duas sementes: A e B.

Um Algoritmo aleatório pode ser visto como procedimento entre um método heurístico e um método exato (metodologia "BAB") . A limitação no uso de um procedimeto aleatório no tratamento de problemas determinísticos é a falta de um estudo quantitativo de seu comportamento, ou melhor, da garantia de que se dentro da dimensão do número de seqüências a ser analisada existe e qual a probabilidade da seqüência ótima ser gerada. No apêndice C é mostrada a estatística de distribuição em percentagem dos números previamente estabelecidos (números da faixa de 1 a 9) obtidos aleatoriamente. A análise de aleatoriedade do gerador pseudo-randômico mostrou bons resultados de obtenção dos números na faixa desejada, como mostram os gráficos.

CAPÍTULO 5 - CONCLUSÕES

Na presente tese, foram desenvolvidos vários algoritmos direcionados à solução de problemas com tempos de estabelecimento não negligenciáveis e seqüenciamento com ordem de manufatura. Utilizando-se de procedimentos heurísticos e de métodos mais exatos, como a técnica "Branch and Bound", os problemas em uma linha de processamento propostos foram solucionados. Os algoritmos desenvolvidos neste trabalho foram:

A - Algoritmo A: Seqüenciamento de tarefas com tempos de estabelecimento independentes da seqüência de produção.

Este algoritmo mostrou-se viável para a determinação do roteiro ótimo de manufatura, onde somente tempos de processamento são considerados. Várias heurísticas de seqüenciamento para o cálculo do limitante inferior na metodologia "BAB são citadas na literatura, mas poucos são trabalhos que mostram o desenvolvimento de um programa com esta finalidade. A construção do Algoritmo A foi o primeiro passo para o desenvolvimento dos demais Algoritmos, analisados a seguir.

B - Algoritmo B modificado: Minimização do tempo total de estabelecimento via métodologia "BAB".

Mesmo com as alterações realizadas no procedimento original(Algoritmo B) de minimização do tempo total de estabelecimento via metodologia "BAB", o Algoritmo B modificado mostou-se viável a esta otimização.

Uma análise feita neste Algoritmo, mostrou que:

a) Muitas sequências iguais são completamente analisadas, em virtude da eliminação da escolha arbitrária do método original, o que corresponde a análise de vários pares promissores a minimizarem o tempo total de estabelecimento.
b) Um determinado par de tarefas tomado arbitrariamente, pode levar a um tempo de estabelecimento não mínimizado, principalmente, quando os tempos da matriz de estabelecimento estão próximos. Logo, a análise de todos os pares promissores à minimização do tempo total de estabelecimento aumenta a possibilidade de se encontrar uma seqüência minimizada;
c) O tempo de execução do programa aumenta devido à análise dos vários pares promissores à minimização do tempo total de estabelecimento.

C - Algoritmo C: Seqüenciamento de tarefas com tempos de estabelecimeto dependentes da seqüência de produção e independentes do processador .

As seqüências com tempos de estabelecimento total ótimo e sub-ótimos são canditadas à minimização do tempo total de execução . A busca heurística desenvolvida por este Algoritmo não oferece nenhuma garantia de determinação de um roteiro de manufatura ótimo, assim como qualquer procedimento heurístico. No problema com o número de tarefas elevado ($N > 10$) o tempo de exeçução mostrou-se razoavelmente elevado,

principlamente quando os tempos de estabelecimento estão próximos. No problema exemplo da seção 4.2.2, o algoritmo C gerou todas as seqüências possíveis.

D- Algoritmo D: Seqüenciamento de tarefas com tempos de estabelecimento dependentes da seqüência de produção e dependentes do processador.

As fórmulas propostas para o tratamento deste problema viabilizaram a aplicação da metodologia "BAB". A inclusão do termo que contempla os tempos de estabelecimento e da estratégia adotada no cálculo do custo dos subproblemas gerados atenderam à exigência básica do método "BAB", que é a construção de uma função de custo que reflita as características do processamento. Esta conclusão é tirada pelos problemas solucionados por D.

F- Algoritmo E: Seqüenciamento com Ordem Prioritária de Manufatura

Nesta abordagem testou-se principalmente a flexibilidade da metodologia "BAB". A formação de uma pseudo-tarefa, constituída por um grupo de tarefas a serem desevolvidas não afetou a determinação exata do roteiro ótimo tal como realizado por A. A parte crucial deste procedimento foi a determinação do instante de término de uma seqüência parcial quando esta pseudo-tarefa tem acesso à linha de processamento. A construção deste algoritmo teve como base o Algoritmo A.

G- Algoritmo C versus Algoritmo D

O problema apresentado na secção 4.5.4, caso 1, foi solucionado pelos Algoritmos C e D. Por intermédio do procedimento heurístico (Algoritmo C), todas as seqüências foram geradas(enumeração completa), o que não aconteceu com a aplicação do Algoritmo D, que gerou somente 2 seqüências completas, evidenciando a superioridade da metodologia "BAB".

Poucos são os trabalhos existentes na literatura que consideram tempos de estabelecimento, pois quando estes não são negligenciáveis, a incorporação de uma nova tarefa à seqüência parcial é bastante crítica, e sem a determinação de um roteiro ótimo, tal como realizado neste trabalho, é impossível o aumento da produtividade e diminuição de custos em Estrutura de Processamento Batelada. Com certeza, a proposta apresentada neste trabalho é uma passo para a solução de problemas que incorporem este detalhe de processamento

SUGESTÕES

Neste trabalho, com o não negligenciamento dos tempos de estabelecimento, indubitavelmente o seqüenciamento tornou-se mais realístico, tendo em vista o surgimento de tempos improdutivos (tempos mortos) que se não minimizados podem levar a uma seqüência de produção com alto custo de produção. Um outro detalhe de produção que poderia ser imediatamente incorpado ao problema é o tempo gasto para tranferir uma batelada da tarefa i de um processador j para outro j+1 . A consideração de tempos de transferência é bem oportuna a partir das fórmulas desenvolvidas na secção 4.5, já que estes tempos não tornam o seqüenciamento tão crítico como os tempos de estabelecimento.

Uma diversidade de outros detalhes de uma Linha de processamento poderiam ser incorporados ao problema, mas o importante é ressalvar que para cada incorporação, tem-se um problema com complexidade de solução diferente, sendo necessário com isso, o desenvolvimento de algoritmos específicos para a solução do problema.

NOMENCLATURA

N - Número de tarefas
M - Número de processadores
i - indicador de tarefa
j - indicador de processador
q - Seqüência parcial de tarefas
qi - Seqüência parcial de tarefas com incorporação da tarefa i
t_{ik} - tempo de processamento da tarefa i no processador k
$C_{qi,j}$ - instante de término da seqüência qi no processador j
R_j - Carga de trabalho mínima remanescente no processador j
Uj - Carga de trabalho mínima nos processadores subsequentes a j, supondo que trabalho em j já esteja terminado.
b_j - tempo mínimo para processar todas as tarefas a partir do processador j
LB - Limitante inferior em tempo de execução
te - tempo total de execução de uma seqüência de tarefas
R_{ik} - Razão de processamento da tarefa i no processador k
B_i - Batelada de uma tarefa i
U - Conjunto de tarefas a serem programadas
V_{ik} - Volume de uma batelada da tarefa i
ST - Solução tentativa
Q - Nível da árvore de busca
S - Matriz de tempos de estabelecimento
Sr - Matriz de tempos de estabelecimento reduzida
θi - valor de redução inicial
θ - valor de redução corrente
λ - soma dos valores mínimos da linha e coluna de cada elemento nulo da matriz Sr

LB* - Limitante inferior em tempo de estabelecimento

bj* - tempo mínimo para processar as tarefas no processador j

r - indicador de redução (nível da árvore de busca)

sc^{j}_{qk} - tempo de estabelecimento já comprometido pela seqüência parcial qk no processador j

$S^{j}_{k-1,k}$ - Tempo necessário para preparar o processador j para a nova tarefa k, depois que a tarefa anteriormente processada , (k-1), termine o seu proocessamento.

APÊNDICE

Apêndice A : Cálculo do instante de término da seqüência parcial 2-3-4- quando a pseudo-tarefa tem acesso à linha de processamento para o exemplo da secção 4.6.3

a) Cálculo do instante de término da seqüência parcial 2-3-4-

a.1- seqüência parcial qi = 2

$C_{qi,1} = \max\{0,0\} + 4 = 4$

$C_{qi,2} = \max\{4,0\} + 5 = 9$

$C_{qi,3} = \max\{9,0\} + 3 = 12$

a.2.- seqüência parcial qi = 2-3

$C_{qi,1} = \max\{0,4\} + 2 = 6$

$C_{qi,2} = \max\{6,9\} + 2 = 11$

$C_{qi,3} = \max\{11,12\} + 6 = 18$

a.3.- seqüência parcial qi=2-3-4

$C_{qi,1} = \max\{0,6\} + 4 = 10$

$C_{qi,2} = \max\{10,11\} + 3 = 14$

$C_{qi,3} = \max\{14,18\} + 2 = 20$

b) Cálculo do instante de término da seqüência parcial qi = 2-3-4-7

Pseudo-tarefa: 7
Desmembramento de 7 nas tarefas 6 e 1

b.1 - Cálculo do instante de término da seqüência parcial qi = 2-3-4-6

$C_{qi,1} = \max\{0,10\} + 1 = 11$

$C_{qi,2} = \max\{11,14\} + 4 = 18$

$C_{qi,3} = \max\{18, 20\} + 3 = 23$

b.2 - Cálculo do instante de término da seqüência parcial qi = 2-3-4-6-1

$C_{qi,1} = \max\{0, 11\} + 3 = 14$

$C_{qi,2} = \max\{15, 20\} + 4 = 24$

$C_{qi,3} = \max\{24, 26\} + 1 = 27$

Apêndice B: Listagem do Programa Random

Program Random

{Documentação: Este programa gera 20.000 números pseudo-randômicos. A subrotina original foi desenvolvida em Fortran por George Marsaglia e Arif Zaman (Universidade Estadual da Flórida) . Neste trabalho está sendo utilizada uma versão em Turbo Pascal do programa Random.

Neste programa está sendo feito o teste de aleatoriedade da subrotina geradora de números pseudo-randômicos. No apêndice C é mostrado a estatística da distribuição em percentagem dos numeros da faixa analisada (1 a 9). Os números obtidos são multiplicados por um fator de 10..

Descrição das Principais variáveis

ij = Primeira semente
kl = Segunda semente
temp = vetor dos números aleatórios}

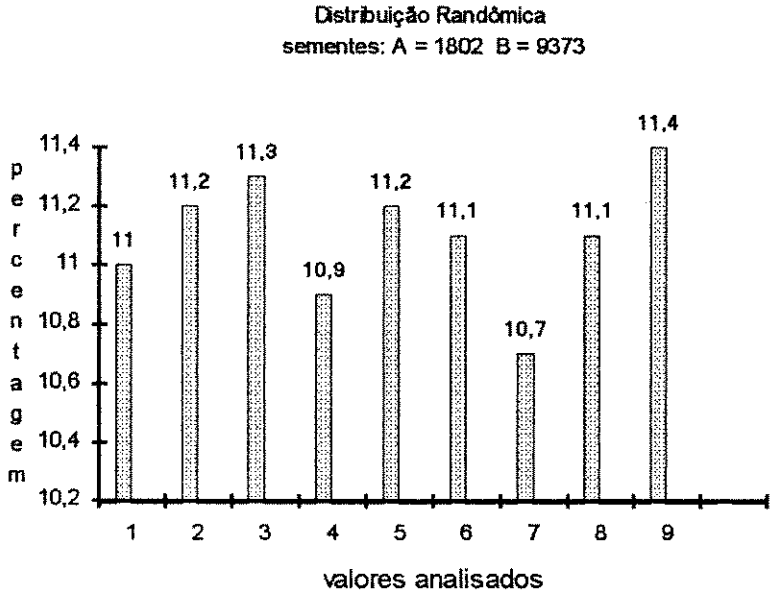
```
{Definição de tipos}
type
 vetor1 = array[1..100] of real;
 vetor2 = array[1..97] of real;

{Definição de variáveis}
var
  ij,kl,len,i97,j97,i,ii,iii          : integer;
  c,cd,cm,num,soma                    : real;
  teste                               : boolean;
  ind                                 : char;
  temp                                : vetor1;
  u                                   : vetor2;

function modi (genA,genB:real):real;
begin
 modi:= genA - (trun(genA/genB)*genB);
end;

Procedure  rmarin(ij,kl:integer;var ind: char);
 var
      i2,j2,k2,l2,s,t,m                    : real;
```

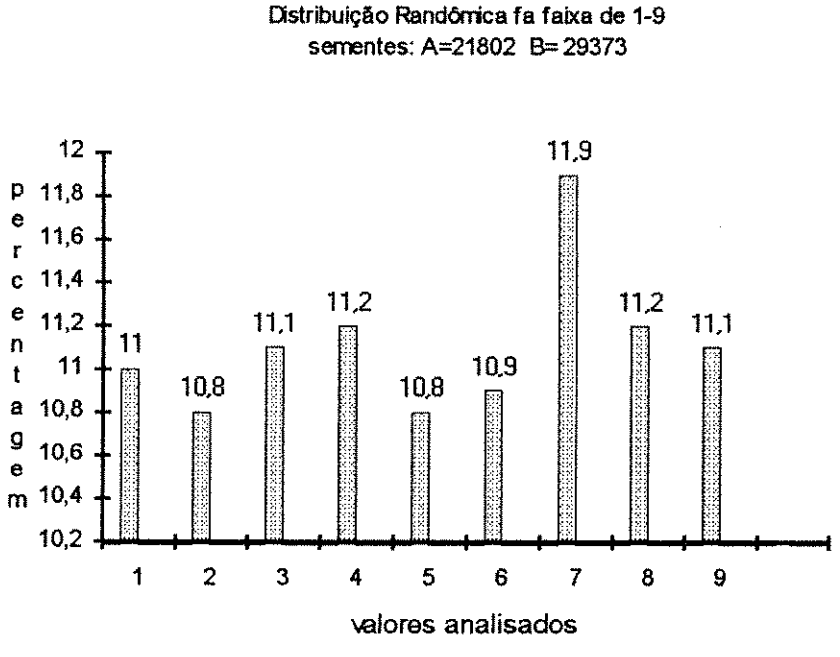
```
  ii,jj                                  : integer;
begin
ind:= 'S';
teste:= true;
if ((ij < 0) or (ij > 31328)  or (kl<0) or (kl > 30081)) then
begin
writeln ('(A)','a primeira semente deve estar na faixa 0-31328');
writeln('(B)','a segunda semente deve estar na faixa 0-30081');
ind:= 'N';
end;

if ind = 'S' then
begin
i2:= trunc(modi(ij/177,177) + 2);
j2:= trunc(modi(ij      ,177) + 2);
k2:= trunc(modi(kl/169,178) + 1);
l2:= trunc(modi(kl,169));

for ii:= 1 to to 97 do
begin
s:= 0.;
t:= 0.5;
for  jj:= 1 to 24 do
begin
m:= modi(modi(i2*j2,179)*k2,179);
i2:= j2;
k2:= m;
l2:= modi((53*l2) +1,169);
if (modi((l2*m),64) >= 32) then
s:= s+t;
t:= 0.5 * t;
end;
u[ii]:= s;
end;
end;

Procedure ranmar(len: integer; var revec: vetor1; ind:char);
var
  ivec     : integer;
  uni      : real;
begin
ind:= 'S';
if (not teste) then
```

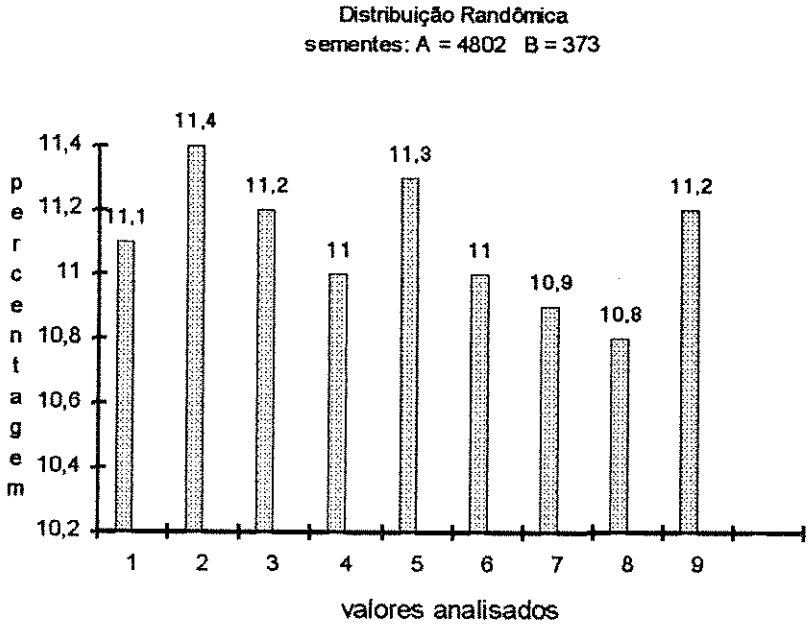
```
begin
writeln('Chamar rmarin antes de cahmar ranmar');
ind:= 'N';
end;
if ind = 'S' then
begin
for ivec:= 1 to len do
begin
if ( uni < 0) then
uni:= uni + 1;
u[i97]:= uni;
i97:= i97 - 1;
if (i97 = 0) then
j97:= 97;
c:= c - cd;
if ( c< 0) then
c:= c+ cm;
uni:= uni - c;
if ( uni < 0) then
uni:= uni + 1;
revec[ivec]:= uni;
end;
end;
end;
Begin
 clrscr;
 c:= 362436.0/16777216.0;
 cd:= 7654321/16777216.0;
 cm:= 16777213/16777216.0;
 i96:= 97;
 j97:= 33;
 ij:= 1802;
 kl:= 9373;
 rmarin(ij,kl,ind);
 if ind = 'S' then
 begin
 len:= 100;
 for i:= 1 to 200 do
 ranmar(len,tem,ind);
 for i:=  1 to 200 do
 begin
 len:= 100;
 ranmar(len,temp,ind);
 for ii:= 1 to 10 do
```

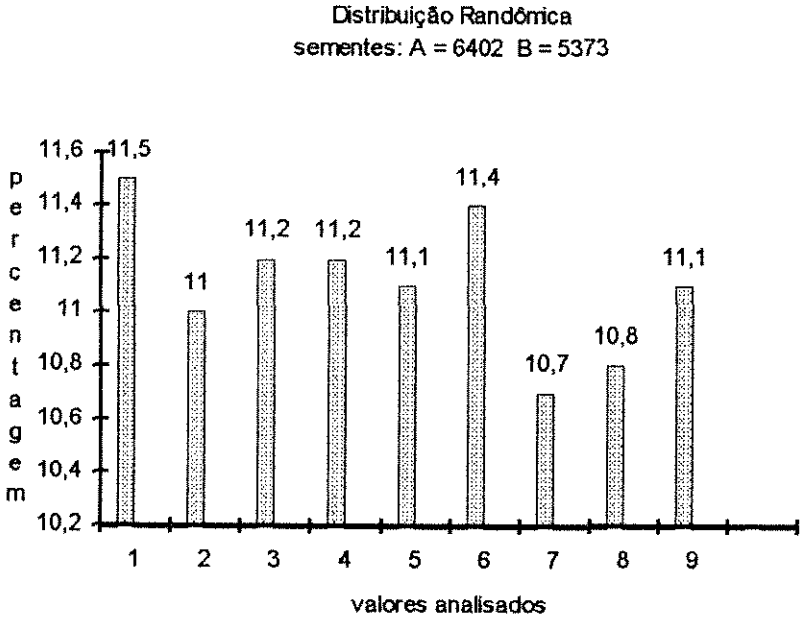
```
for iii:= 1 to 100 do
if ii = trunc(10*temp[iii]) then
vet[ii]:= vet[ii] + 1;
end;
soma:= 0.0;
for i:= 1 to 9 do
soma:= soma + vet[i];
writeln('Percentagem dos numeros');
for i:= 1 to 9 do
writeln(i,'=',((vet[i]/soma)*100):4:1);
end;
end.
```

Apêndice C: Estatística de Distribuição Randômica em Percentagem de Números (faixa de 1 a 9) obtidos por Random






Distribuição Randômica
sementes: A = 4802 B = 373
percentagem
11,4
11,2
11
10,8
10,6
10,4
10,2
11,1
11,4
11,2
11
11,3
11
10,9
10,8
11,2
1 2 3 4 5 6 7 8 9
valores analisados


Distribuição Randômica
sementes: A = 6402 B = 5373
percentagem
11,6
11,4
11,2
11
10,8
10,6
10,4
10,2
11,5
11
11,2
11,2
11,1
11,4
10,7
10,8
11,1
1 2 3 4 5 6 7 8 9
valores analisados

Apêndice D - Listagem dos Programas Desenvolvidos

Program Batjs1mo;

Tese de Mestrado

FACULDADE DE ENGENHARIA QUIMICA

Departamento de Engenharia de Sistemas Químicos - D.E.S.Q.

Descrição do Programa

Algoritmo A - Seqüenciamento de tarefas com tempos de estabelecimento independentes da Seqüência de Produção

Descrição das variáveis principais

N	número de tarefas a serem programadas
M	número de processadores presentes na planta batelada
Q	plano da árvore corrente
seqüencia	lista das tarefas programadas
upbound1	valor da função "makespan" da seqüência canditada à solução ótima upper bound")
novomestre	nó filho selecionado
time	tempos de processamento das tarefas
mplano	árvore de busca
mprodback	histórico dos nós sondados

Descrição das unidades

nosfilhos	unidade que expande um nó mestre, gerando nós filhos
robat81	avalia o "upper bound"
maior_va	maior valor de um vetor, usada em nosfilhos}

```
Uses nosfilhos_2,maior_va_1,fixa_4,robat81,printer,dos,crt;

{Definição de tipos}
type
str12  = string[15];
{Definição de variáveis}
var
n,m,q,i,j,prod1,qaux                    : integer;
upbound1,upboundaux,novomestre          : real;
ind1,ind2,ind3,ind5                     : char;
t1                                      : str12;
hora1,minuto1,segundo1,seg1001          : word;
hora2,minuto2,segundo2,seg1002          : word;
tempo                                   : word;
sequencia,seqotima                      : vetorbat81;
c,time                                  : matbat81;
```

```
c,time                                  : matbat81;
mplano, mprodback                       : matbat2;
nosondados                              : integer;

{Ler arquivo}
Procedure learquivo1(var tt: matbat81;nome: str12;
var n :integer);
var
f1      : text;
ch1   :char;
ncs1: string[2];
i1,j1,k1,e1 : integer;
s1 : array[1..bat1,1..bat2] of string[8];
begin
assign(f1,nomc);
reset(f1);
for i1 := 1 to bat1 do
for j1 := 1 to bat2 do
s1[i1,j1] := '';
ni1 := 0;
j1 := 1;
k1 := 0;
ncs1:='';
while not eof(f1) do
begin
read(f1,ch1);
if (k1 <> 2) and (ch1 <> #13) then
begin
k1:= k1 + 1;
ncs1:= ncs1+ch1;
val(ncs1,n,e1);
end
else
if ch1=#32 then
j1:= j1+1
else
if ch1=#13 then
begin
i1 := i1+1;
j1 := 1;
k1:= 2;
end
else
if ch1<>#10 then
s1[i1,j1] := s1[i1,j1]+ch1;
end;
for i1 := 1  to n do
for j1 := 1 to m do
vl(s1[i1,j1],tt[i1,j1],e1);
```

```
close(f1);
end;
{FIM}

Procedure Retirar_tarefa_2(q:integer;sequencia:vetorbat81);
var j2,prod2: integer;
begin
for j2:= 1 to N do
if mplano[q,j2] >= upbound1 then
mplano[q,j2]:=0;
end;

Procedure Imprima_sequencia_4(sequencia:vetorbat81;upbound1:real);
var i4 :integer;
begin
write(lst,'sequencia: ');
for i4:= 1 to n do
write(lst,sequencia[i4],'-');
writeln(lst);
writeln(lst,'te = ',upbound1:4:1);
end;

Procedure Ver_Plano1_3(n3,q3:integer;sequencia3:vetorbat81;var ind3:char;
var novomestre:real);
var i3,j3          : integer;
    boundaux3  : real;
begin
novomestre:=0;
{Procedimento para retirada de tarefas com menor limitante superior ao
upbound-Processo "fathomed"}
retirar_tarefa_2(q3,sequencia3);
ind3:= 'N'; {lista vazia no plano 1}
for j3:= 1 to n3 do
if mplano[q3,j3] <> 0 then
begin
novomestre:= j3;
boundaux3:= mplano[q3,j3];
j3:= n3;
ind3:= 'S';
end;
for j3:= 1 to n3  do
if (mplano[q3,j3] <> 0) and (mplano[q3,j3] < upbound1) and
(mplano[q3,j3] < boundaux3) then
begin
novomestre:= mplano[0,j3];
boundaux3:= mplano[q3,j3];
ind3:= 'S';
end;
if novomestre <> 0 then
```

```
begin
mplano[q3,trunc(novomestre)]:=0;
mprodback[q3,trunc(novomestre)]:= novomestre;
end;
end;

Procedure Ver_Plano_atual_5(n4,q4:integer;sequencia4:vetorbat81;
var ind4:char;var novomestre:real);
var i4,j4,j44  : integer;
boundaux4  : real;
indicador4 : char;
begin
{este procedimento encontra o mais baixo valor para LB - menor limitante}
novomestre:=0;
{Procedimento para retirada de tarefas com menor limitante superior ao
upbound - Processo "fathomed"}
retirar_tarefa_2(q4,sequencia4);
ind4:= 'N';  {lista vazia no plano > 1}
indicador4:= 'N';
for j4:= 1 to n4 do
if mplano[q4,j4] <> 0 then
begin
novomestre:= j4;
boundaux4:= mplano[q4,j4];
ind4:= 'S';
if ind5 <> 'N' then
upboundaux:= mplano[q4,j4];   {melhor solucao factivel}
indicador4:= 'S';
j44:=j4;
j4:= n4;
end;
if indicador4 = 'S' then
begin
for j4:= j44 to N4  do
if (mplano[q4,j4] <> 0) and (mplano[q4,j4] < upbound1) and
(mplano[q4,j4] < boundaux4)  then
begin
novomestre:= mplano[0,j4];
boundaux4:= mplano[q4,j4];
if ind5 <> 'N' then
upboundaux:= mplano[q4,j4];   {melhor solucao factivel}
ind4:= 'S';
end;
if (ind5 = 'N') and (q = N) then  {primeira solucao factivel}
begin                                {sem contar com a solucao tentativa}
ind5:= 'S';
upbound1:= mplano[n,trunc(novomestre)];
upboundaux:= upbound1;
end;
```

```
mplano[q4,trunc(novomestre)]:=0;
mprodback[q4,trunc(novomestre)]:= novomestre;
end;
end;

Begin                {inicio Batjs1mo}
clrscr;
{leitura dos dados}
write(':6,'Entre com o numero de processadores M =  ');
readln(M);
write(':6,'Entre com o nome do arquivo = ');
readln(t1);
learquivo1(time,t1,n);
{Impressao da matriz tempo}
writeln(lst);
writeln(lst,':20,'tempos de Processamento');writeln(lst);
writeln(lst,':28,'Processadores');writeln(lst);
write(lst,':27);
for j:= 1 to m do
write(lst,j,':6);writeln(lst);
for i:= 1 to n Do
begin
write(lst,':25);
for j:= 1 to m do
write(lst,time[i,j]:4:1,'   ');
writeln(lst);
end;
write(lst);
write(lst);
writeln;
writeln('Programa em execucao, aguarde...');
writeln(lst,'Sequencias analisasdas');
writeln(lst);

{*** inicializacao das variaveis ***}
{inicializar sequencia}
for i:= 1 to (bat1+1) do
sequencia[i]:=0;
{inicializar variaveis da arvore}
for i:= 0 to bat1 do
for j:= 1 to bat1 do
mplano[i,j]:= 0;
{inicializar matriz de tarefas analisadas}
for i:= 0 to bat1 do
for j:= 1 to bat1 do
mprodback[i,j]:=0.;
nosondados:= 0;

{atribuir primeira linha de mplano com as tarefas}
```

```
for j:= 1 to N do
mplano[0,j]:=j;

{*** criar primeiro plano da arvore de busca ***}
{**tempo de inicio}
gettime(hora1,minuto1,segundo1,seg1001);
{setando q para primeiro plano da arvore}
q:= 1;
qaux:=q;  {qaux indicara a sequencia parcial inicial}
nosondados:= n;
gerar_filhos(n,m,q,time,sequencia,mplano);
calc_upbound(n,m,time,mplano);}

{**** Calculo do "upper bound" ****}
{Montar uma sequencia de permutacao}
for i:= 1 to n do
sequencia[i]:= i;
for i:=1 to n do
seqotima[i]:= sequencia[i];
{calcular upbound1}
calc_upbound1(n,m,sequencia,time,upbound1);
upboundaux:= upbound1;
{*** fim do calculo ***}
ind1:= 'S';
ind5:= 'N';  {indicara se o primeiro upbound foi encontrado entao
ind4:= 'S'}
while ind1 <> 'N' do
begin
for i:= 2 to n do
for j:= 1 to n do
mplano[i,j]:=0;
{Buscar novo no mestre}
ver_Plano1_3(n,q,sequencia,ind2,novomestre);
if ind2 = 'N' then
ind1:= 'N'        {fim do programa}
else
begin
sequencia[q]:= trunc(novomestre);
q:= succ(q);
while q <> qaux do
begin
{Zerar sequencia de q ate n}
for i:= q to n do
sequencia[i]:= 0;
{explodir nomestre - gerar nos filhos}
if q <= n then
gerar_filhos(n,m,q,time,sequencia,mplano);
if (q-1) = N then
begin
```

```
q:= q-2;
for i:= (q+1) to n do
for j:= 1 to n do
mprodback[i,j]:=0;
for i:= q to n do
sequencia[i]:= 0;
end;
{Eliminar os nos das tarefas ja explodidos}
for i:=1 to n do
if mprodback[q,i] <> 0 then
mplano[q,i]:= 0;
{Buscar novomestre- elimina os menores limitante maiores que upbound1}
ver_plano_atual_5(n,q,sequencia,ind3,novomestre);
if ind3 = 'N' then
begin
{retroceder no plano}
q:= pred(q);
for i:= (q+1) to n do
for j:= 1 to n do
mprodback[i,j]:=0;
end
else
begin
sequencia[q]:= trunc(novomestre);
nosondados:= nosondados + (n-q) +1;
if q = n then
begin
if upboundaux <= upbound1 then
begin
if upboundaux < upbound1 then
upbound1:= upboundaux;
for i:= 1 to n do
seqotima[i]:= sequencia[i];
imprima_sequencia_4(sequencia,upbound1);
end;
end;{fim do if q = n}
q:= succ(q);
end; {fim do else}
end; {fim do while}
end; {fim do else1}
end; {fim while}
gettime(hora2,minuto2,segundo2,seg1002);
tempo:= ((hora2-hora1)*3600) + ((minuto2-minuto1)*60) +
(segundo2 -segundo1);
writeln(lst);
writeln(lst);*
```

```
writeln(lst,'Recurso computacional: Computador 486 DX');
writeln(lst,'Linguagem de programacao: Turbo Pascal');
writeln(lst,'Tempo de execucao: ',tempo,' ','segundos');
writeln(lst);
writeln(lst);
writeln(lst,'Programa de Producao');
writeln(lst);
write(lst,'sequencia otima: ');
for i:= 1 to n do
write(lst,seqotima[i],'-');
write(lst,'   ');
writeln(lst,'te = ',upbound1:3:1,' u.t.');
writeln('fim do programa');
writeln(lst);
writeln(lst);
fixatela;
End.                    {fim Batjs1}
```

```
Unit Nosfilhos_2;
Interface
{       Autor: Edilson J. Santos - UNICAMP - 110693

                        Tese de Mestrado

                FACULDADE DE ENGENHARIA QUÍMICA

        Departamento de Engenharia de Sistemas Químicos - D.E.S.Q.

Descrição do Programa

Esta unidade expande um nó mestre, gerando nós filhos

Descrição das variáveis principais

Obs: O menor limitante é calculado pela heurística Full Machine Based Bound
Menor limitante B = max(dj)  j = 1..M

Q2          - Indica o plano corrente da árvore de busca
N2          - número de tarefas a serem programadas
M2          - número de processadores presentes na planta batelada
rj2         - carga de trabalho remanescente na máquina corrente
dj2         - vetor da função custo avaliada em cada máquina
bj2         - vetor dos valores máximo de dj - "lower bound"
c2          - matriz de tempos de execução de sequências parciais
matu2       - carga de trabalho mínima dos trabalhos nao sequenciados
sequencia2  - lista das tarefas programadas
time        - tempos de processamento das tarefas
```

```
mplano2      - arvore de busca}

uses maior_va_1,menor_va_3,robat81,crt;

{Descrição de tipos}
type
vetorbat2 = array[1..bat1] of real;
matbat2   = array[0..bat1,1..bat1] of real;

{Descrição das variáveis}
var
q2,i2,j2,ppos2,pant2,pos2,m2,n2              : integer;
max12,max22,soma2,mindj2                     : real;
rj2,vetor_min2                               : vetorun1;
c2                                           : matbat81;
vetaux12                                     : vetorun2;
vbat12                                       : vetorun1;
dj2                                          : vetorun1;
bj2                                           vetorbat2;
matU2                                        : matbat81;
Procedure Gerar_filhos(n2,m2,q:integer;time2:matbat81;sequencia2:vetorbat81;
 var mplano2:matbat2);

Implementation
Procedure Gerar_filhos(n2,m2,q:integer;time2:matbat81;sequencia2:vetorbat81;
var mplano2:matbat2);

Procedure Armaz_Lin(ppos2:integer; var vetorbat3: vetorUn1);
Var
j1 : Integer;
begin
For J1:= 1 to M2 Do
vetorbat3[J1]:= Time2[Ppos2,J1];
end;

Procedure Zerar_Lin(Ppos2: integer);
Var
J: integer;
begin
For J:= 1 to M2 Do
Time2[Ppos2,J]:=0.;
end;

Procedure Restituir_Lin(Ppos2: integer);
Var
J : integer;
begin
For J:= 1 to M2 Do
Time2[Ppos2,J]:= vbat12[J];
```

```
end;

Procedure Calc_RJ(A: Matbat81;N,M:integer;Var RJ: VetorUn1);
Var I,J : integer;
begin
For J:= 1 to M Do
begin
Rj[J]:= 0.;
For I:= 1 to N Do
Rj[J]:= Rj[J] + A[I,J];
end;
end;

Procedure Calc_U(B: Matbat81;N,M:integer;Var MatU: Matbat81);
Var J,J1,K1,K2: integer;
begin
K1:= 0;
For J:= 1 to (M-1) Do
begin
For K2:= 1 to N Do
begin
MatU[K2,J]:= 0.;
For J1:= 2 to M-K1 Do
MatU[K2,J]:= MatU[K2,J] + B[K2,J1+K1];
end;
K1:= K1 + 1;
end;
end;

{Procedimento para a calculo do trabalho minimo,remanescente, Umin,considerando os
Processadores J+1..M}
Procedure Calc_UMin(MatU:matbat81;P,N,M:Integer;Var Vetor_Min: VetorUn1);
Var VetorU    : vetorUn1;
Min        .: Real;
Pos,j1,i1  : Integer;
Begin
For J1:= 1 to (M-1) Do
Begin
For I1:= 1 to N Do
VetorU[I1]:= MatU[I1,J1];
busca1(VetorU,N,Min);
Vetor_Min[J1]:= Min;
end;
End;
{FIM}
```

```
{Procedimento para a calculo dos Dj em cada Processdor}
Procedure Calc_DJ(D :Matbat81;RJ,Vetor_Min: VetorUn1;Var DJ: vetorUn1);
Var J,I: integer;
Begin
For J:= 1 to M2 Do
DJ[J]:= RJ[J] + D[Ppos2,J] + Vetor_Min[J];
End;
{FIM}

{Procedimento para a calculo do Menor Limitante em para cada sequencia gerada}
Procedure Max_DJ(Bj:vetorUn1;M: Integer;var Max:real);
Var
k: Integer;
Begin
max:=BJ[1];
For k:= 2 to M Do
If Bj[k] > Max then
max:= BJ[k];
end;
{FIM}

Procedure Deter_tempo_seqparcial(q:integer;time2:matbat81;sequencia2:vetorbat81;
var c2:matbat81);
var i2,j2,Pant2,Ppos2          : integer;
begin
for i2:= 1 to N2 Do
begin
soma2:= 0.;
For j2:=1 to M2 do
begin
c2[i2,j2]:= soma2 + time2[i2,j2];
soma2:=  c2[i2,j2];
end;
end;
for i2:= 1 to (q-1) do
begin
Ppos2:= Sequencia2[i2];
{Calculo do Tempo de execucao da sequencia parcial C2}
if i2 <> 1 Then
begin
Pant2:= sequencia2[i2-1];
For J2:= 1 TO m2 Do
begin
if J2 = 1 Then
begin
vetaux12[1]:= 0.;
vetaux12[2]:= c2[Pant2,J2];
end
else
```

```
begin
vetaux12[1]:= c2[ppos2,j2-1];
vetaux12[2]:= c2[pant2,j2];
end;
Busca(vetaux12,2,max22);
c2[ppos2,j2]:= max22 + time2[ppos2,J2];
end;
end;   {fim if i1}
zerar_lin(ppos2);
end; {fim enquanto }
end;

BEGIN                          {Inicio nosfilhos};
for I2:=1 To M2 Do
vetor_Min2[I2]:=0;
pant2:=0;
for i2:= 1 to N2 do
bj2[i2]:= 0;
for Q2:= Q to Q Do
begin
{Determinacao do tempo de execucao da sequencia parcial}
deter_tempo_seqparcial(q,time2,sequencia2,c2);
for ppos2:= 1 to n2 do    {se sequencia ja tem 1-2 nao faz}
begin
for I2:= 1 To Q Do
If (ppos2 = sequencia2[I2]) Then
pant2:= sequencia2[I2];
If pant2  <> ppos2 Then
begin
{Calculo do Tempo de Completacao C}
if q <> 1 Then
begin
pant2:= Sequencia2[Q2-1];
for J2:= 1 TO M2 Do
begin
if J2 = 1 Then
begin
vetaux12[1]:= 0.;
vetaux12[2]:= c2[Pant2,J2];
end
else
begin
Vetaux12[1]:= C2[Ppos2,J2-1] ;
Vetaux12[2]:= C2[Pant2,J2];
end;
Busca(Vetaux12,2,Max22);
C2[Ppos2,J2]:= Max22 + Time2[Ppos2,J2];
end;
end;
```

```
{Armazenando Linha}
Armaz_lin(Ppos2,Vbat12);
{Zerando Linha}
Zerar_Lin(Ppos2);
{Chamando Calc_RJ}
Calc_RJ(time2,n2,m2,rJ2);
{Chamando Calc_U}
Calc_U(time2,n2,m2,matU2);
{Restituindo linha}
Restituir_Lin(ppos2);
{Chamando Calc_UMin}
Calc_umin(MatU2,Ppos2,N2,M2,Vetor_Min2);
Calculando os DJ}
alc_dj(c2,rJ2,vetor_min2,dJ2);
{Calculando Maximo dos DJ}
max_dj(dJ2,m2,max12);
bj2[ppos2]:= max12;
end             {fim if}
end;                {fim_ppos2}
end;                    {fim de Q}
for j2:= 1 to N2 do
mplano2[q,j2]:= bj2[j2];
end;
end.        {fim nos nosfilhos}
```

Unit robat81;
Interface
{ **Tese de Mestrado**

FACULDADE DE ENGENHARIA QUIMICA

Departamento de Engenharia de Sistemas Quimicos - D.E.S.Q.

Descricao da Unidade

Esta unidade calcula o tempo de execução da solução tentativa ("upper bound")

Descrição das variáveis principais

N - numero de tarefas a serem programadas
M - numero de processadores presentes na planta batelada
Obs: para outras variaveis ver unidade nosfilho}

```
uses maior_va_1,menor_va_3,fixa_4,crt;

{Definição de Tipos}
type
vetorbat81  = array[1..bat1+1] of integer;
```

```
vetorbat82  = array[1..bat1] of real;
matbat81    = array[1..bat1,1..bat2] of real;

{Definição das variáveis}
var
indbat8,i8,j8,q8,ppos8,pant8,pos8      : integer;
max18,max28,soma8                      : real;
rj8,vetor_min8                         : vetorUn1;
c8                                     : matbat81;
vetaux18                               : vetorUn2;
vbat18                                 : vetorUn1;
dj8                                    : vetorUn1;
matu8                                  : matbat81;
vtc                                    : vetorbat82;
Procedure Calc_upbound1(n,m:integer;vbat4:vetorbat81;
time8:matbat81;var tc:real);

Implementation
Procedure Calc_upbound1(n,m:integer;vbat4:vetorbat81;
time8:matbat81;var tc:real);

Procedure Zerar_Lin(P: integer);
Var
J: integer;
begin
For J:= 1 to  M Do
time8[P,J]:=0.;
end;

{Procedimento para a calculo do trabalho remanescente em um dado Processador}
Procedure Calc_RJ(A:Matbat81;N,M:integer;Var RJ: VetorUn1);
Var I,J : integer;
begin
For J:= 1 to M Do
begin
Rj[J]:= 0.;
For I:= 1 to N Do
Rj[J]:= Rj[J] + A[I,J];
end;
end;
{FIM}

Procedure Calc_U(B: Matbat81;N,M:integer;Var MatU: Matbat81);
Var J,J1,K1,K2: integer;
begin
K1:= 0;
For J:= 1 to (M-1) Do
begin
For K2:= 1 to N Do
```

```
begin
MatU[K2,J]:= 0.;
For J1:= 2 to M-K1 Do
MtU[K2,J]:= MatU[K2,J] + B[K2,J1+K1];
end;
K1:= K1 + 1;
end;
end;

{Procedimento para a calculo do trabalho minimo remanescente, Umin,considerando os
Processadores J+1..M}
Procedure Calc_UMin(MatU:matbat81;P,N,M:Integer;Var Vetor_Min: VetorUn1);

Var VetorU    : vetorUn1;
Min           : Real;
Pos           : Integer;
Begin
For J8:= 1 to (M-1) Do
Begin
For I8:= 1 to N Do
VetorU[I8]:= MatU[I8,J8];
BUSCA1(VetorU,N,Min);
Vetor_Min[J8]:= Min;
end;
End;
{FIM}

{Procedimento para a calculo dos Dj em cada Processdor}
Procedure Calc_DJ(D :Matbat81;RJ,Vetor_Min: VetorUn1;Var DJ: vetorUn1);
Var J,I: integer;
Begin
For J:= 1 to M Do
DJ[J]:= RJ[J] + D[Ppos8,J] + Vetor_Min[J];
End;
{FIM}

{Procedimento para a calculo do Menor Limitante em para cada sequencia gerada}
Procedure Max_DJ(Bj:vetorUn1;M: Integer;var Max:real);
Var
I,k: Integer;
Begin
max:=BJ[1];
For k:= 2 to M Do
If Bj[k] > Max then
max:= BJ[k];
end;
{FIM}
```

```
BEGIN                          {Inicio unBatch8};
{Calculo do tempo de Completacao}
For I8:= 1 to N Do
begin
Soma8:=0.;
For J8:=1 to M do
begin
C8[I8,J8]:= Soma8 + time8[I8,J8];
Soma8:=  C8[I8,J8];
end;
end;
For I8:=1 To M Do
Vetor_Min8[I8]:=0;
Pant8:= 0;
For Q8:= 1 to N Do
begin
indbat8:= vbat4[Q8];
For Ppos8:= indBat8 to indbat8 Do
begin
{Calculo do Tempo de Completacao C}
if Q8 <> 1 Then
begin
For J8:= 1 to M Do
begin
if J8 = 1 then
begin
Vetaux18[1]:= 0.;
Vetaux18[2]:= C8[vbat4[Q8-1],J8];
end
else
begin
Vetaux18[1]:= C8[Ppos8,J8-1] ;
Vetaux18[2]:= C8[vbat4[Q8-1],J8];
end;
Busca(Vetaux18,2,Max28);
C8[Ppos8,J8]:= Max28 + time8[Ppos8,J8];
end;
end;
{Zerando Linha}
Zerar_Lin(Ppos8);
{Chamando Calc_RJ}
Calc_RJ(time8,N,M,RJ8);
{Chamando Calc_U}
Calc_U(time8,N,M,MatU8);
{Chamando Calc_UMin}
Calc_Umin(MatU8,Ppos8,N,M,Vetor_Min8);
{Calculando os DJ}
Calc_DJ(C8,RJ8,Vetor_Min8,DJ8);
Calculando Maximo dos DJ}
```

```
Max_DJ(DJ8,M,Max18);
end;                    {Fim_Ppos}
Zerar_Lin(indbat8);
vtc[Q8]:= max18;
end;                    {Fim de Q}
tc:= vtc[N];
{writeln(' ':3,' tc = ',tc:4:1);}
end;
END                     {Fim unBatch8}
```

```
Unit menor_va_3;
Interface
uses maior_va_1;
type
vetorun1 = array[1..bat1] of real;
var
unt2: integer;
Procedure Busca1(un1: vetorun1; unt3: integer;var menorunt:real);
Implementation
Procedure Busca1(un1: vetorun1; unt3: integer;var menorunt:real);
begin
menorunt:=0.;
for unt2:= 1 to unt3 do
if un1[unt2] <> 0.  then
menorunt:= un1[unt2];
for unt2:= 1 to unt3 do
If (un1[unt2] <> 0) and (un1[unt2] < menorunt) then
menorunt:= un1[unt2];
end;
end.
```

```
Unit maior_va_1;
Interface
const
bat1 = 10;
bat2 = 10;
type
vetorun2 = array[1..2] of Real;
var
unt5 : integer;
Procedure Busca(un2:vetorun2;N2:integer;var maiorunt:real);
Implementation
Procedure Busca(un2:vetorun2;N2:integer;var maiorunt:real);
begin
maiorunt:= un2[1];
for unt5:= 2 to N2 do
If un2[unt5] > maiorunt then
maiorunt:= un2[unt5];
end;
```

```
end.

Unit Fixa_4;
Interface
{Descrição da unidade}
{Esta unidade fixa a tela após apresentação dos resultados}
uses crt;
var
inkey: char;

Procedure fixatela;

Implementation
Procedure fixatela;
Begin
Repeat
inkey:= Readkey;
until inkey = ' ';
end;
end.
```

Program Batjs2;

Tese de Mestrado

FACULDADE DE ENGENHARIA QUÍMICA

Departamento de Engenharia de Sistemas Químicos - D.E.S.Q.

Descrição do Programa

Algoritmo B modificado - Minimização do Tempo total de estabelecimento via metodologia Branch and Bound

Descrição das variáveis principais

r	- indicador de reducao
nposi e nposj	- par de tarefas a ser escalonado
laux1	- matriz reduzida, geradora de matzero
matzero	- matriz de rótulos maiores
bound_inicial	- custo ótimo da melhor seqüência encontrada
bound	- custo das seqüências parciais e completas
custo	- custo avaliado pela unidade unbatch6

{Descrição das unidades}
unbatch6 - calcula o custo da seqüência tentativa.}

```
uses unbatch6,fixa,dos,crt,printer;

{Definição de tipos}
type
str12          = string[15];
vetorbat3      = Array[1..bat16] of real;
matbat4        = array[1..bat16,1..2] of integer;
matbat5        = array[1..bat16,1..bat16] of integer;

{Definição de variáveis}
var
r,zer,n,p,i,j,pos,nposi,nposj,posi,posj,fac          : integer;
maior_rotulo,bound_inicial,bound,custo               : real;
ind1,ind2,ind3,ind4,ind5,ind6,ind7                   : char;
cor1,cor2                                            : byte;
hora1,minuto1,segundo1,seg1,tempo                    : word;
hora2,minuto2,segundo2,seg2                          : word;
vmenor_lin,vmenor_Col                                : vetorbat3;
clean,laux1,matzero                                  : unmatbat3;
mreducao,mprodfac                                    : matbat5;
pares                                                : matbat4;
sequencia,seqotima                                   : vetorbat61;
l                                                    : str12;
```

```
rc                                                    : string[10];

{Início dos Procedimentos}
{Leitura do arquivo de tempos de estabelecimentos}
Procedure learquivo(var tt :unmatbat3;nome: str12;
var n :integer);
var
f               : text;
ch              : char;
ncs             : string[2];
i,j,k,e         : integer;
s : array[1..bat16,1..bat16] of string[8];
begin
assign(f,nome);
reset(f);
for i := 1 to bat16 do
for j := 1 to bat16 do
s[i,j] := '';
i := 0;
j := 1;
k := 0;
ncs:='';
while not eof(f) do
begin
read(f,ch);
if (k <> 2) and (ch <> #13) then
begin
k:= k + 1;
ncs:= ncs+ch;
val(ncs,n,e);
end
else
if ch=#32 then
j:= j+1
else
if ch=#13 then
begin
i := i+1;
j := 1;
k:= 2;
end
else
if ch<>#10 then
s[i,j] := s[i,j]+ch;
end;
for i := 1  to n do
for j := 1 to n do
val(s[i,j],tt[i,j],e);
close(f);
```

```
end;
{Fim}

{Procedimento Reducao-Linha inicial da matriz L}
Procedure Sub_Linha(I,N:integer;L:unmatbat3;menor:real;
var Laux2:unmatbat3);
var
j: integer;
begin
for j:= 1 to N do
if (j <> i) and (l[i,j] > 0.0) then
laux2[i,j]:= L[i,j] - menor;
end;

Procedure Reducao_Linha1(N:integer;L:unmatbat3;Var Laux1:unmatbat3;
var vmenor_Lin:vetorbat3);
var
i1,j1          :Integer;
menor       :real;
begin
for i1:= 1 to n do
vmenor_lin[i1]:= 0.;
for i1:= 1 to n do
for j1:= 1 to n do
If i1 = j1 then
Laux1[i1,j1]:= -1.;
for i1:= 1 to n Do
begin
menor:= 1000.;
for j1:= 1 to N Do
If (j1<>i1) and (l[i1,j1] > 0. ) then
if l[i1,j1] <= menor then
begin
menor:= l[i1,j1];
vmenor_Lin[I1]:=L[i1,j1];
end;
Sub_Linha(I1,N,L,menor,Laux1)
end;
end;
{Fim}

{Procedimento Reducao-Coluna inicial da matriz L}
Procedure Sub_Coluna1(J,N:integer;Laux1:unmatbat3;menor:real;
var vaux3:vetorbat3);
var
K1          : integer;
begin
For k1:= 1 to N do
if (k1 <> J) and (clean[k1,J] >= 0.0) then
```

```
vaux3[k1]:= Laux1[k1,J] - menor
else
vaux3[k1]:= -1;
end;

Procedure ReducaoCol1(N:integer;Laux1:unmatbat3;
var vmenor_Col:vetorbat3;var Laux3:unmatbat3);
var
I1,J1,Pos,K1,k2      : integer;
ind1  '              : char;
menor                : real;
vaux3                : vetorbat3;
begin
for i1:= 1 to n do
vmenor_col[I1]:= 0.;
for J1:= 1 to n do
begin
ind1:= 'N';
for i1:= 1 to n do
if laux1[i1,j1] = 0. then
ind1:= 'S';
if ind1 = 'S' then
for k1:=1 to n do
laux3[k1,j1]:= laux1[k1,j1];
if ind1 = 'N' then
begin
menor:=100000.;
for k1:= 1 to n Do
If (j1<>k1) then
if laux1[k1,J1] <= menor then
begin
pos:=k1;
menor:= laux1[k1,J1];
end;
vmenor_col[J1]:=Laux1[Pos,J1];
sub_coluna1(J1,N,Laux1,menor,vaux3);
for k2:=1 to N do
laux3[k2,J1]:= vaux3[k2];
end;
end;
end;
{Fim}

{Procedimento Calculo do Bound}
Procedure Calc_Bound(N:integer;vmenor_lin,vmenor_col:vetorbat3);
var
I,J          :integer;
begin
For I:= 1 to N do
```

```
Bound:= Bound + vmenor_lin[I];
For J:= 1 to N do
Bound:= Bound + vmenor_col[J];
end;
{Fim}

{Procedimeto Busca Menor Linha-Coluna}
Procedure B_MenorLinCol(I,J,N:integer;Laux3:unmatbat3;var Matzero:unmatbat3);
var
K1                              :integer;
menor_linha,menor_coluna  :real;
begin
menor_linha:= 1000.;
For K1:= 1 to N do
if (laux3[I,K1] >= 0.0) and (k1 <> J) then
if laux3[I,K1] < menor_linha then
menor_linha:= laux3[I,K1];
if menor_linha = 1000. then
menor_linha:= 0.;
menor_coluna:= 1000.;
for k1:= 1 to n do
if (laux3[k1,j] >= 0.0) and (k1 <> i) then
if laux3[k1,j] < menor_coluna then
menor_coluna:= laux3[k1,j];
if menor_coluna = 100000.0 then
menor_coluna:= 0.;
matzero[I,J]:= menor_linha + menor_coluna;
end;
{Fim}

{Procedimento Calculo da Matriz de pares de produtos factiveis}

Procedure Calc_MaiorZero(N:integer;Laux3:unmatbat3;
var Matzero:unmatbat3);
var
i,j         :integer;
begin
for i:=1 to n do
for j:=1 to n do
matzero[i,j]:= -1;
for i:=1 to n do
for j:=1 to n do
if laux3[i,j] = 0 then
B_menorLinCol(i,j,n,laux3,matzero);
end;
{Fim}
```

```
{Procedimeto Busca maior par de Produto factivel}

Procedure Calc_Maior_ElementoMatzero(N:integer;Matzero:unmatbat3;
var PosI,PosJ:integer);
var
i,j       : integer;      {calcula o maior elemento de}
maior     : real;         { matzero Matzero[pos1,posj] tem}
begin                     { o rotulo maior}
maior:= -1;
for i:=1 to n do
for j:= 1 to n do
if matzero[i,j] <> -1 then
if matzero[i,j] > maior then
begin
maior:= matzero[I,J];
posi:= i;
posj:= j;
end;
end;
{Fim }

{7 -} Procedure montar_mreducao(r,n,posi,posj:integer;matzero:unmatbat3);
var
i,j               : integer;
maior_rotulo      : real;
begin
maior_rotulo:= matzero[posi,posj];
for i:= 1 to n do
for j:= 1 to n do
if matzero[i,j] = maior_rotulo then
begin
mreducao[r,i]:= i;
j:= n;
end;
end;
{Fim}

{Procedimento Reducao Linha2}
Procedure ReducaoLinha2(N,rr: integer;Laux4:unmatbat3;
var vmenor_Linha:vetorbat3;var Laux5:unmatbat3);
var
k1,i,j,Pos        :integer;
ind1,ind2         :char;
menor             :real;
begin
for i:= 1 to n do
for j:= 1 to n do
laux5[I,J]:= laux4[I,J];
for i:= 1 to n do
```

```
vmenor_Linha[I]:= 0.;
for i:= 1 to n do
begin
ind2:= 'N';
ind1:= 'N';
for k1:= 1 to rr do
if i = pares[k1,1] then
ind1:= 'S';
for k1:= 1 to n do
if laux4[i,k1] = 0 then
begin
ind2:= 'S';
k1:= n;
end;
if ind1 = 'S' then
if (ind2 = 'S')  then
for k1:= 1 to N do
laux5[i,k1]:= laux4[i,k1];
if ind1 = 'N' then
if (ind2 = 'N') then
begin
menor:= 100000.;
For j:= 1 to n do
begin
if j <> i then
begin
if (Laux4[I,J] < menor) and (Laux4[I,J] <> -1) and
Laux4[I,J] <> 0) then
begin
pos:= j;
menor:= Laux4[I,J];
end;
vmenor_Linha[I]:=Laux4[I,Pos];
end;
sub_Linha(I,N,Laux4,menor,Laux5);
end;
end;
end;
end;
{Fim }

{Procedimento Reducao de Coluna2}
Procedure Sub_Coluna2(J,N:integer;Laux5:unmatbat3;menor:real;var Laux6:unmatbat3);
var
I: integer;
begin
For I:= 1 to N do
if (J <> I) and (Laux5[I,J] > 0.0) then
Laux6[I,J]:= Laux5[I,J] - menor;
```

```
end;

Procedure ReducaoCol2(N,rr:integer;Laux5:unmatbat3; var vmenor_col:vetorbat3;var
Laux6:unmatbat3);
var
k1,I,J,Pos      :integer;
ind1,ind2       :char;
menor           :real;
begin
for I:= 1 to N do
vmenor_col[I]:= 0.;
for I:= 1 to N do
for J:= 1 to N do
laux6[I,J]:= Laux5[I,J];
for j:= 1 to N do
begin
ind1:= 'N';
ind2:= 'N';
for k1:= 1 to rr do
if j = Pares[k1,2] then
ind1:= 'S';
for i:= 1 to n do
if Laux5[i,j] = 0. then
ind2:= 'S';
if ind1 = 'S' then
if (ind2 = 'S')  then
for k1:= 1 to N do
laux6[k1,J]:= laux5[K1,J];
if ind1 = 'N' then
if (ind2 = 'N')  then
begin
menor:= 100000.;
for I:= 1 to N do
if I <> J then
if (Laux5[I,J] < menor) and (Laux5[I,J] <> -1) then
begin
pos:= i;
menor:= Laux5[I,J];
end;
vmenor_col[J]:=Laux5[Pos,J];
sub_Coluna2(J,N,Laux5,menor,Laux6);
end;
end;
end;
{Fim}

{Procedimento Escalonar Par Produto}
Procedure Escalonar_Produto(P,PosI,PosJ:integer;var Pares:matbat4);
var
```

```
I,J      : integer;
begin
Pares[P,1]:= PosI;
Pares[P,2]:= PosJ;
end;
{FIM}

{Procedimento Eliminacao de Par de Produto Escalonado}
Procedure Eliminar_Linha_Col(N,Pr,PosI,PosJ:integer;Laux3:unmatbat3;
var Laux4:unmatbat3);
var
k1,k2    : integer;
ind2     : char;
begin
For k1:= 1 to N do
For k2:= 1 to N do
Laux4[k1,k2]:= Laux3[k1,k2];
For k1:= 1 to N do
Laux4[PosI,K1]:= -1;
For k1:= 1 to N do
Laux4[k1,PosJ]:= -1;
if Laux3[PosJ,PosI] = 0 then
Laux4[PosJ,PosI]:= -1;
end;
{FIM}

Procedure montar_sequencia_final(N,P:integer;Pares:matbat4);
var
i,k1,pos,pprod     : integer;
begin
Pos:= 1;
sequencia[Pos]:= Pares[Pos,1];
sequencia[Pos+1]:= Pares[Pos,2];
pprod:= Pares[Pos,2];
pos:= succ(Pos);
for k1:= 1 to (P-1) do
for I:= 1 to P do
if pares[I,1] = pprod then
begin
pos:= succ(pos);
sequencia[pos]:= pares[i,2];
pprod:= pares[i,2];
i:= p;
end;
end;
```

```
{Procedimento Montar Sequencia}
Procedure Eliminar_Pares_Infactiveis(N,P:integer;Pares:matbat4);

var
I1,I2,k1,Posinf     : integer;
Posjinf,Posicao     : integer;
Pos,Pos1,Pprod      : integer;
seq                  : vetorbat61;
begin
For Pos1:= 1 to P do
begin
For I:= 1 to (N+1) do
seq[I]:= 0;
Pos:=1;
Seq[Pos]:= Pares[Pos1,1];
Seq[Pos+1]:= Pares[Pos1,2];
Pprod:= Pares[Pos1,2];
Pos:= succ(pos);
For k1:= 1 to (P-1) do
For I1:= 1 to P do
if Pares[I1,1] = Pprod then
begin
Pos:= succ(Pos);
seq[Pos]:= Pares[I1,2];
Pprod:= Pares[I1,2];
I1:= P;
end;
{eliminar pares infactiveis}
for i2:= 1 to N do
if  seq[i2] = 0 then
begin
Posicao:= i2-1;
i2:= N
end;
{Eliminar pares de Laux1}
posinf:= seq[posicao];
for i2:= 1 to (posicao-1) do
begin
posjinf:= seq[i2];
if Laux1[Posinf,Posjinf] <> -1 then
laux1[posinf,posjinf]:= -1;
end;
end; {fim de Pos1}
end;
{FIM}

{Montar matriz reducao }
Procedure ver_reducao1(r,n,pos:integer;mreducao:matbat5; var: posi:integer;var ind2:char);
var
```

```
j:          :integer; {pos indicara a posicao de inicio da procura}
begin
ind2:= 'S';
nposi:=0;
pos:=succ(pos);
for j:= pos to n do
if mreducao[r,j] <> 0 then
begin
nposi:= mreducao[r,j];
ind2:= 'N';
j:=n;
end;
end;
{Fim}

Procedure ver_posj(n,posi:integer;maior_rotulo:real;matzero:unmatbat3; pares:matbat4;var
nposj:integer;var ind3:char);
var
j,jj              :integer;
begin
nposj:=0;
for j:= 1 to n do
begin
ind3:='N';
if matzero[posi,j] = maior_rotulo then
begin
ind3:= 'S';
{verificar se este posj ja foi analisado}
nposj:= j;
for jj:=1 to n do
if nposj = mprodfac[posi,jj] then
begin
ind3:='N';
jj:= n;
end;
if ind3 = 'S' then      {existe nposj}
begin
mprodfac[posi,j]:= j;
j:= n;
end
else
begin
nposj:=0;
ind3:= 'N';
end;
end;        {fim do  if}
end;      {fim do 1 for}
end;   {fim da subrotina}
```

```
Procedure ver_matzero(n,posi,posj,nposi:integer;matzero:unmatbat3; var nposj:integer);
var
j:   integer;
maior_rotulo: real;
begin
if ((posi=0) and (posj=0)) then
begin
maior_rotulo:= -1;
for j:= 1 to n do
if matzero[nposi,j] > maior_rotulo then
begin
nposj:= j;
maior_rotulo:= matzero[nposi,j];
end;
mprodfac[nposi,nposj]:= nposj;
end
else
begin
maior_rotulo:= matzero[posi,posj];
for j:= 1 to n do
if matzero[nposi,j] = maior_rotulo then
begin
nposj:= j;
mprodfac[nposi,j]:= j;
j:=n;
end;
end;
end;

Procedure gerar_matzero(n,r:integer;clean:unmatbat3;pares:matbat4; var matzero:unmatbat3);
var
rr:      integer;
begin
bound:=0.;
rr:= 1;
Reducao_Linha1(n,clean,laux1,vmenor_lin);
ReducaoCol1(n,laux1,vmenor_col,laux1);
Calc_Bound(n,vmenor_lin,vmenor_Col);
Calc_MaiorZero(n,laux1,matzero);
nposi:= pares[rr,1];
nposj:= pares[rr,2];
Eliminar_Pares_Infactiveis(n,rr,pares);
Eliminar_Linha_Col(n,pares[1,1],nPosI,nposj,laux1,laux1);
for rr:= 2 to r do
begin
reducaoLinha2(n,rr-1,laux1,vmenor_lin,laux1);
reducaoCol2(n,rr-1,laux1,vmenor_col,laux1);
calc_Bound(N,vmenor_lin,vmenor_Col);
calc_MaiorZero(n,laux1,matzero);
```

```
nposi:= pares[rr,1];
nposj:= pares[rr,2];
eliminar_Pares_Infactiveis(n,rr,pares);
eliminar_Linha_Col(n,pares[1,1],nPosI,nposj,laux1,laux1);
end;
end;



Procedure cor_texto_fundo(cor1,cor2:byte);
begin
textcolor(cor1);
textbackground(cor2);
clrscr;
end;

Procedure Imprimir_sequencia;
var
i  : integer;
begin
montar_Sequencia_Final(n,n,pares);
{Calculo do Custo da Sequencia}
{Calculo_Custo(n,clean,sequencia,custo);}
write(lst,' ':8,' Sequencia: ');
for i:= 1 to (n+1) do
write(lst,sequencia[i],'-');writeln(":10);
writeln(lst,":9,'custo = ',bound:4:1,":15);
end;
{Fim dos Procedimentos}

BEGIN                {Inicio Batjs2};
clrscr;
writeln;
cor1:= 5; cor2:=15;
cor_texto_fundo(cor1,cor2);
{lendo recurso computacional}
write('Recurso computacional: ');
readln(rc);
{Lendo o Numero de Produtos}
write('Nome do arquivo de tempos de estabelecimento: ');
read(l);
learquivo(clean,l,n);
{inicializando variaveis}
clrscr;
for i:= 1 to bat16 do
for j:= 1 to bat16 do
begin
laux1[I,J]:=0.;
matzero[i,j]:=0;
```

```
end;
for i:= 1 to bat16 do
for j:= 1 to bat16 do
begin
mreducao[i,j]:=0;
mprodfac[i,j]:=0;
end;
for i:= 1 to (bat16+1) do
begin
sequencia[I]:=0;
seqotima[i]:= 0;
end;
for i:=1 to bat16 do
for j:= 1 to 2 do
pares[i,j]:= 0;
writeln(lst);
writeln(lst,':4,' Tempos de Estabelecimento');writeln;
writeln(lst,':3);
for i:= 1 to n Do
begin
for j:= 1 to n Do
write(lst,clean[i,j]:3:1,' ');
writeln(lst)
end;
writeln(lst);
{** setando tempo de inicio}
gettime(hora1,minuto1,segundo1,seg1);
writeln('Programa em execucao, aguarde...');
writeln(lst,'Sequencias analisadas');
writeln(lst);
{montar sequencia tentativa}
for i:= 1 to n do
sequencia[i]:= i;
sequencia[n+1]:= sequencia[1];
for i:= 1 to n+1 do
seqotima[i]:= sequencia[i];
calculo_custo(n,clean,sequencia,custo);
bound:=0;
bound_inicial:= custo;
ind1:= 'N';  {indicara se no 1 na primeira reducao existe ainda tarefas}
ind2:= 'N';  {a serem avaliadas}
ind3:= 'N';
nposi:=0;
nposj:=0;
r:=1;
while ind1 <> 'S' do
begin
bound:=0.;          {colocar ind6 = N}
reducao_linha1(n,clean,laux1,vmenor_lin);
```

```
reducaoCol1(n,laux1,vmenor_col,laux1);
calc_bound(n,vmenor_lin,vmenor_col);
calc_maiorzero(n,laux1,matzero);
calc_maior_elementomatzero(n,matzero,posi,posj);
{montar matriz reducao}
montar_mreducao(r,n,posi,posj,matzero);
pos:= pares[r,1];
{procurar em matzero na linha posi novo posj, tomar posi:= pares[r,1]}
if pos <> 0 then
begin
{ver na linha pares[r,1] mat. posj diferente posj existente em mprodfac}
posi:= pares[r,1];
posj:= pares[r,2];
maior_rotulo:= matzero[posi,posj];
ver_posj(n,posi,maior_rotulo,matzero,pares,nposj,ind3);
end;
if ind3 = 'N' then    {nao exite mais posj}
begin
ver_reducao1(r,n,pos,mreducao,nposi,ind2);
posi:= pares[r,1];
posj:= pares[r,2];
ver_matzero(n,posi,posj,nposi,matzero,nposj);
{antes de escalonar as novas tarefas zerar linha de mprodfac de posi}
for j:=1 to n do
mprodfac[pares[r,1],j]:=0;
{escalonar nposi e nposj}
pares[r,1]:= nposi;
pares[r,2]:= nposj;
end
else
begin
nposi:=pares[r,1];
pares[r,2]:=nposj;
end;
eliminar_Pares_Infactiveis(n,r,pares);
eliminar_Linha_Col(n,pares[1,1],nposI,nposJ,laux1,laux1);
for i:= 2 to n do
for j:= 1 to n do
mreducao[i,j]:=0;
for I:= 2 to N do
for J:= 1 to 2 do
pares[I,J]:= 0;
if ind2 = 'S' then   {nao existe mais tarefa em reducao 1}
begin
ind1:= 'S'
end
else
begin
r:= succ(r);
```

```
ind6:= 'N';
ind7:='S';
while r <> 1 do
begin
if (r < n+1) and (ind6 <> 'S') then
begin
reducaoLinha2(n,r,laux1,vmenor_lin,laux1);
reducaoCol2(n,r,laux1,vmenor_col,laux1);
calc_Bound(n,vmenor_lin,vmenor_col);
if bound >  bound_inicial then
begin
r:= pred(r);
pos:=  pares[r,1];
posi:= pares[r,1];
posj:= pares[r,2];
pares[r,1]:=0;
pares[r,2]:=0;
gerar_matzero (n,r-1,clean,pares,matzero);
reducaoLinha2(n,r-1,laux1,vmenor_lin,laux1);
reducaoCol2(n,r-1,laux1,vmenor_col,laux1);
calc_MaiorZero(n,laux1,matzero);
ind7:='N';
end;
if ind7 <> 'N' then
begin
calc_MaiorZero(n,laux1,matzero);
calc_Maior_ElementoMatzero(n,matzero,posi,posj);
{montar matriz reducao}
montar_mreducao(r,n,posi,posj,matzero);  {seta mreducao com os nposi}
end;
end;
if r > n then
begin
{montar e imprimir sequencia}
imprimir_sequencia;
{atualizar sequencia e menor limitante}
if bound < bound_inicial then
begin
bound_inicial:= bound;
for i:= 1 to n+1 do
seqotima[i]:= sequencia[i];
end;
ind6:= 'S';       {indicador retrocesso na arvore binaria}
r:= n-2;          {reducao ate r = n-2}
for i:= (r+1) to n do
for j:= 1 to n do
mreducao[i,j]:=0;
{limpar mprodfac}
zer:= r+1;
```

```
for i:= zer to n do
for j:= 1 to n do
mprodfac[pares[i,1],j]:=0;
for i:= (r+1) to n do
for j:= 1 to 2 do
pares[I,J]:= 0;
gerar_matzero (n,r,clean,pares,matzero);  {ja deixa laux1 pronta}
end;   {para calcular matzero}
if ind7 <> 'N' then      {indica se o bound foi maior}
begin                    {pos,posi e posj ja foram guardados}
pos:= pares[r,1];
ind7:= 'S';
end;
ind4:='N';
{procurar em matzero na linha posi novo posj, tomar posi:= pares[r,1]}
if pos <> 0 then
begin
{ver na linha pares[r,1] matzero posj diferente posj exit. em mprodfac}
if ind7 <> 'N' then
begin
posi:= pos;
posj:= pares[r,2];
end;
maior_rotulo:= matzero[posi,posj];
ver_posj(n,posi,maior_rotulo,matzero,pares,nposj,ind4);
end;
if ind4 = 'N' then   {nao exite mais posj}
begin
ver_reducao1(r,n,pos,mreducao,nposi,ind5);if ind5 = 'N' then
begin
posi:= pares[r,1];
posj:= pares[r,2];
ind7:= 'S';
ver_matzero(n,posi,posj,nposi,matzero,nposj);
pos <> 0 then  {limpar linha nposi anterior}
for j:=1 to n do
mprodfac[pos,j]:=0;
{escalonar nposi e nposj}
pares[r,1]:= nposi;
pares[r,2]:= nposj;
if ind6 = 'S' then   {se houve retrocesso}
begin
{chamar gerar matzero houve modificacao em pares}
gerar_matzero(n,r,clean,pares,matzero);
ind6:= 'N';
end;
if r < n-1 then
eliminar_Pares_Infactiveis(n,r,pares);
eliminar_Linha_Col(n,pares[1,1],nPosI,nposj,laux1,laux1);
```

```
r:=succ(r);
end   {fim ind5}
else
begin
ind6:= 'S';
{limpar mprodfac}
for i:= r to zer do
for j:= 1 to n do
mprodfac[pares[i,1],j]:=0;
for i:= r to n do
for j:= 1 to n do
mreducao[i,j]:=0;
for i:= r to n do
for j:= 1 to 2 do
pares[i,j]:= 0;
r:= pred(r);
gerar_matzero(n,r,clean,pares,matzero);
ind7:= 'S';
end;     {fim do else}
end  {fim ind4}
else
begin
ind6:= 'N';
if ind7 = 'N' then
begin
nposi:= posi;
ind7:= 'S';
end;
pares[r,1]:=nposi;
pares[r,2]:=nposj;
gerar_matzero(n,r,clean,pares,matzero);
r:= succ(r);
end;          {fim do else}
end;          {fim 2 while}
end;          {fim do else}
end;          {fim do 1 while}
gettime(hora2,minuto2,segundo2,seg2);
tempo:= ((hora2-hora1)*3600) + ((minuto2-minuto1)*60) + (segundo2 -segundo1);
writeln(lst);
writeln(lst);
writeln(lst,'Recurso computacional: ',rc);
writeln(lst,'Linguagem de programacao: Turbo Pascal');
writeln(lst,'Tempo de execucao: ',tempo,' ','segundos');
writeln(lst);
write(lst,'sequencia otima: ');
for i:= 1 to n+1 do
write(lst,seqotima[i],'-'); writeln(lst);
writeln(lst,'custo associado = ',bound_inicial:3:1,' u.t.');
writeln(lst);
```

```
writeln('fim do programa');
writeln('TECLE ESPACO P/ CONTINUAR');
fixatela;
End.            {fim Batjs2}
```

Unit unBatch6;
Interface

Tese de Mestrado

FACULDADE DE ENGENHARIA QUÍMICA

Departamento de Engenharia de Sistemas Químicos - D.E.S.Q.

Descricao do Programa
Programa que estabelece o custo inicial da solucao tentativa

Descrição da variáveis principais

```
n           - numero de tarefas a serem programadas
m           - numero de processadores presentes na planta batelada
sequencia   - lista das tarefas programadas
custo       - custo da sequencia}
uses crt;
const
bat16 = 4;
type
vetorbat61  = Array[1..(bat16+1)] of integer;
unmatbat3  = Array[1..bat16,1..bat16] of real;
var
n,k,i,j                         : integer;
custo                           : real;
clean                           : unmatbat3;
Procedure calculo_custo(n:integer;clean:unmatbat3;sequencia:vetorbat61; var custo: real);

Implementation
Procedure calculo_custo(n:integer;clean:unmatbat3;sequencia:vetorbat61; var custo: real);

Begin           {unbatch6}
custo:=0;
for k:= 1 to n do
begin
i:= sequencia[k];
j:= sequencia[k+1];
custo:= custo + clean[i,j];
end;
end;
end.            {fim unbatch6}
```

Program Batjs7;

Tese de Mestrado

FACULDADE DE ENGENHARIA QUÍMICA

Departamento de Engenharia de Sistemas Químicos - D.E.S.Q.-UNICAMP

Descrição do Programa

Algoritmo C - Seqüenciamento de tarefas com tempos de estabelecimento dependentes da Seqüência de Produção e independentes do processador

```
uses unsetup,unbatch6,unbatch8,crt,printer,fixa;

{Definição de tipos}
type
srt12 = string[15];

Definição de variáveis}
var
n,m,ij7,iij7,iiij7,jj7,nseq      : integer;
tc1,tc2                          : real;
l,t                              : srt12;
time                             : matbat81;
clean                            : unmatbat3;
pares                            : matbat4;
cor1,cor2                        : integer;

{Inicio de Procedimentos}

{Leitura do arquivo de tempos de processamento}
Procedure learquivo1(var tt :matbat81;nome: str12; var n :integer);
var
f          : text;
ch         : char;
ncs        : string[2];
i,j,k,e    : integer;
s          : array[1..bat16,1..bat16] of string[8];
begin
assign(f,nome);
reset(f);
for i := 1 to bat16 do
for j := 1 to bat16 do
s[i,j] := '';
i := 0;
j := 1;
k := 0;
ncs:='';
while not eof(f) do
begin
read(f,ch);
```

```
if (k <> 2) and (ch <> #13) then
begin
k:= k + 1;
ncs:= ncs+ch;
val(ncs,n,e);
end
else
if ch=#32 then
j:= j+1
else
if ch=#13 then
begin
i := i+1;
j := 1;
k:= 2;
end
else
if ch<>#10 then
s[i,j] := s[i,j]+ch;
end;
for i := 1  to n do
for j := 1 to n do
val(s[i,j],tt[i,j],e);
close(f);
end;
{Fim}
{Leitura do arquivo de tempos de estabelecimento}

Procedure learquivo2(var tt :unmatbat3;nome: str12; var n :integer);
var
f              : text;
ch             : char;
ncs            : string[2];
i,j,k,e        : integer;
s : array[1..bat16,1..bat16] of string[8];
begin
assign(f,nome);
reset(f);
for i := 1 to bat16 do
for j := 1 to bat16 do
s[i,j] := '';
i := 0;
j := 1;
k := 0;
ncs:='';
while not eof(f) do
begin
read(f,ch);
if (k <> 2) and (ch <> #13) then
begin
k:= k + 1;
ncs:= ncs+ch;
```

```
val(ncs,n,e);
end
else
if ch=#32 then
j:= j+1
else
if ch=#13 then
begin
i := i+1;
j := 1;
k:= 2;
end
else
if ch<>#10 then
s[i,j] := s[i,j]+ch;
end;
for i := 1  to n do
for j := 1 to n do
val(s[i,j],tt[i,j],e);
close(f);
end;
{Fim}
Procedure cor_texto_fundo(cor1,cor2:byte);
begin
textcolor(cor1);
textbackground(cor2);
clrscr;
end;

Begin            {Inicio Batjs7}
clrscr;
writeln;
cor1:=5; cor2:=15;
cor_texto_fundo(cor1,cor2);
write('Entre com o numero de maquinas = ');
readln(m);
writeln;
write('Entre com o arquivo tempos de processamento = ');
readln(t);
learquivo1(time,t,n);
write('Entre com o arquivo tempos de estabelecimento = ');
readln(l2);
learquivo2(clean,l2,n);

{Inicializacao das variaveis}
for i:= 1 to n do
for j:= 1 to 2 do
pares[i,j]:= 0;
nseq:= 0;
{Impressao dos Dados}
writeln(lst,'Tempos de Processamento');
writeln(lst);
```

```
for i:= 1 to n do
begin
for j:= 1 to m do
write(lst,time[i,j]:3:1,' ');
writeln(lst);
end;
writeln(lst);
writeln(lst,'Tempos de Estabelccimento');
writeln(lst);
for i:= 1 to n do
begin
for j:= 1 to n do
write(lst,clean[i,j]:3:1,' ');
writeln(lst);
end;
tc1:=1000;
{Montar suconjuntos}
for ij7:= 1 to n do
begin
writeln(lst);
writeln(lst,' subgrupo',ij7);
writeln(lst);
for iij7:= 1 to n do
if ij7 <> iij7 then
begin
pares[1,1]:= ij7;
pares[1,2]:= iij7;
for iiij7:= 1 to n do
begin
if (iiij7 <> ij7) and (iiij7<> iij7) then
begin
pares[2,1]:= iiij7;
pares[2,2]:= ij7;
{chamar procedimento de minimizacao}
min_setup(n,m,nseq,pares,clean,time,tc1,seqotima,tc2,nseq);
tc1:=tc2;
end;
end;
end;
writeln(lst);
write(lst,' ':4,' Sequencia Otima: ');
for iiij7:= 1 to (n+1) do
write(lst,seqotima[iiij7],'-');write('':4);
writeln(lst,'tc = ',tc2:4:1);
writeln(lst);
writeln(lst,nseq);
fixatela;
tc1:=1000;
end;
End.     {fim Batjs7}
```

```
Program Batjs6;

                    Tese de Mestrado

              FACULDADE DE ENGENHARIA QUÍMICA

        Departamento de Engenharia de Sistemas Químicos - D.E.S.Q.-UNICAMP

Descrição do Programa

Esta unidade minimiza os tempos de estabelecimento, gerando seqüências candidatas
à otimização do tempo total de execução.

Dada a semelhança deste problema com o problema clássico da otimização, problema
do caxeiro viajante, é utilizada aqui a técnica "Branch and Bound" para a
minimização do tempo total de estabelecimeto}

{Descrição das variáveis}

r                   - indicador de redução
nposi e nposj       - par a ser escalonado
laux1               - matriz reduzida, geradora de matzero
matzero             - matriz de rótulos maiores
bound_inicial       - custo ótimo da melhor sequência encontrada
bound               - custo das sequências parciais e completas
custo               - custo avaliado pela unidade unbatch6 (e uma unidade inativa)

uses unbatch6,fixa,crt;

{Definição de tipos}
type
str12       = string[15];
vetorbat3   = Array[1..bat16] of real;
matbat4     = array[1..bat16,1..2] of integer;
matbat5     = array[1..bat16,1..bat16] of integer;

{Definição de variáveis}
var
r,zer,n,p,i,j,pos,nposi,nposj,posi,posj,fac     : integer;
ind1,ind2,ind3,ind4,ind5,ind6                   : char;
maior_rotulo,bound_inicial,bound,custo          : real;
vmenor_lin,vmenor_Col                           : vetorbat3;
clean,laux1,matzero                             : unmatbat3;
mreducao,mprodfac                               : matbat5;
pares                                           : matbat4;
sequencia,seqotima                              : vetorbat61;
l,l1,l2                                         : str12;
cor1,cor2                                       : byte;
```

```
{Inicio dos Procedimentos}

{Leitura do arquivo de tempos de estabelecimentos
Procedure learquivo(var tt :unmatbat3;nome: str12; var n :integer);
var
f            : text;
ch           : char;
ncs          : string[2];
i,j,k,e      : integer;
s            : array[1..bat16,1..bat16] of string[8];
begin
assign(f,nome);
reset(f);
for i := 1 to bat16 do
for j := 1 to bat16 do
s[i,j] := '';
i := 0;
j := 1;
k := 0;
ncs:='';
while not eof(f) do
begin
read(f,ch);
if (k <> 2) and (ch <> #13) then
begin
k:= k + 1;
ncs:= ncs+ch;
val(ncs,n,e);
end
else
if ch=#32 then
j:= j+1
else
if ch=#13 then
begin
i := i+1;
j := 1;
k:= 2;
end
else
if ch<>#10 then
s[i,j] := s[i,j]+ch;
end;
for i := 1  to n do
for j := 1 to n do
val(s[i,j],tt[i,j],e);
close(f);
 end;
{Fim}

{Procedimento Reducao-Linha inicial da matriz L}
```

```
Procedure Sub_Linha(I,N:integer;L:unmatbat3;menor:real; var Laux2:unmatbat3);
var
j: integer;
begin
for j:= 1 to N do
if (j <> i) and (l[i,j] > 0.0) then
laux2[i,j]:= L[i,j] - menor;
end;

Procedure Reducao_Linha1(N:integer;L:unmatbat3;Var Laux1:unmatbat3; var
vmenor_Lin:vetorbat3);
var
i1,j1       :Integer;
menor       :real;
begin
for i1:= 1 to n do
vmenor_lin[i1]:= 0.;
for i1:= 1 to n do
for j1:= 1 to n do
If i1 = j1 then
Laux1[i1,j1]:= -1.;
for i1:= 1 to n Do
begin
menor:= 1000.;
for j1:= 1 to N Do
If (j1<>i1) and (l[i1,j1] > 0. ) then
if l[i1,j1] <= menor then
begin
menor:= l[i1,j1];
vmenor_Lin[I1]:=L[i1,j1];
end;
Sub_Linha(I1,N,L,menor,Laux1)
end;
end;
{Fim}

{Procedimento Reducao-Coluna inicial da matriz L}

Procedure Sub_Coluna1(J,N:integer;Laux1:unmatbat3;menor:real; var vaux3:vetorbat3);
var
K1          : integer;
begin
For k1:= 1 to N do
if (k1 <> J) and (clean[k1,J] >= 0.0) then
vaux3[k1]:= Laux1[k1,J] - menor
else
vaux3[k1]:= -1;
end;
Procedure ReducaoCol1(N:integer;Laux1:unmatbat3;var vmenor_Col:vetorbat3; var
Laux3:unmatbat3);
var
I1,J1,Pos,K1,k2     : integer;
```

```
ind1            : char;
menor           : real;
vaux3           : vetorbat3;
begin
For I1:= 1 to N do
vmenor_col[I1]:= 0.;
For J1:= 1 to N do
begin
ind1:= 'N';
For I1:= 1 to N do
if Laux1[I1,J1] = 0. then
ind1:= 'S';
if ind1 = 'S' then
For K1:=1 to N do
Laux3[K1,J1]:= Laux1[K1,J1];
if ind1 = 'N' then
begin
menor:=1000.;
For K1:= 1 to N Do
If (J1<>K1) then
if Laux1[k1,J1] <= menor then
begin
Pos:=k1;
menor:= Laux1[k1,J1];
end;
vmenor_Col[J1]:=Laux1[Pos,J1];
Sub_Coluna1(J1,N,Laux1,menor,vaux3);
For K2:=1 to N do
Laux3[k2,J1]:= vaux3[k2];
end;
end;
ed;
{Fim}

{Procedimento Calculo do Bound}

Procedure Calc_Bound(N:integer;vmenor_lin,vmenor_col:vetorbat3);
var
I,J       :integer;
begin
For I:= 1 to N do
Bound:= Bound + vmenor_lin[I];
For J:= 1 to N do
Bound:= Bound + vmenor_col[J];
end;
{Fim}

{Procedimeto Busca Menor Linha-Coluna}

Procedure B_MenorLinCol(I,J,N:integer;Laux3:unmatbat3;var Matzero:unmatbat3);
var
K1              :integer;
```

```
menor_linha,menor_coluna  :real;
begin
menor_linha:= 1000.;
For K1:= 1 to N do
if (Laux3[I,K1] >= 0.0) and (k1 <> J) then
if Laux3[I,K1] < menor_linha then
menor_linha:= Laux3[I,K1];
if menor_linha = 1000. then
menor_linha:= 0.;
menor_coluna:= 1000.;
For K1:= 1 to N do
if (Laux3[K1,J] >= 0.0) and (k1 <> I) then
if Laux3[K1,J] < menor_coluna then
menor_coluna:= Laux3[K1,J];
if menor_coluna = 1000.0 then
menor_coluna:= 0.;
Matzero[I,J]:= menor_linha + menor_coluna;
end;
{Fim}

{Procedimento Calculo da Matriz de pares de produtos factiveis}
Procedure Calc_MaiorZero(N:integer;Laux3:unmatbat3;var Matzero:unmatbat3);
var
I,J       :integer;
begin
For I:=1 to N do
For J:=1 to N do
Matzero[I,J]:= -1;
For I:=1 to N do
For J:=1 to N do
if Laux3[I,J] = 0 then
B_MenorLinCol(I,J,N,Laux3,Matzero);
end;
{Fim}

{Procedimeto Busca maior par de Produto factivel}
Procedure Calc_Maior_ElementoMatzero(N:integer;Matzero:unmatbat3; var
PosI,PosJ:integer);
var
I,J    : integer;          {calcula o maior elemento de}
maior   :real;              { matzero Matzero[pos1,posj] tem}
begin                      { o rotulo maior}
maior:= -1;
For I:=1 to N do
For J:= 1 to N do
if Matzero[I,J] <> -1 then
if Matzero[I,J] > maior then
begin
maior:= Matzero[I,J];
PosI:= I;
PosJ:= J;
end;
```

```
end;
{Fim}

Procedure montar_mreducao(r,n,posi,posj:integer;matzero:unmatbat3);
var
i,j              : integer;
maior_rotulo  : real;
begin
maior_rotulo:= matzero[posi,posj];
for i:= 1 to n do
for j:= 1 to n do
if matzero[i,j] = maior_rotulo then
begin
mreducao[r,i]:= i;
j:= n;
end;
end;
{Fim ***}

{Procedimento Reducao Linha2}

Procedure ReducaoLinha2(N,rr: integer;Laux4:unmatbat3; var vmenor_Linha:vetorbat3;var
Laux5:unmatbat3);
var
k1,I,J,Pos      :integer;
ind1,ind2       :char;
menor           :real;
begin
For I:= 1 to N do
For J:= 1 to N do
Laux5[I,J]:= Laux4[I,J];
For I:= 1 to N do
vmenor_Linha[I]:= 0.;
For I:= 1 to N do
begin
ind2:= 'N';
ind1:= 'N';
For k1:= 1 to rr do
if I = Pares[k1,1] then
ind1:= 'S';
For k1:= 1 to N do
if Laux4[I,k1] = 0 then
begin
ind2:= 'S';
k1:= N;
end;
if ind1 = 'S' then
if (ind2 = 'S')  then
For k1:= 1 to N do
Laux5[I,K1]:= Laux4[I,K1];
if ind1 = 'N' then
if (ind2 = 'N') then
```

```
begin
menor:= 1000.;
For J:= 1 to N do
begin
if J <> I then
begin
if (Laux4[I,J] < menor) and (Laux4[I,J] <> -1) and (Laux4[I,J] <> 0) then
begin
Pos:= J;
menor:= Laux4[I,J];
end;
vmenor_Linha[I]:=Laux4[I,Pos];
 end;
sub_Linha(I,N,Laux4,menor,Laux5);
end;
end;
end;
end;
{Fim}


{Procedimento Reducao de Coluna2}
Procedure Sub_Coluna2(J,N:integer;Laux5:unmatbat3;menor:real; var Laux6:unmatbat3);
var
I:      integer;
begin
For I:= 1 to N do
if (J <> I) and (Laux5[I,J] > 0.0) then
Laux6[I,J]:= Laux5[I,J] - menor;
end;

Procedure ReducaoCol2(N,rr:integer;Laux5:unmatbat3; var vmenor_col:vetorbat3;var
Laux6:unmatbat3);
var
k1,I,J,Pos      :integer;
ind1,ind2       :char;
menor           :real;
begin
For I:= 1 to N do
vmenor_col[I]:= 0.;
For I:= 1 to N do
For J:= 1 to N do
Laux6[I,J]:= Laux5[I,J];
for j:= 1 to N do
begin
ind1:= 'N';
ind2:= 'N';
for k1:= 1 to rr do
if j = Pares[k1,2] then
ind1:= 'S';
for i:= 1 to n do
if Laux5[i,j] = 0. then
```

```
ind2:= 'S';

if ind1 = 'S' then
if (ind2 = 'S')  then
For k1:= 1 to N do
Laux6[k1,J]:= Laux5[K1,J];
if ind1 = 'N' then
if (ind2 = 'N')  then
begin
menor:= 1000.;
For I:= 1 to N do
if I <> J then
if (Laux5[I,J] < menor) and (Laux5[I,J] <> -1) then
begin
Pos:= I;
menor:= Laux5[I,J];
end;
vmenor_col[J]:=Laux5[Pos,J];
sub_Coluna2(J,N,Laux5,menor,Laux6);
end;
end;
end;
{Fim}

{Procedimento Escalonar Par Produto}
Procedure Escalonar_Produto(P,PosI,PosJ:integer;var Pares:matbat4);
var
I,J     : integer;
begin
Pares[P,1]:= PosI;
Pares[P,2]:= PosJ;
end;
{FIM}

{Procedimento Eliminacao de Par de Produto Escalonado}
Procedure Eliminar_Linha_Col(N,Pr,PosI,PosJ:integer;Laux3:unmatbat3; var
Laux4:unmatbat3);
var
k1,k2    : integer;
ind2     : char;
begin
For k1:= 1 to N do
For k2:= 1 to N do
Laux4[k1,k2]:= Laux3[k1,k2];
For k1:= 1 to N do
Laux4[PosI,K1]:= -1;
For k1:= 1 to N do
Laux4[k1,PosJ]:= -1;
if Laux3[PosJ,PosI] = 0 then
Laux4[PosJ,PosI]:= -1;
end;
{FIM}
```

```
Procedure montar_sequencia_final(N,P:integer;Pares:matbat4);
 var
I,k1,Pos,Pprod     : integer;
begin
Pos:= 1;
Sequencia[Pos]:= Pares[Pos,1];
Sequencia[Pos+1]:= Pares[Pos,2];
Pprod:= Pares[Pos,2];
Pos:= succ(Pos);
for k1:= 1 to (P-1) do
for I:= 1 to P do
if Pares[I,1] = Pprod then
begin
Pos:= succ(Pos);
sequencia[Pos]:= Pares[I,2];
Pprod:= Pares[I,2];
I:= P;
end;
end;

{Procedimento Montar Sequencia}
Procedure Eliminar_Pares_Infactiveis(N,P:integer;Pares:matbat4);
var
I1,I2,k1,Posinf   : integer;
Posjinf,Posicao   : integer;
Pos,Pos1,Pprod    : integer;
seq               :vetorbat61;
begin
for Pos1:= 1 to P do
begin
For I:= 1 to (N+1) do
seq[I]:= 0;
Pos:=1;
Seq[Pos]:= Pares[Pos1,1];
Seq[Pos+1]:= Pares[Pos1,2];
Pprod:= Pares[Pos1,2];
Pos:= succ(pos);
For k1:= 1 to (P-1) do
For I1:= 1 to P do
if Pares[I1,1] = Pprod then
begin
Pos:= succ(Pos);
seq[Pos]:= Pares[I1,2];
Pprod:= Pares[I1,2];
I1:= P;
end;
{eliminar pares infactiveis}
for i2:= 1 to N do
if  seq[i2] = 0 then
begin
Posicao:= i2-1;
```

```
i2:= N
end;
{Eliminar pares de Laux1}
posinf:= seq[posicao];
for i2:= 1 to (posicao-1) do
begin
posjinf:= seq[i2];
if Laux1[Posinf,Posjinf] <> -1 then
laux1[posinf,posjinf]:= -1;
end;
end; {fim de Pos1}
end;
{FIM}

{Montar matriz reducao}
Procedure ver_reducao1(r,n,pos:integer;mreducao:matbat5;var nposi:integer; var ind2:char);
var
j: integer;              {pos indicara a posicao de inicio da procura}
begin
ind2:= 'S';
nposi:=0;
pos:=succ(pos);
for j:= pos to n do
if mreducao[r,j] <> 0 then
begin
nposi:= mreducao[r,j];
ind2:= 'N';
j:=n;
end;
end;
{Fim}

Procedure ver_posj(n,posi:integer;maior_rotulo:real;matzero:unmatbat3; pares:matbat4;var
nposj:integer;var ind3:char);
var
j,jj          :integer;
begin
nposj:=0;
for j:= 1 to n do
begin
ind3:='N';
if matzero[posi,j] = maior_rotulo then
begin
ind3:= 'S';
{verificar se este posj ja foi analisado}
nposj:= j;
for jj:=1 to n do
if nposj = mprodfac[posi,jj] then
begin
ind3:='N';
jj:= n;
end;
```

```
if ind3 = 'S' then     {existe nposj}
begin
mprodfac[posi,j]:= j;
j:= n;
end
else
begin
nposj:=0;
ind3:= 'N';
end;
end;       {fim do  if}
end;      {fim do 1 for}
end;   {fim da subrotina}

Procedure ver_matzero(n,posi,posj,nposi:integer;matzero:unmatbat3; var nposj:integer);
var
j:   integer;
maior_rotulo: real;
begin
if ((posi=0) and (posj=0)) then    {fazer este teste antes}
begin
maior_rotulo:= -1;
for j:= 1 to n do
if matzero[nposi,j] > maior_rotulo then
begin
nposj:= j;
maior_rotulo:= matzero[nposi,j];
end;
mprodfac[nposi,nposj]:= nposj;
end
else
begin
maior_rotulo:= matzero[posi,posj];
for j:= 1 to n do
if matzero[nposi,j] = maior_rotulo then
begin
nposj:= j;
mprodfac[nposi,j]:= j;
j:=n;
end;
end;
end;

Procedure gerar_matzero(n,r:integer;clean:unmatbat3;pares:matbat4; var matzero:unmatbat3);
var
rr:      integer;
begin
bound:=0.;
rr:= 1;
Reducao_Linha1(n,clean,laux1,vmenor_lin);
ReducaoCol1(n,laux1,vmenor_col,laux1);
Calc_Bound(n,vmenor_lin,vmenor_Col);
```

```
Calc_MaiorZero(n,laux1,matzero);
nposi:= pares[rr,1];
nposj:= pares[rr,2];
Eliminar_Pares_Infactiveis(n,rr,pares);
Eliminar_Linha_Col(n,pares[1,1],nPosI,nposj,laux1,laux1);
for rr:= 2 to r do
begin
reducaoLinha2(n,rr-1,laux1,vmenor_lin,laux1);
reducaoCol2(n,rr-1,laux1,vmenor_col,laux1);
calc_Bound(N,vmenor_lin,vmenor_Col);
calc_MaiorZero(n,laux1,matzero);
nposi:= pares[rr,1];
nposj:= pares[rr,2];
eliminar_Pares_Infactiveis(n,rr,pares);
eliminar_Linha_Col(n,pares[1,1],nPosI,nposj,laux1,laux1);
end;
end;

Procedure cor_texto_fundo(cor1,cor2:byte);
begin
textcolor(cor1);
textbackground(cor2);
clrscr;
end;

Procedure Imprimir_sequencia;
var
i : integer;
begin
montar_Sequencia_Final(n,n,pares);
{Calculo do Custo da Sequencia}
{Calculo_Custo(n,clean,sequencia,custo);}    {*** procedimento inativo }
write(' ':8,' Sequencia Otima: ');
for i:= 1 to (n+1) do
write(sequencia[i],'-');writeln(':10);
writeln(':9,'custo da sequencia= ',bound:4:1,':15);
end;
{Fim dos Procedimentos}


BEGIN        {Inicio Batjs2};
clrscr;
writeln;
cor1:= 5; cor2:=15;
cor_texto_fundo(cor1,cor2);
{Lendo o Numero de Produtos}
write('Nome do arquivo de tempos de estabelecimento: ');
read(l1);
learquivo(clean,l1,n);
{inicializar variaveis}
clrscr;
for i:= 1 to bat16 do
```

```
for j:= 1 to bat16 do
begin
laux1[I,J]:=0.;
matzero[i,j]:=0;
end;
for i:= 1 to bat16 do
for j:= 1 to bat16 do
begin
mreducao[i,j]:=0;
mprodfac[i,j]:=0;
end;
for i:= 1 to (bat16+1) do
begin
sequencia[I]:=0;
seqotima[i]:= 0;
end;
for i:=1 to bat16 do
for j:= 1 to 2 do
pares[i,j]:= 0;
writeln;
writeln(':4,' Tempos de Estabelecimento');writeln;
writeln;
for I:= 1 to n Do
begin
for J:= 1 to n Do
write(clean[i,j]:3:1,' ');
writeln;
end;
writeln;
ind1:= 'N';  {indicara se no 1 na primeira reducao existe ainda tarefas}
ind2:= 'N';  {a serem avaliadas}
ind3:= 'N';
bound:=0;
bound_inicial:= 100; {utilizar uma heuristica para o calcula do B inicial}
nposi:=0;
nposj:=0;
r:=1;
while ind1 <> 'S' do
begin
bound:=0.;          {colocar ind6 = N}
Reducao_Linha1(N,clean,Laux1,vmenor_lin);
ReducaoCol1(N,Laux1,vmenor_Col,Laux1);
Calc_Bound(N,vmenor_lin,vmenor_Col);
Calc_MaiorZero(N,Laux1,Matzero);
Calc_Maior_ElementoMatzero(n,matzero,posI,posJ);
{montar matriz reducao}
montar_mreducao(r,n,posi,posj,matzero);
pos:= pares[r,1];
{procurar em matzero na linha posi novo posj, tomar posi:= pares[r,1]}
if pos <> 0 then
begin
{ver na linha pares[r,1] mat. posj diferente posj existente em mprodfac}
```

```
posi:= pares[r,1];
posj:= pares[r,2];
maior_rotulo:= matzero[posi,posj];
ver_posj(n,posi,maior_rotulo,matzero,pares,nposj,ind3);
end;
if ind3 = 'N' then    {nao exite mais posj}
begin
ver_reducao1(r,n,pos,mreducao,nposi,ind2);
posi:= pares[r,1];
posj:= pares[r,2];
ver_matzero(n,posi,posj,nposi,matzero,nposj);
{antes de escalonar as novas tarefas zerar linha de mprodfac de posi}
for j:=1 to n do
mprodfac[pares[r,1],j]:=0;
{escalonar nposi e nposj}
pares[r,1]:= nposi;
pares[r,2]:= nposj;
end
else
begin
nposi:=pares[r,1];
pares[r,2]:=nposj;
end;
Eliminar_Pares_Infactiveis(n,r,pares);
Eliminar_Linha_Col(n,pares[1,1],nposI,nposJ,laux1,laux1);
for i:= 2 to n do
for j:= 1 to n do
mreducao[i,j]:=0;
for I:= 2 to N do
for J:= 1 to 2 do
pares[I,J]:= 0;
if ind2 = 'S' then   {nao existe mais tarefa em reducao 1}
begin
ind1:= 'S'
end
else
begin
r:= succ(r);
ind6:= 'N';
while r <> 1 do
begin
if (r < n+1) and (ind6 <> 'S') then
begin
ReducaoLinha2(n,r,laux1,vmenor_lin,laux1);
ReducaoCol2(n,r,laux1,vmenor_col,laux1);
calc_Bound(n,vmenor_lin,vmenor_col);
calc_MaiorZero(n,laux1,matzero);
calc_Maior_ElementoMatzero(n,matzero,posi,posj);
{montar matriz reducao}
montar_mreducao(r,n,posi,posj,matzero);  {seta mreducao com os nposi}
end;
if r > n then
```

```
begin
{montar e imprimir sequencia}
imprimir_sequencia;
{atualizar sequencia e menor limitante}
if bound < bound_inicial then
begin
bound_inicial:= bound;
for i:= 1 to n+1 do
seqotima[i]:= sequencia[i];
end;
ind6:= 'S';          {indicador retrocesso}
r:= n-2;             {reducao ate r = n-2}
for i:= (r+1) to n do
for j:= 1 to n do
mreducao[i,j]:=0;
{limpar mprodfac}
zer:= r+1;
for i:= zer to n do
for j:= 1 to n do
mprodfac[pares[i,1],j]:=0;
for i:= (r+1) to n do
j:= 1 to 2 do
pares[I,J]:= 0;
gerar_matzero (n,r,clean,pares,matzero);  {ja deixa laux1 pronta}
end;                                  {para calcular matzero}
pos:= pares[r,1];
ind4:='N';
if pos <> 0 then
begin
{ver na linha pares[r,1] matzero posj diferente posj exit. em mprodfac}
posi:= pos;
posj:= pares[r,2];
maior_rotulo:= matzero[posi,posj];
ver_posj(n,posi,maior_rotulo,matzero,pares,nposj,ind4);
end;
if ind4 = 'N' then    {nao exite mais posj}
begin
ver_reducao1(r,n,pos,mreducao,nposi,ind5);
if ind5 = 'N' then
begin
posi:= pares[r,1];
posj:= pares[r,2];
ver_matzero(n,posi,posj,nposi,matzero,nposj);
if pos <> 0 then  {limpar linha nposi anterior}
for j:=1 to n do
mprodfac[pos,j]:=0;
{escalonar nposi e nposj}
pares[r,1]:= nposi;
pares[r,2]:= nposj;
if ind6 = 'S' then   {se houve retrocesso}
begin
{chamar gerar matzero houve modificacao em pares}
```

```
gerar_matzero(n,r,clean,pares,matzero);
ind6:= 'N';
end;
if r < n-1 then
Eliminar_Pares_Infactiveis(n,r,pares);
Eliminar_Linha_Col(n,pares[1,1],nPosI,nposj,laux1,laux1);
r:=succ(r);
end   {fim ind5}
else
begin
ind6:= 'S';
{limpar mprodfac}
for i:= r to zer do
for j:= 1 to n do
mprodfac[pares[i,1],j]:=0;
for i:= r to n do
for j:= 1 to n do
mreducao[i,j]:=0;
for i:= r to n do
for j:= 1 to 2 do
pares[i,j]:= 0;
r:= pred(r);
gerar_matzero(n,r,clean,pares,matzero);
end;     {fim do else}
end  {fim ind4}
else
begin
ind6:= 'N';
nposi:= posi;
pares[r,1]:=nposi;
pares[r,2]:=nposj;
gerar_matzero(n,r,clean,pares,matzero);
r:= succ(r);
end;        {fim do else}
 end;       {fim 2 while}
end;       {fim do else}
end;      {fim do 1 while}
end.     {fim Batjs6}
```

Unit UnBatch8;
interface

Tese de Mestrado

FACULDADE DE ENGENHARIA QUÍMICA

Departamento de Engenharia de Sistemas Químicos - DESQ-UNICANP

Descrição do Programa

{Este Programa calcula o tempo total de execucao tarefas Programadas.}

```
uses unbatch6,Maior_Va,Menor_Va,Fixa,Crt;

{Descrição de tipos}
type
vetorbat81  = Array[1..bat16+1] of integer;
vetorbat82  = Array[1..bat16] of real;
matbat81    = Array[1..bat16,1..bat16] of real;

{Descrição de variáveis}
var
m,indbat8,I8,J8,Q8,Ppos8,Pant8,Pos8          : integer;
TC,max18,max28,soma8                         : real;
Rj8,Vetor_Min8                               : vetorUn1;
C8                                           : matbat81;
vetaux18                                     : vetorUn2;
vbat18                                       : vetorUn1;
Dj8                                          : vetorUn1;
MatU8                                        : matbat81;
vTC                                          : vetorbat82;
Procedure calc_tempo_execucao(n:integer;m:integer;vbat4:vetorbat61; setup8:matbat81;var
tc:real);

implementation

Procedure calc_tempo_execucao(n:integer;m:integer;vbat4:vetorbat61; setup8:matbat81;var
tc:real);
Procedure Zerar_Lin(P: integer);
Var
J: integer;
begin
For J:= 1 to  M Do
setup8[P,J]:=0.;
end;

{Procedimento para a calculo do trabalho remanescente em um dado Processador}
Procedure Calc_RJ(A:Matbat81;N,M:integer;Var RJ: VetorUn1);
```

```
Var I,J : integer;
begin
For J:= 1 to M Do
begin
Rj[J]:= 0.;
For I:= 1 to N Do
Rj[J]:= Rj[J] + A[I,J];
end;
end;
{FIM}

Procedure Calc_U(B: Matbat81;N,M:integer;Var MatU: Matbat81);
Var J,J1,K1,K2: integer;
begin
K1:= 0;
For J:= 1 to (M-1) Do
begin
For K2:= 1 to N Do
begin
MatU[K2,J]:= 0.;
For J1:= 2 to M-K1 Do
MatU[K2,J]:= MatU[K2,J] + B[K2,J1+K1];
end;
K1:= K1 + 1;
end;
end;

{Procedimento para a calculo do trabalho minimo, remanescente, Umin,considerando os
Processadores J+1..M}
Procedure Calc_UMin(MatU:matbat81;P,N,M:Integer;Var Vetor_Min: VetorUn1);
var VetorU    : vetorUn1;
Min           : Real;
Pos           : Integer;
Begin
For J8:= 1 to (M-1) Do
Begin
For I8:= 1 to N Do
VetorU[I8]:= MatU[I8,J8];
BUSCA1(VetorU,N,Min);
Vetor_Min[J8]:= Min;
end;
end;
{FIM}
{Procedimento para a calculo dos Dj em cada Processdor}
Procedure Calc_DJ(D :Matbat81;RJ,Vetor_Min: VetorUn1;Var DJ: vetorUn1);
Var J,I: integer;
Begin
For J:= 1 to M Do
DJ[J]:= RJ[J] + D[Ppos8,J] + Vetor_Min[J];
End;
{FIM}
```

```
{Procedimento para a calculo do Menor Limitante para cada sequencia gerada}
Procedure Max_DJ(Bj:vetorUn1;M: Integer;var Max:real);
var
I,k: Integer;
Begin
max:=BJ[1];
For k:= 2 to M Do
If Bj[k] > Max then
max:= BJ[k];
end;
{FIM}

BEGIN                    {Inicio unBatch8};
{Calculo do tempo de Completacao}
For I8:= 1 to N Do
begin
Soma8:=0.;
For J8:=1 to M do
begin
C8[I8,J8]:= Soma8 + setup8[I8,J8];
Soma8:=  C8[I8,J8];
end;
end;
For I8:=1 To M Do
Vetor_Min8[I8]:=0;
Pant8:= 0;
For Q8:= 1 to N Do
begin
indbat8:= vbat4[Q8];
For Ppos8:= indBat8 to indbat8 Do
begin
{Calculo do Tempo de Completacao C}
if Q8 <> 1 Then
begin
For J8:= 1 to M Do
begin
if J8 = 1 then
begin
Vetaux18[1]:= 0.;
Vetaux18[2]:= C8[vbat4[Q8-1],J8];
end
else
begin
Vetaux18[1]:= C8[Ppos8,J8-1] ;
Vetaux18[2]:= C8[vbat4[Q8-1],J8];
end;
Busca(Vetaux18,2,Max28);
C8[Ppos8,J8]:= Max28 + setup8[Ppos8,J8];
end;
end;
{Zerando Linha}
Zerar_Lin(Ppos8);
```

```
{Chamando Calc_RJ}
Calc_RJ(setup8,N,M,RJ8);
{Chamando Calc_U}
Calc_U(setup8,N,M,MatU8);
{Chamando Calc_UMin}
Calc_Umin(MatU8,Ppos8,N,M,Vetor_Min8);
{Calculando os DJ}
Calc_DJ(C8,RJ8,Vetor_Min8,DJ8);
{Calculando Maximo dos DJ}
Max_DJ(DJ8,M,Max18);
end;                    {Fim_Ppos}
Zerar_Lin(indbat8);
vtc[Q8]:= max18;
end;                    {Fim de Q}
tc:= vtc[N];
{writeln('     ','TE = ',tc:4:1);}
end;
END.          {Fim unBatch8}
```

```
Program Batjs10;

{                              Tese de Mestrado

                    FACULDADE DE ENGENHARIA QUÍMICA

            Departamento de Engenharia de Sistemas Químicos - D.E.S.Q. - UNICAMP

Descrição do Programa

Algoritmo D - Seqüenciamento de tarefas com, com tempos de estabelecimento
dependentes da seqüência de produção e dependentes do processador.

Descrição da variáveis principais

N           - numero de tarefas a serem programadas
M           - numero de processadores presentes na planta batelada
Q           - nivel da arvore corrente
nnos        - numero de nos sondados
nseq        - numero de sequencia geradas
retrocesso  - indicador de "bactraking"
seqüência   - lista das tarefas programadas
upbound1    - valor da  funcao  da  sequencia  canditada a solucao otima
bound20     - menor limitante inicial da matriz clean ("upper bound") - solucao tentativa
novomestre  - no filho selecionado
time        - tempos de processamento das tarefas
clean       - tempos de estabelecimento
mplano              - arvore de busca
mprodback   - historico dos nos sondados

Descrição das unidades

nosfilhos   - unidade que expande um no mestre, gerando nos filhos
maior_va    - maior valor de um vetor, usada em nosfilhos
unbatch6    - esta  inativa  (avalia o custo de estabelecimento de uma sequencia
unbatch9    - reducao inicial de clean

{Unidades utilizadas}
uses nosfilhos_2,unbatch9,maior_va_1,fixa_4,unbatch6,dos,robat81,crt, printer;
-
{Definicao de tipos}
type
str12  = string[15];

{Definição de variáveis}
var
nnos,nseq,n,m,q,i,ii,j,ultimoproduto,qaux    : integer;
upbound1,upboundaux,novomestre,te            : real;
bound21,bound22                              : real;
cor1,cor2                                    : byte;
ind1,ind2,ind3,ind5,ind6,retrocesso          : char;
t,l1,l2                                      : str12;
```

```
hora1,minuto1,segundo1,seg1001          : word;
hora2,minuto2,segundo2,seg1002          : word;
tempo                                   : word;
sequencia,seqotima                      : vetorbat81;
c,time,time2                            : matbat81;
clean,clean2,laux11,laux12              : unmatbat3;
mplano, mprodback                       : matbat2;
rc                                      : string;

{Início dos Procedimentos}

{Ler arquivo}
Procedure learquivo1(var tt: matbat81;nome: str12;var n :integer);
var
f1 : text;
ch1  :char;
ncs1: string[2];
i1,j1,k1,e1 : integer;
s1 : array[1..bat1,1..bat2] of string[8];
begin
assign(f1,nome);
reset(f1);
for i1 := 1 to bat1 do
for j1 := 1 to bat2 do
s1[i1,j1] := '';
i1 := 0;
j1 := 1;
k1 := 0;
ncs1:='';
while not eof(f1) do
begin
read(f1,ch1);
if (k1 <> 2) and (ch1 <> #13) then
begin
k1:= k1 + 1;
ncs1:= ncs1+ch1;
val(ncs1,n,e1);
end
else
if ch1=#32 then
j1:= j1+1
else
if ch1=#13 then
begin
i1 := i1+1;
j1 := 1;
k1:= 2;
end
else
if ch1<>#10 then
s1[i1,j1] := s1[i1,j1]+ch1;
end;
```

```
for i1 := 1  to n do
for j1 := 1 to m do
val(s1[i1,j1],tt[i1,j1],e1);
close(f1);
end;
{FIM}

Procedure learquivo2(var tt: unmatbat3;nome: str12;var n :integer);
var
f1 : text;
ch1  :char;
ncs1: string[2];
i1,j1,k1,e1 : integer;
s1 : array[1..bat1,1..bat1] of string[8];
begin
assign(f1,nome);
reset(f1);
for i1 := 1 to bat1 do
for j1 := 1 to bat1 do
s1[i1,j1] := '';
i1 := 0;
j1 := 1;
k1 := 0;
ncs1:='';
while not eof(f1) do
begin
read(f1,ch1);
if (k1 <> 2) and (ch1 <> #13) then
begin
k1:= k1 + 1;
ncs1:= ncs1+ch1;
val(ncs1,n,e1);
end
else
if ch1=#32 then
j1:= j1+1
else
if ch1=#13 then
begin
i1 := i1+1;
j1 := 1;
k1:= 2;
end
else
if ch1<>#10 then
s1[i1,j1] := s1[i1,j1]+ch1;
end;
for i1 := 1  to n do
for j1 := 1 to n do
val(s1[i1,j1],tt[i1,j1],e1);
close(f1);
end;
```

```
Procedure Retirar_tarefa_2(q:integer;sequencia:vetorbat81);
var j2,prod2: integer;
begin
for j2:= 1 to N do
if mplano[q,j2] >= upbound1 then
mplano[q,j2]:=0;
end;

Procedure imprima_sequencia_4(sequencia:vetorbat81;upbound1:real);var i4 :integer;
 begin
write('sequencia: ');
for i4:= 1 to n+1 do
write(sequencia[i4],'-');
write(' ');
writeln(lst,'te = ',upbound1:4:1);
end;

Procedure ver_plano1_3(n3,q3:integer;sequencia3:vetorbat81;var ind3:char; var
novomestre:real);
var i3,j3       :integer;
boundaux3    : real;
begin
novomestre:=0;
{Procedimento para retirada de tarefas com menor limitante superior ao
upbound-Processo "fathomed"}
retirar_tarefa_2(q3,sequencia3);
ind3:= 'N'; {lista vazia no plano 1}
for j3:= 1 to n3 do
if mplano[q3,j3] <> 0 then
begin
novomestre:= j3;
boundaux3:= mplano[q3,j3];
j3:= n3;
ind3:= 'S';
end;
for j3:= 1 to n3  do
if (mplano[q3,j3] <> 0) and (mplano[q3,j3] < upbound1) and (mplano[q3,j3] < boundaux3)
then
begin
novomestre:= mplano[0,j3];
boundaux3:= mplano[q3,j3];
ind3:= 'S';
end;
if novomestre <> 0 then
begin
mplano[q3,trunc(novomestre)]:=0;
mprodback[q3,trunc(novomestre)]:= novomestre;
end;
end;
```

```
Procedure ver_plano_atual_5(n4,q4:integer;sequencia4:vetorbat81; var ind4:char;var
novomestre:real);
var i4,j4,j44  : integer;
boundaux4  : real;
indicador4 : char;
begin
{Este procedimento encontra o mais baixo valor para LB - menor limitante}
novomestre:=0;
{Procedimento para retirada de tarefas com menor limitante superior ao
upbound - Processo "fathomed"}
retirar_tarefa_2(q4,sequencia4);
ind4:= 'N';        {lista vazia no plano > 1}
indicador4:= 'N';
for j4:= 1 to n4 do
if mplano[q4,j4] <> 0 then
begin
novomestre:= j4;
boundaux4:= mplano[q4,j4];
ind4:= 'S';
if ind5 <> 'N' then
upboundaux:= mplano[q4,j4];   {melhor solucao factivel}
indicador4:= 'S';
j44:=j4;
j4:= n4;
end;
if indicador4 = 'S' then
begin
for j4:= j44 to n4  do
if (mplano[q4,j4] <> 0) and (mplano[q4,j4] < upbound1) and(mplano[q4,j4] < boundaux4)
then
begin
novomestre:= mplano[0,j4];
boundaux4:= mplano[q4,j4];
if ind5 <> 'N' then
upboundaux:= mplano[q4,j4];   {melhor solucao factivel}
ind4:= 'S';
end;
if (ind5 = 'N') and (q4 = n4-2) then  {primeira solucao factivel}
begin
ind5:= 'S';
upbound1:= mplano[q4,trunc(novomestre)];
upboundaux:= upbound1;
end;
mplano[q4,trunc(novomestre)]:=0;
mprodback[q4,trunc(novomestre)]:= novomestre;
end;
end;

Procedure cor_texto_fundo(cor1,cor2:byte);
begin
textcolor(cor1);
textbackground(cor2);
```

```
clrscr;
end;

Begin            {inicio Batjs10}
clrscr;
cor1:= 5; cor2:= 7;
cor_texto_fundo(cor1,cor2);

{Leitura dos Dados}
write(':6,'Entre com o numero de processadores M =  ');
readln(m);
write(':6,'Entre com o nome do arquivo processamento = ');
readln(t);
learquivo1(time,t,n);
write(':6,'Entre com o nome do arquivo1 estabelecimento = ');
readln(l1);
learquivo2(clean,l1,n);
write(':6,'Entre com o nome do arquivo2 estabelecimento = ');
readln(l2);
learquivo2(clean2,l2,n);writeln;
write(':6,'Entre com o recurso computacional= ');
readln(rc);

{Impressao dos Dados}
writeln(lst);
writeln(lst,':20,'tempos de Processamento');writeln(lst);
writeln(lst,':28,'Processadores');writeln(lst);
write(lst,':27);
for j:= 1 to m do
write(lst,j,':6);writeln(lst);
for i:= 1 to n do
begin
write(lst,':25);
for j:= 1 to m do
write(lst,time[i,j]:4:1,' ');
writeln(lst);
end;
writeln(lst);
writeln(lst,':20,'tempos de Estabelecimento-Proc 1');writeln(lst);
writeln(lst,':28,'Produtos');writeln(lst);
for i:= 1 to n do
begin
write(lst,':25);
for j:= 1 to n do
write(lst,clean[i,j]:4:1,' ');
writeln(lst);
end;
writeln(lst);
writeln(lst,':20,'tempos de Estabelecimento-Proc 2');writeln(lst);
writeln(lst,':28,'Produtos');writeln(lst);
for i:= 1 to n do
begin
```

```
write(lst,'':25);
for j:= 1 to n do
write(lst,clean2[i,j]:4:1,' ');
writeln(lst);
end;

writeln('Programa em execucao, aguarde...');
writeln(lst,'Sequencias analisadas');
writeln(lst);
{Inicializacao das Variaveis}
{inicializar sequencia}
for i:= 1 to (bat1+1) do
begin
sequencia[i]:=0;
seqotima[i]:=0;
end;
{inicializar variaveis da arvore}
for i:= 0 to bat1 do
for j:= 1 to bat1 do
mplano[i,j]:= 0;
{inicializar matriz de tarefas analisadas}
for i:= 0 to bat1-1 do
for j:= 1 to bat1 do
mprodback[i,j]:=0.;
{inicializar matriz de tempo de execucao parcial}
for i:= 1 to bat1 do
for j:= 1 to bat2 do
c[i,j]:= 0;
{setando nivel 0 da arvore com produtos}
for j:= 1 to n do
mplano[0,j]:= j;

nseq:=0;
nnos:=0;
novomestre:=0;
{Inicio de tomada de tempo}
gettime(hora1,minuto1,segundo1,seg1001);
{Calculo da solucao tentativa}
{Montar uma sequencia de permutacao}
for i:= 1 to n do
sequencia[i]:= i;
sequencia[n+1]:= sequencia[1];
for i:= 1 to n do
seqotima[i]:= sequencia[i];
upbound1:= 1000;
{Calcular reducao inicial de clean(unidade robat81)}
{Calcular bound inicial de reducao}
cal_bound20(n,clean,laux11,bound21);
cal_bound20(n,clean2,laux12,bound22)
for ii:= 1 to n do
begin
{setando 1 plano}
```

```
for i:= 1 to n do
sequencia[i]:= 0;
{inicializar matriz de tempo de execucao parcial}
for i:= 1 to bat1 do
for j:= 1 to bat2 do
c[i,j]:= 0;
q:= 1;    {vai equivaler ao segundo nivel da arvore,o primeiro e ocupado por designacao}
qaux:=q;  {qaux indicara a sequencia parcial inicial}
sequencia[q]:=ii;
sequencia[n+1]:= ii;
retrocesso:= 'S';
gerar_filhos(n,m,q,time,laux11,laux12,c,clean,clean2,bound21,bound22,
sequencia,retrocesso, mplano,c);
ind1:= 'S';
ind5:= 'N';{indicara se o primeiro upbound foi encontrado entao
ind4:= 'S'}
while ind1 <> 'N' do
begin
{Buscar novo no mestre}
retrocesso:='S';
ver_plano1_3(n,q,sequencia,ind2,novomestre);
if ind2 = 'S' then
nnos:=succ(nnos);
if ind2 = 'N' then
ind1:= 'N'  {if ind1 = N fim do programa}
else
begin
q:= succ(q);
sequencia[q]:= trunc(novomestre);
while q <> qaux do
begin
{Zerar sequencia de q ate n}
for i:= q+1 to n do
sequencia[i]:= 0;
{explodir nomestre - gerar nos filhos}
if q <= n-1 then
begin
if q = n-1 then
begin
q:= pred(q);
retrocesso:= 'N';
for i:= q+1 to n do
for j:= 1 to n do
mprodback[i,j]:=0;
for i:= q+1 to n do
sequencia[i]:= 0;
end;
gerar_filhos(n,m,q,time,laux11,laux12,c,clean,clean2,bound21,bound22,
equencia,retrocesso,mplano,c);
end;
{Eliminar os nos das tarefas ja sondados}
for i:=1 to n do
```

```
if mprodback[q,i] <> 0 then
mplano[q,i]:= 0;
{Buscar novomestre- elimina os menores limitante maiores que upbound1}
ver_plano_atual_5(n,q,sequencia,ind3,novomestre);
if novomestre <> 0 then
begin
nnos:= succ(nnos);
q:= succ(q);
retrocesso:= 'S';
sequencia[q]:= trunc(novomestre);
end;
if ind3 = 'N' then
begin
{retroceder no plano}
{limpar matriz tempo de completacao}
c[sequencia[q],1]:= 0;
c[sequencia[q],2]:= 0;
q:= pred(q);
c[sequencia[q],1]:= c[sequencia[q],1] -
clean[sequencia[q],sequencia[q+1]];
c[sequencia[q],2]:= c[sequencia[q],2] -
clean2[sequencia[q],sequencia[q+1]];
{retirar de c tempo de estabelcimento}
{antes de zerar sequencia de q+1 a n}
retrocesso:= 'N';
for i:= (q+1) to n do
for j:= 1 to n do
mprodback[i,j]:=0;
for i:= q+1 to n do
sequencia[i]:= 0;
end
else
begin
if q = n-1 then
begin
if upboundaux <= upbound1 then
{completar sequencia}
for ultimoproduto:= 1 to n do
begin
ind6:= 'S';
for i:= 1 to n do
if sequencia[i] = ultimoproduto then
begin
ind6:= 'N';
i:= n;
end;
if ind6 = 'S' then
begin
sequencia[n]:= ultimoproduto;
ultimoproduto:= n;
end;
end;
```

```
imprima_sequencia_4(sequencia,upboundaux);
nseq:= nseq+1;
{substituir o upbound1}
if upboundaux <= upbound1 then
begin
upbound1:= upboundaux;
for i:= 1 to n+1 do
seqotima[i]:= sequencia[i];
end;
end;{fim do if q = n}
end; {fim do else}
end; {fim do while}
end; {fim do else1}
end; {fim while}
end; {fim for}
{Fim de tomada de tempo}
gettime(hora2,minuto2,segundo2,seg1002);
tempo:= ((hora2-hora1)*3600) + ((minuto2-minuto1)*60) +
(segundo2 -segundo1);
{Impressao dos Resultados}
writeln;
writeln;
writeln('Recurso computacional: Computador ',rc);
writeln('Linguagem de programacao: Turbo Pascal');
writeln('Tempo de execucao: ',tempo,' ','segundos');
writeln;
writeln;
writeln('Programa de Producao');
writeln(lst);
write(lst,'sequencia otima: ');
for i:= 1 to n+1 do
write(lst,seqotima[i],'-');
write(lst,'  ');
writeln(lst,'te = ',upbound1:3:1,' u.t.');
writeln('fim do programa');
writeln;
write('Numero de sequencia= ');
write(nseq);
fixatela;
End.    {fim Batjs10}
```

```
Unit nosfilhos_2;
Interface

{Descrição do Programa

Esta unidade expande um nó mestre, gerando nós filhos

Descrição das variáveis principais}

Q2          - Indica o plano corrente da arvore de busca
N2          - numero de tarefas a serem programadas
M2          - numero de processadores presentes na planta batelada
rj2         - carga de trabalho remanescente na maquina corrente
dj2         - vetor da funcao custo avaliada em cada maquina
bj2         - vetor dos valores maximo de dj - "lower bound"
c2          - matriz de tempos de execucao de sequencias parciais
matu2       - carga de trabalho minima dos trabalhos nao sequenciados
sequencia2  - lista das tarefas programadas
time        - tempos de processamento das tarefas
mplano2     - arvore de busca
lbound1     - menor limitante utilizando a Heuristica "Full Machine" para a sequencia parcial
lbound2 = lbound22 + bound2
lbound2     - menor limitante do metodo de reducao
sij_seq     - tempo de estabelecimento comprometido pela sequencia parcial

uses maior_va_1,menor_va_3,unbatch6,unbatc10,robat81,crt;

{Definição de tipos}
type
vetorbat2 = array[1..bat1] of real;
matbat2   = array[0..bat1-2,1..bat1] of real;

{Dedinição de variáveis}
var
indj,q2,i2,ii2,iii2,j2,ppos2,pant2,pant22,pos2,m2,n2      : integer;
LB,max22,soma2,minbj2,sij_seq1,sij_seq2                   : real;
deltalb,lbound1,lbound21,lbound22                         : real;
ind22                                                     : char;
rj2,vetor_min2                                            : vetorun1;
c2                                                        : matbat81;
vetaux12                                                  : vetorun2;
vbat12,vpant2                                             : vetorun1;
dj2                                                       : vetorun1;
bj2                                                       : vetorbat2;
matU2                                                     : matbat81;
vlbound2                                                  : vetorun1;
Procedure Gerar_filhos(n2,m2,q:integer;time2:matbat81;laux11,laux12:unmatbat3;
c2:matbat81;clean1,clean2:unmatbat3;bound1,bound2:real;sequencia2:vetorbat81;
retrocesso:char;var mplano2:matbat2; var c3:matbat81);
```

```
Implementation
Procedure Gerar_filhos(n2,m2,q:integer;time2:matbat81;laux11,laux12:unmatbat3;
c2:matbat81;clean1,clean2:unmatbat3;bound1,bound2:real;sequencia2:vetorbat81;
retrocesso:char;var mplano2:matbat2; var c3:matbat81);

Procedure Armaz_Lin(ppos2:integer; var vetorbat3: vetorUn1);
var
j1 :integer;
begin
for j1:= 1 to m2 do
vetorbat3[J1]:= time2[ppos2,j1];
end;

Procedure zerar_lin(ppos2: integer);
var
J: integer;
begin
for J:= 1 to m2 Do
time2[Ppos2,J]:=0.;
end;

Procedure Restituir_Lin(Ppos2: integer);
var
J :integer;
begin
For J:= 1 to M2 Do
Time2[Ppos2,J]:= vbat12[J];
end;

{Procedimento para a calculo do trabalho remanescente em um dado Processador}
Procedure calc_rj(A: Matbat81;N,M:integer;Var RJ: VetorUn1);
var i,j : integer;
begin
for j:= 1 to m Do
begin
rj[J]:= 0.;
for I:= 1 to N Do
rj[J]:= rj[J] + a[I,J];
end;
end;
{FIM}



Procedure Calc_U(B: Matbat81;N,M:integer;Var MatU: Matbat81);
var j,j1,k1,k2: integer;
begin
k1:= 0;
for j:= 1 to (m-1) Do
begin
for k2:= 1 to n Do
```

```
begin
matu[k2,j]:= 0.;
for j1:= 2 to m-k1 Do
matu[k2,j]:= matu[k2,J] + B[k2,j1+k1];
end;
k1:= k1 + 1;
end;
end;

{Procedimento para a calculo do trabalho minimo, remanescente, Umin,considerando os
Processadores J+1..M}

Procedure Calc_UMin(MatU:matbat81;P,N,M:Integer;Var Vetor_Min: VetorUn1);
var vetoru    : vetorUn1;
min        : real;
pos,j1,i1  : integer;
Begin
for J1:= 1 to (m-1) Do
Begin
for I1:= 1 to n Do
vetorU[I1]:= matU[I1,J1];
BUSCA1(Vetoru,n,min);
vetor_min[J1]:= min;
end;
End;
{FIM}

{Procedimento para a calculo dos Dj em cada Processdor}
Procedure Calc_DJ(D :Matbat81;RJ,Vetor_Min: VetorUn1;Var DJ: vetorUn1);
var j,i: integer;
begin
for j:= 1 to m2 do
dj[j]:= rj[J] + d[ppos2,j] {+ vetor_min[j]};
end;
{FIM}

{Procedimento para a calculo do Menor Limitante em para cada sequencia gerada}
Procedure max_dJ(bj:vetorun1;m: integer;var max:real);
var
i,k: integer;
begin
max:=bJ[1];
for k:= 2 to m do
If bj[k] > max then
max:= bJ[k];
end;
{FIM}

Procedure incluir_tempo_estabelecimento(m2,pant2,ppos2:integer);
var
j2   : integer;
begin
```

```
c2[pant2,1]:= c2[pant2,1] + clean1[pant2,ppos2];
c2[pant2,2]:= c2[pant2,2] + clean2[pant2,ppos2];
end;

Procedure deter_tempo_seqparcial(q2:integer;vpant2:vetorun1; sequencia2:vetorbat81;var
c2:matbat81);
var i2,j2,pant2,ppos2     : integer;
begin
if q2 = 1 then
begin
soma2:=0;
ppos2:=sequencia2[q];
for j2:=1 to m2 do
begin
c2[ppos2,j2]:= soma2 + vpant2[j2];
soma2:=  c2[ppos2,j2];
end;
end;
if q2 <> 1 then
begin
pant2:=sequencia2[q2-1];
ppos2:= sequencia2[q2];
incluir_tempo_estabelecimento(m2,pant2,ppos2);
for j2:= 1 to m2 do
begin
if j2 = 1 then
begin
vetaux12[1]:= 0.;
vetaux12[2]:= c2[pant2,j2];
end
else
begin
vetaux12[1]:= c2[ppos2,j2-1];
vetaux12[2]:= c2[pant2,j2];
end;
busca(vetaux12,2,max22);
c2[ppos2,j2]:= max22 + vpant2[j2];
end; {fim j2}
end;
zerar_lin(ppos2);
end;


Procedure calc_bj2_min(n2:integer;bj2:vetorbat2;var minbj2:real);
var
i2:  integer;
begin
minbj2:=100000;
for i2:= 1 to n2 do
if bj2[i2] <> 0 then
if bj2[i2] < minbj2 then
minbj2:= bj2[i2];
```

```
end;

Procedure cal_sij_seq_parcial(q2:integer;sequencia2:vetorbat81; clean:unmatbat3;var
sij_seq:real);
var
i2   : integer;
begin
sij_seq:= 0;
for i2:= 1 to q2 do
sij_seq:= sij_seq + clean[sequencia2[i2],sequencia2[i2+1]];
end;

Procedure retirar_tempo_estabelecimento(m2,pant2,ppos2:integer);
var
j2   : integer;
begin
c2[pant2,1]:= c2[pant2,1] - clean1[pant2,ppos2];
c2[pant2,2]:= c2[pant2,2] - clean2[pant2,ppos2];
end;
{FIM}

BEGIN    {Inicio nosfilhos};
for i2:=1 to m2 do
vetor_min2[I2]:=0;
for i2:= 1 to n2 do
bj2[i2]:= 0;
if retrocesso = 'S' then
begin
ppos2:= sequencia2[q];
for j2:=1 to n2 do
vpant2[j2]:= time2[ppos2,j2];
deter_tempo_seqparcial(q,vpant2,sequencia2,c2);
end;
{zerar n ja sequenciadas de time2}
if q <> 1 then
for i2:= 1 to q do
for j2:= 1 to m2 Do
time2[sequencia2[i2],j2]:=0.;
for iii2:= 1 to n2 do
begin
pant2:=0;
q2:= q;
ind22:= 'S';
q2:= succ(q2);
for ii2:= q2 to n2 do
sequencia2[ii2]:= 0;
for ii2:= 1 to n2 do
if sequencia2[ii2] = iii2 then
begin
ind22:= 'N';
ii2:= n2;
end;
```

```
if ind22 = 'S' then
begin
{armazenar linha sequencia2[q2]}
armaz_lin(iii2,vpant2);
for i2:= q2 to n2 do
sequencia2[i2]:= 0;
sequencia2[q2]:=iii2;
{Determinacao do tempo de execucao da sequencia parcial}
deter_tempo_seqparcial(q2,vpant2,sequencia2,c2);
for ppos2:= 1 to n2 do    {se sequencia ja tem 1-2 nao faz}
begin
for i2:= 1 to q2 do
if (ppos2 = sequencia2[i2]) then
pant2:= sequencia2[i2];
if pant2 <> ppos2 then
begin
{adicionar ppos2 em sequencia2}
q2:=q+1;
q2:=succ(q2);
pant2:= sequencia2[q2-1];
sequencia2[q2]:= ppos2;
{Armazenando Linha}
Armaz_lin(ppos2,vbat12);
{calculo do tempo de execucao C}
deter_tempo_seqparcial(q2,vbat12,sequencia2,c2);
{calcular novos tempo de execucao com tempos de estabelecimento}
{Chamando Calc_rj}
calc_RJ(time2,n2,m2,rj2);
{Chamando Calc_U}
{Calc_U(time2,n2,m2,matu2);}
{Restituindo linha}
Restituir_Lin(ppos2);
{Chamando Calc_umin}
{Calc_umin(matu2,ppos2,n2,m2,vetor_min2);}
{Calculando os DJ}
{Para o Primeiro Processador}
calc_dj(c2,rJ2,vetor_min2,dj2);
calc_lbound2(q2-1,n2,sequencia2,laux11,lbound21);
cal_sij_seq_parcial(q2-1,sequencia2,clean1,sij_seq1);
dj2[1]:= dj2[1] + (lbound21+bound1-sij_seq1);
{Para o segundo Processador}
cal_sij_seq_parcial(q2-1,sequencia2,clean2,sij_seq2);
calc_lbound2(q2-1,n2,sequencia2,laux12,lbound22);
dj2[2]:= dj2[2] + (lbound22 + bound2 - sij_seq2);
{Calculando Maximo dos DJ}
max_dj(dj2,m2,LB);          {indicar a maquina limitante-gargalo}
{calcular o lbound2}        {do processo}
bj2[ppos2]:= LB;
{Retirar de pant2 sij}
retirar_tempo_estabelecimento(m2,pant2,ppos2);
{zerar linha ppos2 de c2}
for j2:= 1 to m2 do
```

```
c2[ppos2,j2]:= 0;
end;    {fim if}
end;   {fim_ppos2}
{calculo do lbound2}
q2:= pred(q2);
calc_bj2_min(n2,bj2,minbj2);
mplano2[q2-1,iii2]:= minbj2;
{zerar bj2}
for i2:= 1 to n2 do
bj2[i2]:=0.;
{restituir linha de time2}
for j2:= 1 to m2 do
time2[iii2,j2]:= vpant2[j2];
{Retirar de pant2 sij}
retirar_tempo_estabelecimento(m2,sequencia2[q2-1],iii2);
{zerar linha ppos2 de c2}
for j2:= 1 to m2 do
c2[iii2,j2]:= 0;
end;   {fim if2}
end;  {fim for}
for i2:= 1 to n2 do
for j2:= 1 to m2 do
c3[i2,j2]:= c2[i2,j2];
end;
end.              {fim nos nosfilhos}
```

```
Unit unbatch9;
Interface
{                              Tese de Mestrado

                    FACULDADE DE ENGENHARIA QUÍMICA

            Departamento de Engenharia de Sistemas Químicos - D.E.S.Q.

Descrição do Programa
Esta unidade executa a redução inicial da matriz de estabelecimento

Descrição da variáveis principais

laux1          - matriz reduzida, geradora de matzero}

uses unbatch6,crt;

{Definicao de tipos}
type
vetorbat3  = Array[1..bat16] of real;

{Definição de variáveis}
var
i,j                          : integer;
vmenor_lin,vmenor_Col        : vetorbat3;
laux1                        : unmatbat3;
Procedure cal_bound20(n:integer;clean:unmatbat3;var laux1:unmatbat3; var bound20:real);

Implementation
Procedure cal_bound20(n:integer;clean:unmatbat3;var laux1:unmatbat3; var bound20:real);

{Início dos Procedimentos}

{Procedimento Reducao-Linha inicial da matriz L}
Procedure Sub_Linha(i,n:integer;l:unmatbat3;menor:real;var laux2:unmatbat3);
var
j: integer;
begin
for j:= 1 to n do
if (j <> i) and (l[i,j] > 0.0) then
laux2[i,j]:= l[i,j] - menor;
end;



Procedure Reducao_Linha1(n:integer;l:unmatbat3;var laux1:unmatbat3; var
vmenor_lin:vetorbat3);
var
i1,j1      :Integer;
```

```
menor    :real;
begin
for i1:= 1 to n do
vmenor_lin[i1]:= 0.;
for i1:= 1 to n do
for j1:= 1 to n do
If i1 = j1 then
laux1[i1,j1]:= -1.;
for i1:= 1 to n Do
begin
menor:= 1000.;
for j1:= 1 to n Do
If (j1<>i1) and (l[i1,j1] > 0. ) then
if l[i1,j1] <= menor then
begin
menor:= l[i1,j1];
vmenor_lin[I1]:=L[i1,j1];
end;
Sub_Linha(i1,n,l,menor,laux1)
end;
end;
{Fim}

{Procedimento Reducao-Coluna inicial da matriz L}
Procedure Sub_Coluna1(j,n:integer;laux1:unbactbat3;menor:real; var vaux3:vetorbat3);
var
k1          : integer;
begin
for k1:= 1 to N do
if (k1 <> j) and (clean[k1,j] >= 0.0) then
vaux3[k1]:= laux1[k1,j] - menor
else
vaux3[k1]:= -1;
end;

Procedure ReducaoCol1(n:integer;laux1:unmatbat3;var vmenor_col:vetorbat3; var
laux3:unmatbat3);
var
i1,j1,pos,k1,k2     : integer;
ind1                : vetorbat3;
begin
for i1:= 1 to n do
vmenor_col[I1]:= 0.;
for j1:= 1 to n do
begin
ind1:= 'N';
for i1:= 1 to n do
if laux1[i1,j1] = 0. then
ind1:= 'S';
if ind1 = 'S' then
for k1:=1 to n do
laux3[k1,j1]:= laux1[k1,j1];
```

```
if ind1 = 'N' then
begin
menor:=100000.;
for k1:= 1 to n Do
If (j1<>k1) then
if laux1[k1,J1] <= menor then
begin
pos:=k1;
menor:= laux1[k1,J1];
end;
vmenor_col[J1]:=Laux1[Pos,J1];
sub_coluna1(J1,N,Laux1,menor,vaux3);
for k2:=1 to n do
laux3[k2,J1]:= vaux3[k2];
end;
end;
end;
{Fim}

{Procedimento Calculo do Bound }
Procedure Calc_Bound(N:integer;vmenor_lin,vmenor_col:vetorbat3; var bound2i:real);
var
i,j    :integer;
begin
for i:= 1 to n do
bound2i:= bound2i + vmenor_lin[I];
for j:= 1 to n do
bound2i:= bound2i + vmenor_col[J];
end;
{Fim}

Begin                    {Inicio Bound2i};
bound20:=0;
for i:= 1 to bat16 do
for j:= 1 to bat16 do
laux1[I,J]:=0.;
reducao_linha1(n,clean,laux1,vmenor_lin);
reducaoCol1(n,laux1,vmenor_col,laux1);
calc_bound(n,vmenor_lin,vmenor_col,bound20);
end;
End.  {fim bound2i}
```

Program Batjs8;

Tese de Mestrado

FACULDADE DE ENGENHARIA QUÍMICA

Departamento de Engenharia de Sistemas - DESQ - UNICAMP

Descrição do Programa

Este Programa estabelece Seqüência ótima de tarefas com restrição de ordem de manufatura. via Metodologia Branch and Bound

```
{bat1 e bat2 estão definidas na unidade 1}

uses nosfilhos_2,maior_va_1,fixa_4,crt;

type
str12  = string[15];

{Definição de variáveis}
var
N,M,i,j,Prod1,q,qaux,np,produltimo          : integer;
soma,upbound1,upboundaux,novomestre         : real;
ind1,ind2,ind3,ind5                         : char;
sequencia,seqnp                             : vetorbat1;
c,time,timelabel                            : matbat1;
mplano, mprodback                           : matbat2;
t                                           : str12;

{Ler arquivo}
Procedure learquivo1(var tt: matbat1;nome: str12;var n :integer);
var
f1 : text;
ch1  :char;
ncs1: string[2];
i1,j1,k1,e1 : integer;
s1 : array[1..bat1,1..bat2] of string[8];
begin
assign(f1,nome);
reset(f1);
for i1 := 1 to bat1 do
for j1 := 1 to bat2 do
s1[i1,j1] := '';
i1 := 0;
j1 := 1;
k1 := 0;
ncs1:='';
while not eof(f1) do
```

```
begin
read(f1,ch1);
if (k1 <> 2) and (ch1 <> #13) then
begin
k1:= k1 + 1;
ncs1:= ncs1+ch1;
val(ncs1,n,e1);
end
else
if ch1=#32 then
j1:= j1+1
else
if ch1=#13 then
begin
i1 := i1+1;
j1 := 1;
k1:= 2;
end
else
if ch1<>#10 then
s1[i1,j1] := s1[i1,j1]+ch1;
end;
for i1 := 1  to n do
for j1 := 1 to m do
val(s1[i1,j1],tt[i1,j1],e1);
close(f1);
end;
{FIM}

Procedure Retirar_tarefa_2(Q:integer;sequencia:vetorbat1);
var j2,prod2: integer;
begin
for j2:= 1 to n+1 do
if mplano[q,j2] > upbound1 then
mplano[q,j2]:=0;
end;

Procedure Imprima_sequencia_4(sequencia:vetorbat1;upbound1:real);
var i4 :integer;
begin
write('sequencia: ');
for i4:= 1 to ((n-np)+1) do
write(sequencia[i4],'-');
writeln('  tc = ',upbound1:4:1);
end;

Procedure Ver_Plano1_3(n3,q3:integer;sequencia3:vetorbat1;var ind3:char; var
novomestre:real);
var i3,j3,jj3        : integer;
```

```
    menor3        : real;
begin
novomestre:=0;
{Procedimento para retirada de tarefas com menor limitante superior ao upbound}
retirar_tarefa_2(q3,sequencia3);
ind3:= 'N';
for j3:=1 to n3 do
if mplano[q3,j3] <> 0 then
begin
menor3:= mplano[q3,j3];
ind3:= 'S';
jj3:= n3;
j3:= n3;
end;
for j3:= 1 to n3  do
if (mplano[q3,j3] <> 0) and (mplano[q3,j3] <= upbound1) and (mplano[q3,j3] < menor3) then
begin
menor3:= mplano[q3,j3];
jj3:= j3;
ind3:= 'S';
end;
novomestre:= mplano[0,jj3];
mprodback[q3,jj3]:= novomestre;
mplano[q3,jj3]:=0;
end;

Procedure ver_plano_atual_5(n4,q4:integer;sequencia4:vetorbat1; var ind4:char;var
novomestre:real);
var i4,j4,jj4        : integer;
    menor4           : real;
begin
novomestre:=0;
{Procedimento para retirada de tarefas com menor limitante superior ao
upbound}
retirar_tarefa_2(q4,sequencia4);
ind4:= 'N';
for j4:=1 to n4+1 do
if mplano[q4,j4] <> 0 then
begin
menor4:= mplano[q4,j4];
ind4:= 'S';
jj4:= j4;
upboundaux:= menor4;
j4:= n4+1;
end;
for j4:= 1 to n4+1  do
if (mplano[q4,j4] <> 0) and (mplano[q4,j4] <= upbound1) and (mplano[q4,j4] < menor4) then
begin
menor4:= mplano[q4,j4];
```

```
if ind5 <> 'N' then
upboundaux:= menor4;
if (ind5 = 'N') and (q = ((n-np)+1)) then
begin
ind5:= 'S';
upbound1:= mplano[((n-np)+1),trunc(novomestre)];
upboundaux:= upbound1;
end;
ind4:= 'S';
jj4:= j4;
end;
if ind4 = 'S' then
begin
novomestre:= mplano[0,jj4];
mplano[q4,jj4]:=0;
mprodback[q4,jj4]:= novomestre;
end;
end;

Begin                    {Início Batjs8}
clrscr;
{leitura dos dados}
for i:= 1 to bat1+1 do
for j:= 1 to bat2 do
time[i,j]:=0;
write(':6,'Entre com o numero de processadores M =  ');
readln(M);
write(':6,'Entre com o nome do arquivo = ');
readln(t);
learquivo1(time,t,n);
{inicializando timelabel}
for i:= 1 to bat1+1 do
for j:= 1 to bat2 do
timelabel[i,j]:=0;
for i:= 1 to bat1+1 do
seqnp[i]:= 0;

{Impressão da matriz tempo}
clrscr;
writeln;
writeln(':20,'tempos de Processamento');writeln;
writeln(':28,'Processadores');writeln;
write(':27);
for J:= 1 to m do
write(J,':6);writeln;
for I:= 1 to n Do
begin
write(':25);
for J:= 1 to m Do
```

```
write(time[i,j]:4:1,'  ');
writeln;
end;
{ler produtos com restricao de precedencia}
write('Entre com numero de tarefas ao grupo de precedencia = ');
readln(np);
writeln;writeln;
writeln('Entre com a sequencia de tarefas');
for i:= 1 to np do
readln(seqnp[i]);

{Montar timelabel a partir de time}
for i:= 1 to n do
for j:= 1 to m do
timelabel[i,j]:= time[i,j];
{zerar linhas de timelabel das tarefas presentes em seqnp}
for i:= 1 to np do
for j:= 1 to m do
timelabel[seqnp[i],j]:=0;
{Gerar pseudo-produto na linha (n+1) de timelabel}
for j:= 1 to m do
begin
soma:=0.;
for i:= 1 to np do
soma:= soma + time[seqnp[i],j];
timelabel[n+1,j]:= soma;
end;

{inicializar sequencia}
for i:= 1 to (bat1+1) do
sequencia[i]:=0;
{inicializar matriz de tarefas analisadas}
for i:= 0 to bat1+1 do
for j:= 1 to bat1+1 do
begin
mprodback[i,j]:=0.;
mplano[i,j]:=0;
end;
{atribuir primeira linha de mplano com as tarefas}
for j:= 1 to n+1 do
mplano[0,j]:=j;
{setar produltimo com o ultimo produto da sequencia de precedencia}
produltimo:= seqnp[np];
q:=1;
qaux:=q;  {qaux indicara a sequencia parcial inicial}
Gerar_filhos(n,m,q,np,produltimo,timelabel,time,seqnp,sequencia,mplano);
{calc_upbound(n,m,time,mplano);}
upbound1:=10000.;
upboundaux:= upbound1;
```

```
ind1:= 'S';
ind5:= 'N';  {indicara se o primeiro upbound foi encontrado entao ind4:= 'S'}
while ind1 <> 'N' do
begin

for i:= 2 to ((n-np)+1) do
for j:= 1 to n do
mplano[i,j]:=0;

{Buscar novo no mestre}
ver_Plano1_3(n+1,q,sequencia,ind2,novomestre);
sequencia[q]:= trunc(novomestre);

if ind2 = 'N' then
ind1:= 'N'        {fim do programa}
else
begin
q:= succ(q);
while q <> qaux do
begin
{Zerar sequencia de q ate n}
for i:= q to ((n-np)+1) do
sequencia[i]:= 0;
{explodir nomestre - gerar nos filhos}
if q <= ((n-np)+1) then
gerar_filhos(n,m,q,np,produltimo,timelabel,time,seqnp,sequencia,mplano);
if (q-1) = ((n-np)+1) then
begin
q:= q-2;
for i:= (q+1) to ((n-np)+1) do
for j:= 1 to n+1 do
mprodback[i,j]:=0;
for i:= q to ((n-np)+1) do
sequencia[i]:= 0;
end;
{Eliminar os nos das tarefas ja explodidos}
for j:=1 to n+1 do
if mprodback[q,j] <> 0 then
mplano[q,j]:= 0;
{Buscar novomestre- elimina os menores limitante maiores que upbound1}
ver_plano_atual_5(n,q,sequencia,ind3,novomestre);
if ind3 = 'N' then
begin
{retroceder no plano}
q:= pred(q);
for i:= (q+1) to ((n-np)+1) do
```

```
for j:= 1 to n+1 do
mprodback[i,j]:=0;
end
else
begin
sequencia[q]:= trunc(novomestre);
if q = ((n-np)+1) then
begin
if upboundaux <= upbound1 then
begin
if upboundaux < upbound1 then
upbound1:= upboundaux;
imprima_sequencia_4(sequencia,upbound1);
end;
end;    {fim do if q = n}
q:= succ(q);
end;   {fim do else}
end;   {fim do while}
end;  {fim do else1}
end; {fim while}
end.                    {Fim Batjs8}

Unit nosfilhos_2;
Interface


{                              Tese de mestrado

                    FACULDADE DE ENGENHARIA QUÍMICA

          Departamento de Engenharia  de Sistemas Químicos - DESQ  UNICAMP

Descrição do Programa

Unidade geradora de nós filhos a partir de um nó mestre}

Uses maior_va_1,menor_va_3,crt;

{Definição de tipos}
type
vetorbat1 = array[1..bat1+1] of integer;
vetorbat2 = array[1..bat1+1] of real;
matbat1   = array[1..bat1+1,1..bat2] of real;
matbat2   = array[0..bat1-1,1..bat1+1] of real;

{Definição de variáveis}
var
q2,i2,iii2,ii2,j2,Ppos2,Pant2,Pos2,m2,n2          : integer;
max12,max22,soma2,minDj2                          : real;
```

```
rj2,vetor_min2                                  : vetorUn1;
c2                                              : matbat1;
vetaux12                                        : vetorUn2;
vbat12                                          : vetorUn1;
sequencia22                                     : vetorbat1;
dj2                                             : vetorUn1;
bj2                                             : vetorbat2;
MatU2                                           : matbat1;
Procedure Gerar_filhos(n2,m2,q,np2,produltimo2:integer;time2,time22:matbat1;
seqnp2,sequencia2:vetorbat1;var mplano2:matbat2);

Implementation
Procedure Gerar_filhos(n2,m2,q,np2,produltimo2:integer;time2,time22:matbat1;
seqnp2,sequencia2:vetorbat1;var mplano2:matbat2);

Procedure Armaz_Lin(Ppos2:integer; Var vetorbat3: vetorUn1);
var
j1 : Integer;
begin
for J1:= 1 to M2 Do
vetorbat3[J1]:= Time2[Ppos2,J1];
end;

Procedure Zerar_Lin(Ppos2: integer);
var
J: integer;
begin
for J:= 1 to M2 Do
time2[Ppos2,J]:=0.;
end;

Procedure Restituir_Lin(Ppos2: integer);
var
J : integer;
begin
for J:= 1 to M2 Do
time2[Ppos2,J]:= vbat12[J];
end;

{Procedimento para a calculo do trabalho remanescente em um dado Processador}
Procedure Calc_RJ(A: Matbat1;N,M:integer;Var RJ: VetorUn1);
var I,J : integer;
begin
for J:= 1 to M Do
begin
Rj[J]:= 0.;
for I:= 1 to N+1 Do
Rj[J]:= Rj[J] + A[I,J];
end;
```

```
end;
{FIM}

Procedure Calc_U(B: Matbat1;N,M:integer;Var MatU: Matbat1);
vr J,J1,K1,K2: integer;
begin
K1:= 0;
for J:= 1 to (M-1) Do
begin
for K2:= 1 to N+1 Do
begin
MatU[K2,J]:= 0.;
For J1:= 2 to M-K1 Do
MatU[K2,J]:= MatU[K2,J] + B[K2,J1+K1];
end;
K1:= K1 + 1;
end;
end;

{Procedimento para a calculo do trabalho minimo, remanescente, Umin,considerando os
Processadores J+1..M}
Procedure Calc_UMin(MatU:matbat1;P,N,M:Integer;Var Vetor_Min: VetorUn1);
var VetorU     : vetorUn1;
Min        : Real;
Pos,j1,i1  : Integer;
begin
for j1:= 1 to (m-1) Do
begin
for i1:= 1 to n+1 Do
vetorU[I1]:= matU[I1,J1];
busca1(vetorU,n,min);
vetor_Min[J1]:= min;
end;
end;
{FIM}

{Procedimento para a calculo dos Dj em cada Procecessdor}
Procedure Calc_DJ(D :Matbat1;RJ,Vetor_Min: VetorUn1;Var DJ: vetorUn1);
var J,I: integer;
begin
for J:= 1 to M2 Do
dJ[J]:= rJ[J] + d[Ppos2,J] + vetor_Min[J];
end;
{FIM}

{Procedimento para a calculo do Menor Limitante em para cada sequencia gerada}
Procedure max_dJ(Bj:vetorUn1;M: Integer;var Max:real);
var
I,k: integer;
```

```
begin
max:=bj[1];
for k:= 2 to m Do
If Bj[k] > max then
max:= bJ[k];
end;
{FIM}

Procedure min_bj(bJ:vetorbat2;vbat2:vetorbat1;n,q:integer;var menor:real;var Pos:integer);
var
i: integer;
begin
If q = 1 then
begin
menor:= bJ[1];
Pos:=1;
for i:= 1 to n do
If Bj[I] < menor then
begin
menor:= bj[I];
Pos:= I;
end;
end
else
begin
for i:= n downto 1 do
If bj[I] <> 0 then
begin
menor:=Bj[I];
pos:= i;
end;
for i:= 1 to N do
If vbat2[I] <> I then
If bj[i] < menor then
begin
menor:= bj[I];
pos:= i;
end;
end;
end;

Procedure Correcao_C2_pseudoproduto(q,produltimo2:integer;time2:matbat1;
sequencia2:vetorbat1;var c2:matbat1);
var i2,j2,pant2,ppos2        : integer;
c3                           : matbat1;
begin
for i2:= 1 to (n2+1) do
begin
soma2:= 0.;
```

```
for j2:=1 to m2 do
begin
c3[i2,j2]:= soma2 + time22[i2,j2];
soma2:=  c3[i2,j2];
end;
end;
for i2:= 1 to (q-1) do
begin
ppos2:= sequencia2[i2];
{Calculo do Tempo de execucao da sequencia parcial C2}
if i2 <> 1 then
begin
pant2:= sequencia2[i2-1];
for j2:= 1 tO m2 Do
begin
if j2 = 1 then
begin
vetaux12[1]:= 0.;
vetaux12[2]:= c3[pant2,j2];
end
else
begin
vetaux12[1]:= c3[ppos2,j2-1];
vetaux12[2]:= c3[pant2,j2];
end;
Busca(vetaux12,2,Max22);
c3[ppos2,J2]:= max22 + time22[ppos2,j2];
end;
end;        {fim if i1}
end;       {fim i2}
for j2:= 1 to m2 do
c2[n2+1,j2]:= c3[produltimo2,j2];
ed;

Procedure Deter_tempo_Seqparcial(q:integer;time2:matbat1;sequencia2:vetorbat1;
var c2:matbat1);
var i2,j2,pant2,ppos2          : integer;
begin
for i2:= 1 to (n2+1) do
begin
soma2:= 0.;
for j2:=1 to m2 do
begin
c2[i2,j2]:= soma2 + time2[i2,j2];
soma2:=  c2[i2,j2];
end;
end;
for i2:= 1 to (q-1) do
begin
```

```
ppos2:= sequencia2[i2];
{Calculo do Tempo de execucao da sequencia parcial C2}
if i2 <> 1 then
begin
pant2:= sequencia2[i2-1];
for j2:= 1 tO m2 Do
begin
if j2 = 1 then
begin
vetaux12[1]:= 0.;
vetaux12[2]:= c2[pant2,j2];
end
else
begin
vetaux12[1]:= c2[ppos2,j2-1];
vetaux12[2]:= c2[pant2,j2];
end;
Busca(vetaux12,2,Max22);
c2[ppos2,J2]:= max22 + time2[ppos2,j2];
end;
end;        {fim if i1}
Zerar_Lin(Ppos2);
end;      {fim i2}
end;

BEGIN                    {Inicio nosfilhos};
{Inicializar variaveis}
for I2:=1 to m2 Do
vetor_Min2[I2]:=0;
pant2:=0;
for i2:= 1 to bat1+1 do
for j2:= 1 to bat2 do
c2[i2,j2]:= 0;
for i2:= 1 to bat1+1 do
for j2:= 1 to bat2 do
matu2[i2,j2]:=0;
for i2:= 1 to bat1+1 do
vetor_min2[i2]:= 0;
for i2:= 1 to bat1+1 do
bj2[i2]:= 0;
for i2:= 1 to bat1+1 do
rj2[i2]:=0;
for q2:= q to q do
begin
{Determinacao do tempo de execucao da sequencia parcial}
Deter_tempo_Seqparcial(q,time2,sequencia2,c2);
for ppos2:= 1 to n2+1 do    {se sequencia ja tem 1-2 nao faz}
begin
for i2:= 1 to q do
```

```
if (ppos2 = sequencia2[I2]) then
pant2:= sequencia2[I2];
if time2[ppos2,1] = 0 then {se a tarefa pertencer a seqnp nao fazer}
pant2:= ppos2;
If pant2 <> ppos2 Then
begin
{se ppos2 for o pseudo-produto corrigir c2}
if ppos2 = n2+1 then
begin
{montar sequencia}
for i2:= 1 to ((n2-np2)+1) do
sequencia22[i2]:= sequencia2[i2];
for i2:= 1 to ((n2-np2)+1) do
begin
if sequencia2[i2] = 0 then
begin
iii2:=i2;
for ii2:=1 to np2 do
begin
sequencia22[iii2]:= seqnp2[ii2];
iii2:= iii2 + 1;
end;
i2:=((n2-np2)+1);
end;
end;
correcao_c2_pseudoproduto(q+np2,produltimo2,time22,sequencia22,c2);
end;
{Calculo do Tempo de Completacao C}
if (q <> 1) and (ppos2 <> n2+1) then
begin
pant2:= sequencia2[q2-1];
for j2:= 1 to m2 Do
begin
if j2 = 1 Then
begin
vetaux12[1]:= 0.;
vetaux12[2]:= C2[pant2,J2];
end
else
begin
vetaux12[1]:= C2[Ppos2,J2-1] ;
vetaux12[2]:= C2[Pant2,J2];
end;
busca(vetaux12,2,max22);
c2[Ppos2,J2]:= max22 + time2[ppos2,j2];
end;
end;
{rmazenando Linha}
Armaz_lin(ppos2,vbat12);
```

```
{Zerando Linha}
Zerar_Lin(ppos2);
{Chamando Calc_RJ}
Calc_RJ(Time2,N2,M2,RJ2);
{hamando Calc_U}
Calc_U(Time2,N2,M2,MatU2);
{Restituindo linha}
Restituir_Lin(Ppos2);
{Chamando Calc_UMin}
Calc_Umin(MatU2,Ppos2,N2,M2,Vetor_Min2);
{Calculando os DJ}
Calc_DJ(C2,RJ2,Vetor_Min2,DJ2);
{Calculando Maximo dos DJ}
Max_DJ(DJ2,M2,Max12);
Bj2[Ppos2]:= Max12;
end              {Fim IF}
end;                   {Fim_Ppos2}
end;                       {Fim de Q}
for j2:= 1 to n2+1 do
mplano2[q,j2]:= bj2[j2];
end;
end.                       {fim nos nosfilhos}
```

REFERÊNCIAS BIBLIOGRÁFICAS CONSULTADAS

1. Baker, K. R., Introduction to Sequencing and Scheduling, John Wiley, 1974.
2. Baker, K. R., "A Comparative Study of Flow Shop Algorithms", Operations Research, 23(1), pp 62, 1975.
3. Egli, U. M., Rippin, D. W. T., "Short-term Scheduling for Multiproduct Batch Chemical Plants", Comp. Chem. Engng., 10(4), pp 303, 1986.
4. Garey et. al., " The Complexity of The Flowshop and Jobshop Scheduling" Math. Opns. Res.,1,pp 117,1976.
5. Graves,S.C., " A Review of Production Scheduling", Oper. Res., 29(4), pp 646,1981.
6. Ku,H., Rajagopalan,D., Karimi,I., "Scheduling in Batch Process", Chem. Eng. Prog., 83(8), pp 35,1987.
7. Mah, R. S. H., Chemical Process Structures and Information Flows, Butterworth Publishers, 1990.
8. Nilson,N. J., Problem-Solving Methods in Artificial Intelligence, McGraw-Hill BackCompany, 1971.
9. Panwalkarer, S. S., Iskaqnder, W., A Survey of Scheduling Rules, Oper.Res.,25(1), pp 45, 1977.
10. Parakrama R., The Chem. Eng., p. 24 (1985).
11. Rajagopalan, D., Karimi, I. A., "Completion times in Serial Mixed-Storage Multiproduct Processes with Transfer and Set-up Times, Comp. Chem. Engng, 13(1/2), pp 175-186,1989.
12. Reklaitis, G. V., "Review of Scheduling of Process Operations, AIche Symposium Series, pp 119, 1982.
13. Rodrigues, M. T. M. , Seqüenciamento e Alocação de Operações em Flow Shops na Indústria Química: Uma Abordagem de Busca Orientada por Restrições, Dissertação de Doutourado, UNICAMP, SP,1992.
14. Shah, N., Pantelides, C. C., Optimal Long-Term Campaign Planning and Design of Batch Operations, Ind. Eng. Chem. Res., 1991.

REFERÊNCIAS BIBLIOGRÁFICAS CITADAS

1. Graham, R. L.,E. L. Lawler; J. K. Lenstha and A.H.G. Rinnooy Kan, Optimization and Approximation in Deterministic Sequencing and Scheduling: A Survey, Ann, Discret Math., 5,287-280,1979.

2. Ignall, E., and L. Scharage, Application of teh Branch-and-Bound Technique to Some Flow-Shop Scheduling Problems, Opns. Res., 13, 400-412.

3. Jackson, J. R., Scheduling a Production Line to Minimize Maximum Tardiness, Research Report 43, Management Sciences Research Project, UCLA.

4. MacNaughton, R., Scheduling with Deadlines and Loss Funtions, Mgmt. Sci 6,1-12, 1959.

5. Mauderli, A.,Rippin,D.W.T., Production Planning and Scheduling for Multipurpose Batch Chemical Plants, Computers and Chemical Engineering,3,199-206,1979.

6. Mauderli,A., Rippin,D.W.T., Scheduling Production in Multi-Purpose Batch Plants: the Batchman Program, Chemical Engineering Progress,76(4),37-45,1980.

7. Little, J.D.C., MUrty, K.G. Sweeny, D.W., and Karel. C., An Algorithm for The Traveling Salesman problem, operations Resarch, vol. 11,No. 6(November, 1963).