UNIVERSIDADE ESTADUAL DE CAMPINAS

FACULDADE DE ENGENHARIA QUÍMICA

DEPARTAMENTO DE ENGENHARIA DE SISTEMAS QUÍMICOS

INTERFERÊNCIA LÓGICA EXTERNA EM PROBLEMAS DE PROGRAMAÇÃO DE PRODUÇÃO DE SISTEMAS FLEXÍVEIS

Por Edilson de Jesus Santos

Orientadora: Dra. Maria Teresa Moreira Rodrigues

Tese apresentada à Faculdade de Engenharia Química - UNICAMP como parte dos requisitos exigidos para a obtenção do título de DOUTOR em ENGENHARIA.

março/98

Campinas - SP

Brasil

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

Sa59i

Santos, Edilson de Jesus
Interferência lógica externa em problemas de programação de produção de sistemas flexíveis. / Edilson de Jesus Santos.--Campinas, SP: [s.n.], 1998.

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

1. Sistemas flexíveis de fabricação. 2. Planejamento da produção. 3. Controle da produção. 4. Programação (Matemática). 5. Inferência (Lógica). I. Rodrigues, Maria Teresa Moreira. II. Universidade Estadual de Campinas. Faculdade de Engenharia Química . III. Título.

Tese de Doutorado defendida e aprovada em 27 de março de 1998 pela banca examinadora constituída pelos professores doutores:

Profª Drª Maria Teresa Moreira Rodrigues
Orientadora

Prof. Dr. Paulo Morelato França

Prof. Dr. Márcio Luiz de Andrade Neto

Prof. Dr. João Alexandre Ferreira da Rocha Pereira

Profª Drª Ana Maria Frattini Fileti

Esta versão corresponde à redação final da Tese de Doutorado em Engenharia Química defendida pelo Engenheiro Edilson de Jesus Santos e aprovada pela Comissão julgadora em 27 de março de 1998.

Profª. Dra. Maria Teresa Moreira Rodrigues
Orientadora

Bénissez le Seigneur en tous temps,
Au matin bénissez-le, le soir bénissez-le,
Dieu seul est bon et grand...

Dieu seul est justice et sagesse.

(da obra "SOLOMON")
Haendel, Georg Friedrich

AGRADECIMENTOS

A Deus pela minha objetividade

A minha mãe, Maria Anatilde de J. Santos, pelo apoio e compreensão fornecidos nesta longa jornada de estudo.

A todos os meus Irmãos

A Profª Dra. Maria Teresa M. Rodrigues pela orientação consciente e pelas os esclarecimentos fornecidos em nossas longas reuniões, além dos aconselhamentos de incentivos profissionais que fortaleceram a nossa amizade.

Ao Prof º Dr. Lluis Gimeno Latre pelas contribuições dadas a este trabalho.

Ao Prof º Dr. João Alexandre R. F. Pereira pela amizade e apoio concedidos durante a minha permanência na Faculdade Engenharia Química.

Ao amigo da Universidade Federal de Sergipe, Profº Josias Máximo de Jesus pelos diálogos que realimentaram minha convicção para o término deste trabalho.

A minha singular amiga Profª Lenalda Dias dos Santos, pelo incentivo e aconselhamentos oferecidos.

Ao contemporâneo de Mestrado e Doutorado Sr. Marcone Lopes pelos esclarencimentos de cunho computacionais cedidos ao longo deste trabalho e acima de tudo pela amizade.

Aos conterrâneos e contemporâneos de Mestrado e Doutorado: Sra. Gisélia Cardoso, Sra Sônia Maria, Sr. Pedro Leite de Santana e Sr. Roberto Rodrigues de Souza pela convivência salutar praticada durante essa longa jornada.

A contemporânea de Doutorado Sra. Marta Neto pela atenção, discussões de cunho técnico e acima de tudo pela amizade.

Ao conterrâneo Sr. Cesar Guimarães pelo tratamento singular a mim fornecido.

Aos conterrâneos Sr. Carlos Alexandre Garcia e Sra. Helenice Garcia pelos laços de amizade.

Ao Sr. Marco Fraga pelas discussões de cunho filosófico e sociais e acima de tudo pela amizade.

Ao Sr Ricardo Belchior e Sra. Mila Jorge Arruda Belchior Torres pelo tratamento singular a mim dado e acima de tudo pela amizade

Ao amigo Sr José Vicente d'Angelo pelo tratamento a mim concedido e pelas tardes de sexta-feira, quando colocávamos em prática nosso excelente basquete, juntamente com o Sr. Luis Rodrigues e o Sr. Marcone Lopes.

Ao amigo Sr. Alexandre Denes Arruda por todo o apoio a mim dado, pelas discussões de cunho humanístico e acima de tudo pela amizade.

Ao Sr. Marcos e Sra. Simone Morrone pelos laços de amizade formados nesses longos anos e pelas noites deliciosas de degustação de vinho.

Aos amigos de trabalho e dos cafés vespertinos: Sr. Marcone, Sra. Marta Neto, Sr Cesar, Sr. Frede, Sr. Ricardo, Sr. José Vicente, Sr Alexandre Denes Arruda, Sr. Luis

Rodrigues ("Luis Garcia"), Sr. Arlan, Sr. José Edson, Sr. Jurandir, Sr Reinaldo, Sra. Kelly, Sr. Moacir França, Sr Marcus.

A Sra. Rosa Maria Moretti pela amizade e pela atenção concedida durante a minha permanência na Faculdade de Engenharia Química.

A Sra Andréia pelo tratamento e esclarecimentos técnicos dados durante o desenvolvimento deste trabalho.

A Sra. Tereza pela amizade e atenção concedidos.

A CAPES, órgão de fomento desta pesquisa.

E a todos os que contribuíram de alguma forma para o término desta Tese.

RESUMO

Os sistemas de produção que operam em batelada na indústria química, em geral são projetados para a produção de diferentes produtos usando o mesmo conjunto de equipamentos. Consequentemente, é necessário adotar estratégias que levem ao estabelecimento de um plano de produção que atenda critérios relevantes para o problema de Planejamento e Programação da produção.

A definição de um plano de produção exige não só a determinação das quantidades a serem produzidas mas também do programa temporal de produção. Assim sendo, na definição do plano de produção devem ser considerados todos os fatores importantes para o estabelecimento deste plano, em particular o compartilhamento de recursos tais como equipamentos, mão-de-obra, energia elétrica dentre outros.

Os problemas de programação de produção ligados a esse tipo de processamento são considerados pertencentes à classe NP (Não Polinomial). Quando estão envolvidos recursos compartilhados capacitados, isto é, recursos que podem ser utilizados simultaneamente durante a produção em vários equipamentos tal como vapor, o problema de programação da produção, além de NP-completo, é considerado computacionalmente difícil ("hard problem"). Portanto a busca de estratégias que venham diminuir a dificuldade de resolver tais problemas é objetivo constante de muitos trabalhos na literatura.

Neste trabalho será utilizado o modelo proposto por Kondili et. al. (1993), o qual recorre a uma representação discretizada do tempo, que tem se mostrado bem adaptada para a representação de problemas com limitação na oferta de recursos compartilhados. No entanto, a representação do tempo discretizado exige um número bastante elevado de variáveis binárias, comprometendo a dimensão do problema e portanto, seu tempo de solução. Para reduzir a dificuldade de solução do problema de programação de produção usando o modelo discretizado de representação do tempo, é proposta uma estratégia de interferência lógica externa sobre as variáveis binárias de alocação presentes no modelo utilizado.

A estratégia proposta é implementada usando o pacote OSL ("Optimization System Library") da IBM, pois a sua estrutura permite a interferência externa do usuário durante a solução dos problemas de programação inteira-mista.

ABSTRACT

The chemical batch facilities are essentially multiproduct or multipurpose. Consequentelly, the definition of a production plan and scheduling has to take into account many different aspects in order to satisfy relevant problem constraints as shared resources, production routes etc.

The planning and scheduling problems are NP-complete problems. When capacited shared resources, as vapour, electricity, are envolved, it is also considered as a hard problem.

In this work is utilized the model proposed by Kondili et al. (1993), based on the discret time representation, that seems well suited for problems with shared resources. Nevertherless, this time representation demands a high number of discrete variables and, as a main consequence, the solution time rises even for small problems. In the order to reduce the solution hardness, it is proposed an external logical inference on the allocation binary variables, during the search procedure.

The proposed strategy is implemented using the OSL package that allows external interference through the "user exit subroutines".

SUMÁRIO

INTRODUÇÃO

A identificação e solução de problemas ligados à manufatura de produtos é fato comprovado em qualquer país que busca o seu desenvolvimento econômico. Mas a prática nos mostra que os países que alcançaram esta tal sonhada intitulação "País economicamente desenvolvido" não a conseguiram sem antes controlar os altos índices de inflação, alcançando assim a estabilidade econômica, que sem dúvida parece ser um limiar para se conseguir um avanço no desenvolvimento não só econômico mas também social.

Em períodos de baixos índices inflacionários, onde a estipulação de novos preços aos produtos para cobrir eventuais aumentos de custo de produção é bastante restrita ou porque não dizer quase impraticável. Com isso a modernização da máquina produtiva, a aplicação de otimização de processos, a adoção de metodologias para aumento da produtividade, controle de processos e planejamento de produção são essenciais para qualquer indústria que busca sua sobrevivência em um mercado, que em tempos de "globalização", se torna cada vez mais competitivo.

O panorama econômico mundial concatenado às perspectivas de um mercado cada vez mais exigente justificam a elaboração e implementação de sistemas que executem estratégias eficientes de controle e programação de produção de sistemas de manufatura, visando com isso alcançar as metas estipuladas pela organização, de modo menos oneroso. Tais sistemas não devem levar em conta somente dados operacionais da planta mas principalmente parâmetros de mercado, assim como flutuações na demanda, ocasionadas, por exemplo, devido à mudança de hábito dos consumidores.

Nos últimos anos a atividade de Programação de Produção tem recebido bastante atenção por parte de diversos setores industriais, principalmente os da indústria de processamento químico, onde os produtos possuem um alto valor agregado. Isto faz com que a análise e estudo de metodologias que diminuam a complexidade de solução destes problemas seja objetivo constante dos pesquisadores que atuam nesta área.

Os modelos que utilizam Programação Inteira-Mista surgiram de várias situações práticas (Nemhauser, 1988). A solução desses modelos por pacotes comerciais muitas vezes, a depender do problema, pode exigir alto custo de solução (alto tempo computacional).

Mesmo nos casos em que o espaço de soluções de problemas de programação de produção seja limitado pelas sequências de permutação (sequências em que a ordem de processamento das operações de cada tarefa é a mesma em todos os equipamentos), estes problemas são da classe NP (Não-Polinomial) (Garey et al., 1976). Uma alternativa promissora para contornar este fato é proposta por Raman e Grossmann (1994), que consiste em explorar a estrutura lógica muitas vezes presente em determinados problemas.

A utilização da programação matemática é bastante difundida em diversas áreas para o tratamento de problemas de natureza diversa. Quando se parte para a formulação matemática de um determinado problema, diferentes modelos podem surgir, principalmente quando esta formulação exige presença de variáveis inteiras. Obviamente que uma determinada dificuldade de solução estará associada a cada um desses modelos. Neste fato reside um dos principais objetivos de pesquisadores, que é a obtenção de modelos de baixo custo de solução, possibilitando assim a solução de problemas cada vez mais próximos dos problemas reais, que no caso particular da programação de produção, esta realidade é representada pelo elevado número de restrições envolvidas (as quais podem ser de formulação complexa) e/ou através do grande número de tarefas habilitadas aos equipamentos. Esta última característica é traduzida em uma elevação no número de variáveis binárias, fator determinante e incontestável para o aumento do custo computacional de solução dos problemas.

Além da programação matemática, técnicas de inteligência artificial e procedimentos heurísticos são também abordados na literatura. No entanto, estas utilizam o conhecimento heurístico para tomar decisões pertinentes ao problema. Uma das principais limitações do uso da abordagem de busca em árvore "branch and bound", por exemplo, é a dificuldade de obtenção de um limitante com comportamento monotônico que contemple todas as características de processamento. Este limitante, pela premissa do critério de "monoticidade", para a maioria dos problemas de programação de produção é difícil de ser formulado. Hoje, o uso desta técnica na solução está bastante fundamentada quando se visa minimizar o tempo total de execução das operações ("makespan") com restrições de produção relativamente simples, isto devido aos excelentes procedimentos heurísticos existentes para o cálculo do limitante em "makespan". Além desta limitação, o questionamento do uso de algoritmos que se utilizam da busca em árvore passa também por uma análise técnica, que é a extrema necessidade de gerenciamento de memória, principalmente devido à

manutenção da lista de nós ativos durante o processo de busca, este fato também deve ser considerado quando se parte para o uso de programação matemática. E quando se entra nesta análise, questões sobre recursos da linguagem de programação a ser utilizada, forma de implementação adotada pelo programador dentre outras, devem ser consideradas, o que leva a discussão para uma região de limites subjetivos.

Quando se utiliza a programação matemática em problemas de programação de produção, a representação da habilitação das operações nos equipamentos pode ser formulada matematicamente através da utilização de variáveis binárias de alocação ou de precedência. A solução dos modelos pode ser obtida através de pacotes que se utilizam da abordagem de Programação Linear Inteira-Mista (quando a formulação é linear, como é o caso dos problemas tratados neste trabalho), cuja estratégia consiste basicamente em uma busca em árvore conduzida por um limitante, o qual é avaliado por relaxações das restrições de integralidade (que são as variáveis definidas como binárias). Para tanto o sistema OSL (Optimization Sobroutine Library) já incorpora estratégias avançadas deste tipo de busca em árvore.

Segundo Raman e Grossmann (1994), um dos fatores que vêm motivando a utilização da programação matemática é a crescente disponibilidade de pacotes direcionados à solução de modelos matemáticos: LINDO ("**L**inear **IN**teractive and **D**iscrete **O**ptimizer"), OSL (**O**ptimization **S**ubroutine **L**ibrary"), MINOS (**M**ixed **IN**teger **O**ptimization **S**ubroutine), e em especial a Interface GAMS (**G**eneral **A**lgebraic **M**odeling **S**ystem), a qual incorpora alguns dos resolvedores anteriormente citados e que tem ajudado à geração de modelos (discretização das equações), que é uma das partes mais árdua da modelagem.

Os pacotes matemáticos que cultivam e incentivam cada vez mais a solução de problemas mais laboriosos podem ser vistos no que diz respeito à qualidade da solução como eficientes. No entanto, entre a solução encontrada e o caminho percorrido até chegar a esta solução, existe uma diferença muitas vezes inexplorada por inflexibilidade desses pacotes (o acesso por parte dos usuários aos procedimentos utilizados é impossibilitada). Quanto aos pacotes que se destinam à solução de modelos via programação linear inteira-mista podemos dizer de modo amplo que os mesmos adotam um procedimento "híbrido", melhor dizendo, que estes pacotes unem dois mundos, o primeiro mundo é o da pesquisa operacional representada pela busca em árvore "branch and bound" que garante a integralidade das variáveis inteiras e o

segundo corresponde ao mundo matemático, representado pela álgebra linear, a qual formaliza o método SIMPLEX, que é utilizado para o cálculo do limitante da função objetivo, o qual controlará os nós da árvore de busca. Neste ponto, vale ressaltar uma indiscutível vantagem da programação matemática frente ao uso de algoritmos de busca em árvore, que é a facilidade de avaliar o limitante nos nós seja qual for a função de custo.

A combinação destes dois mundos possibilita o uso de interferências lógicas no tratamento de certos problemas. Seguindo esta abordagem, Raman e Grossmann (1991) mostram como relações lógicas são obtidas e aplicadas em problemas de síntese de redes. Posteriormente, Raman e Grossmann (1992) mostram como restrições lógicas, agora na forma de inequações, podem melhorar o custo de solução de problemas de síntese. Hooker (1994) mostra que a idéia básica da utilização desse conhecimento lógico ("logic cuts") é a substituição de elementos essenciais e cruciais de otimização por elementos de natureza lógica.

Quando se identifica a existência de relação lógica entre variáveis binárias, dois procedimentos poderiam ser adotados, o primeiro seria inserir no modelo em sua forma matemática a respectiva restrição lógica na tentativa de melhorar o "gap" de solução do modelo, ocasionando um aumento no número de restrições no modelo e consequentemente podendo aumentar o custo de solução. O segundo procedimento é utilizar este conhecimento lógico durante a solução do problema, mas para isto é necessário que o pacote matemático permita tais interferências. Neste ponto, o projeto de implementação do OSL permite que o usuário em determinados pontos da busca em árvore possa interferir nos procedimentos padrões das subrotinas do referido pacote. Centrado nesta característica, este trabalho explora interferências lógicas envolvendo características inerentes aos problemas aqui tratados, na tentativa de diminuição do custo de solução do modelo aqui abordado.

Quando se parte para o tratamento de problemas de planejamento e programação de produção deve-se ter sempre em mente que o tempo de solução desses aumenta exponencialmente com o número de variáveis binárias existentes no modelo, logo é fundamental a adoção de estratégias que venham diminuir o tempo computacional de solução desses problemas, sempre tendo em vista o outro lado que permeia todo o problema de otimização, que é a qualidade da solução obtida. A pesquisa de tais estratégias favorecerá a solução de problemas com um maior nível de

detalhamento de processo, ou seja, com maior número de operações e equipamentos. Neste fato está sedimentada a abordagem deste trabalho que visa combinar o conhecimento lógico aos elementos da programação matemática, através da interferência lógica externa.

CAPÍTULO 2

PROGRAMAÇÃO DE PRODUÇÃO EM SISTEMAS FLEXÍVEIS DE PRODUÇÃO

2.1. Introdução

Neste capítulo, são estabelecidos alguns conceitos relevantes para o estabelecimento do problema de Programação de Produção em sistemas flexíveis de manufatura, além de mostrar algumas relações existentes entre este problema e outros existentes no cenário industrial. Dentre estes problemas estão:

- O Problema de Elaboração de Projeto de Unidades Químicas (síntese de idéias e informações que procura a satisfação de uma necessidade de mercado, visando sempre minimizar os riscos envolvidos em um determinado investimento).
- Gerenciamento de Projetos: Planejamento e Controle de todas as atividades determinadas na fase de elaboração de projeto.
- O Problema Planejamento de Projetos (condução do cronograma das atividades de implantação de um projeto considerado viável).

Dentre os problemas citados acima, alguns apresentam características semelhantes, assim como o problema de gerenciamento de projetos, que objetiva a conclusão de um determinado conjunto de atividades em tempo predeterminado, e o problema de programação de produção que, em certos problemas, visa a minimização do tempo total de operações. Além disso, problemas de projeto de plantas batelada são sensivelmente afetados pela programação de produção das operações envolvidas (Birewar e Grossmann, 1990). Nesse trabalho, os autores não só levam em consideração os aspectos de tamanho de equipamentos necessários para a obtenção das demandas de produtos, como também há considerações de síntese de processo e de programação de produção. É sabido que o programa de manufatura a ser utilizado por um determinado setor de produção determina uma maior ou menor ociosidade dos equipamentos durante a manufatura dos produtos, ou seja, se critérios de programação de produção forem levados em consideração na fase de projeto, as dimensões dos mesmos se ajustam ao programa de produção ótimo.

2.2. Elaboração de Projeto de Unidades Químicas

2.2.1. Introdução

A Análise e Elaboração de Projetos pode ser conceituada como uma ferramenta técnico-administrativa e de avaliação econômica (Woille e Washington, 1996). O projeto e o planejamento de produção representam procedimentos lógicos e racionais

que substituem o comportamento intuitivo e empírico. Além disso, a elaboração e análise de projeto podem ser vistas como um mecanismo de avaliação econômica das conseqüências de uma tomada de decisão por parte do investidor, uma vez que as tomadas de decisão afetam a rentabilidade dos recursos próprios empregados.

O problema de projeto está inserido em diversas áreas. A depender do setor, social ou privado, o projeto, apesar de ser uma ferramenta que diminui o risco de um determinado investimento, é visto por prismas diferentes. Do ponto de vista de interesse social, considera-se o conjunto de informações sistematicamente ordenadas, que nos permite estimar custos e benefícios sociais de um determinado investimento. Já na visão da iniciativa privada, o projeto é um instrumento que permite avaliar as vantagens relativas de um determinado uso de recursos, face às diversas alternativas de investimento existentes. Seja qual for o interesse envolvido, pode-se dizer que as fases de elaboração e análise de projetos são essenciais para a diminuição do risco de um determinado investimento.

A elaboração diz respeito à fase de busca de justificativas e dos recursos necessários para por em andamento um determinado projeto. A Análise, por sua vez, corresponde à fase de avaliação das diversas alternativas existentes.

Quando se estudam os modos possíveis de alcançar o objetivo do projeto o projetista estará limitado por muitos fatores, os quais reduzirão as soluções possíveis. É perfeitamente normal que se tenham várias soluções possíveis para o problema, soluções estas que levam ao mesmo projeto ótimo. Esta possibilidade está intrinsecamente ligada às características do problema.

As limitações sobre as soluções possíveis de um problema têm muitas origens, alguns condicionantes são invariáveis, como aqueles que resultam das leis físicas e de regulamentos governamentais. Outros são menos rígidos e suscetíveis de acomodação pelo projetista. Existem também os condicionantes externos aos projetos, como por exemplo os provenientes da política econômica do Governo (política de importação, instabilidade da economia) que devem ser levados em consideração durante a preparação do projeto.

O tempo é uma restrição que muitas vezes limita o número de projetos alternativos que podem ser praticados, pois geralmente se tem um horizonte de tempo pré-determinado para se completar um determinado projeto.

Uma classificação de projetos pode ser dada pela sua finalidade, conforme o objetivo: gerar novos meios de produção, ampliar a capacidade e aumentar

produtividade de meios de produção já existentes. No que diz respeito aos projetos industriais, os mesmos podem ser classificados:

- Implantação: criação de novos centros produtores;
- Expansão ou ampliação: integração de novos centros produtores aos existentes (uma nova linha de produção);
- Modernização: Substituição de unidades de equipamentos ou centros produtivos considerados obsoletos, sem alterar portanto a capacidade de instalação.
- Relocalização: Mudança na localização do projeto (programa de modernização ou em decorrência de alterações nos preços dos fatores).

2.2.2.Etapas do Projeto

As etapas principais de um projeto são:

- Especificação do Projeto: Segundo Coulson e Richardson (1983) o projeto procura satisfazer ao máximo os clientes.
- Aquisição de dados, propriedades físicas e Metodologias de projeto: Nesta etapa são levantadas as propriedades físicas, a eficiência de equipamentos. Os condicionalismos do projeto, especialmente os de origem externa, devem ser identificados na fase inicial do processo.
- Geração de soluções possíveis: Geração de alternativas para se atingir um objetivo.
- Análise de viabilidade: Nesta fase, critérios econômicos são utilizados para a escolha do projeto maior rentabilidade (análise de custos e benefícios).
- Execução do Projeto Final

As etapas de desenvolvimento de um projeto, desde a identificação inicial do objetivo até ao projeto final, estão apresentadas na figura 2.1.

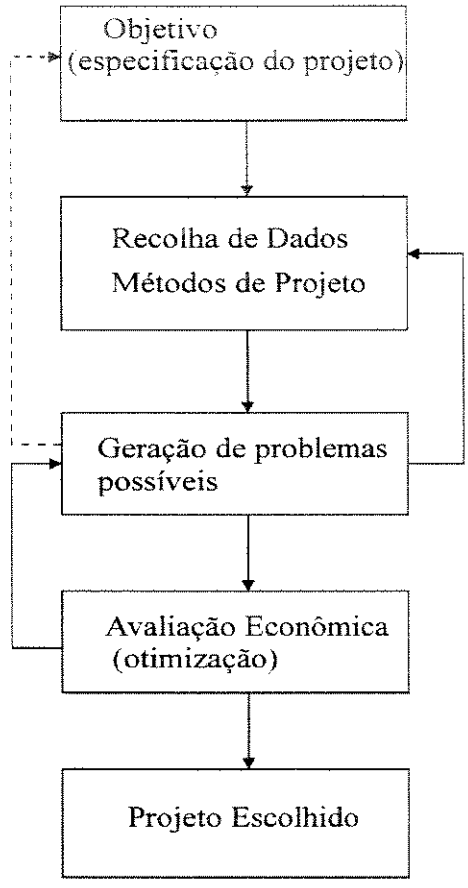


Figura 2.1: O processo de projeto

Os diferentes aspectos de um projeto devem estar perfeitamente harmonizados entre si, formando um conjunto homogêneo e coerente. Além disso, os estudos que permeiam os aspectos podem ser realizados com níveis de detalhamento diferentes. Este nível de aprofundamento que deve ser considerado dependerá da relação entre custo adicional e o benefício das novas investigações. Assim, antes de pormenorizar o estudo de um determinado aspecto do projeto, deve-se verificar a importância relativa que um determinado aspecto poderá ter em relação aos fatores decisivos para o sucesso do projeto, além de examinar se os dados disponíveis possuem a fidedignidade compatível com a precisão pretendida.

2.3. Projeto de Plantas Químicas

2.3.1. Introdução

No que se refere aos projetos de novas instalações industriais, uma pergunta que imediatamente surge é: a produção dos produtos será contínua ou descontínua? Para

fornecer a resposta a esta indagação, alguns conceitos mais comuns no cenário da indústria química e aspectos econômicos (como análise de mercado) são necessários. Uma vez tomada a decisão pelo modo de operação da planta, tanto as fases de gerenciamento do projeto e de programação da produção são afetados.

No que se refere ao processamento em sistemas flexíveis estão sempre presentes três componentes:

- As condições que devem ser oferecidas para os diferentes produtos (insumos, recursos compartilhados);
- As operações a serem desenvolvidas e as rotas para a produção dos diferentes produtos a partir das matérias-primas (receita de produção);
- Os equipamentos nos quais as tarefas de processo devem ser executadas.

Qualquer unidade de processamento seja contínua ou não contínua é dimensionada para uma determinada capacidade de produção. A figura 2.2 mostra uma unidade típica de processamento químico.

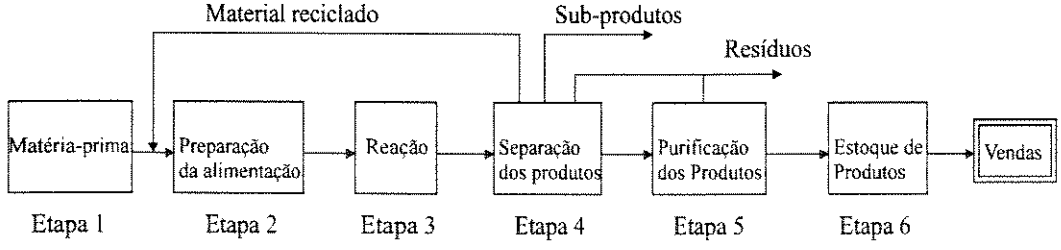


Figura 2.2: Planta de Processamento Químico

No caso de unidades dedicadas, embora a atividade de projeto seja extensa e complexa, ela é, em princípio, mais simples de ser executada do que em plantas que operam em modo descontínuo (sistemas flexíveis), visto que o projeto nestas últimas envolve um maior grau de liberdade (maior número de variáveis envolvidas). Assim, a atividade de projeto de unidades flexíveis engloba não só a definição de fluxos de massa e energia pertinentes ao processo, mas também a sincronização das operações em cada equipamento e sua interação com outras unidades na seqüência de produção.

Até recentemente a formulação do problema de projeto de plantas flexíveis era desenvolvida sem considerar os problemas de programação de produção (Sparrow et al, 1975). A partir deste trabalho foram desenvolvidos vários algoritmos que

incorporam problemas de programação de produção, no entanto sem considerar características relacionadas à política de armazenagem, compartilhamento de recursos, tempos de estabelecimento.

Quanto à literatura que aborda problemas de projeto industriais, parece ser consenso que a maior parcela de investimento total de um projeto é representada pelos custos de aquisição de equipamentos, com isso é comum adotar como função objetivo nos problemas de plantas a minimização das dimensões dos equipamentos. Woiler e Washington (1996) mostram dados dos custos envolvidos na instalação de uma planta de celulose (produção contínua). A tabela 2.1 mostra estes dados.

Tabela 2.1: Elementos que agregam custos na fase de projeto

Constituinte	Percentagem do Investimento Total (%)
Geração de dados e Viabilidade	0.3 a 1
Engenharia e Imprevistos	8 a 12
Supervisão e gastos gerais da construção	9 a 12
Construção da fábrica	
• Preparação do terreno e Edifícios	12 a 20
• Equipamentos e instalação	45 a 52
Gastos pré-operacionais (arranque)	3 a 7
Juros durante a construção	4 a 6
Capital de giro	5 a 13

O custo de compra e de instalação dos equipamentos equivalem em média a 48.5% do investimento de implantação da fábrica, justificando assim a função de custo utilizada nos modelos de projetos que se encontram na literatura. Mas estes dados não impedem que outras considerações que pesam no custo de investimento de implantação de um novo projeto sejam consideradas.

O critério econômico freqüentemente usado no projeto de plantas químicas é a minimização do custo de aquisição dos equipamentos (justificado pela tabela 2.1), o qual é representado matematicamente por uma função de potência da capacidade (volume, área, por exemplo) dos mesmos. A função de custo (C) para implantação de projeto é dada pela expressão:

$$C = \sum_{E=1}^{NE} m_E a_E (V_E)^{\alpha_E} \qquad (2.1)$$

Neste caso foi considerada uma planta que opera em modo batelada com NE estágios, cada estágio E consiste de m_E unidades idênticas dispostas paralelamente. Cada equipamento possui um volume característico V_E. O coeficiente a_E e o expoente α_E, são constantes positivas. O objetivo do problema é então encontrar os volumes de equipamentos e o número de equipamentos operando em paralelo para satisfazer a quantidade de produto i (Q_i) exigida pelo mercado.

Coulson e Richardson (1983) apresentam uma série de valores de α para vários equipamentos. Alguns desses são apresentados na tabela 2.2.

Tabela 2.2: Valores típicos das constantes a e α de equipamentos

Equipamento	Dimensão de tamanho S	Intervalo de tamamhos	Constante "a"	Expoente "α"	Observação
Agitadores de hélice	potência de acionamento	5-25 KW	400	0.50	aço macio
Centrífuga-cesto horizontal	dia,m	0.5-1.0	16000	1.3	-
Centrífuga-cesto vertical	dia,m	0.5-1.0	16000	1.0	-
Filtro de placas	área,m^2	5-50	1000	0.60	ferro fundido
Filtro de tambor	área,m^2	1-10	6000	0.60	aço macio
Filtro a vácuo	área,m^2	1-10	6000	0.60	aço macio
Evaporadores de tubo vertical	área, m^2	10-1000	4000	0.53	aço macio
Secadores rotativo	área, m^2	5-30	6000	0.45	aço macio
Tanques armazenagem	capacidade, m^3	50-8000	650	0.65	aço macio
Tanques Processamento	capacidade, m^3	1-50	500	0.59	pressão atmosférica aço macio x

Além de equipamentos que operam de modo batelada, existem os chamados equipamentos semicontínuos (bombas), as quais exercem a função de carregar e descarregar os equipamentos batelada. Assim sendo, a função objetivo do problema anterior poderia incluir o dimensionamento de equipamentos semicontínuos. Considerando o estágio P como tendo np equipamentos semicontínuos, os quais possuem a mesma vazão de processamento SC_P e sabendo-se que o coeficiente b_P e expoente β_P são constantes positivas, a função objetivo de projeto torna-se:

$$C = \sum_{E=1}^{NE} m_E a_E (V_E)^{\alpha_E} + \sum_{P=1}^{NP} n_P b_P (SC_P)^{\beta_P} \qquad (2.2)$$

Nos problemas de projetos apresentados na literatura, o segundo termo da equação é negligenciado. Esta aproximação também se utiliza, a depender do problema, na programação de produção.

Aries e Newton (1955) propõem o uso de α_j e β_j iguais a 0.6 (regra dos seis décimos). Os valores destes coeficientes dependem do estado do material que a indústria vai produzir (sólido, liquido, sólido-líquido). No entanto, é interessante a partir de séries históricas que forneçam o preço de compra e custo de instalação de equipamentos em função dos seus tamanhos, e com isto avaliar os expoentes α_j e β_j por método de ajuste de curvas.

2.4. Gerenciamento de Projetos

2.4.1. Introdução

Quando se toma a decisão de investir em um determinado projeto, é necessário que o cronograma de atividades estipulado pela gerência do mesmo seja controlado, garantindo assim o cumprimento da data de término. Atrasos, por exemplo, nas atividades de implantação de uma planta muitas vezes acarretam prejuízos monetários, isto devido aos juros que incidem sobre o capital investido durante a implantação, o qual muitas vezes vem de empréstimos bancários, além de comprometer o tempo em que a nova indústria começa a gerar receita. Procurando a minimização desses custos, técnicas para o controle de atividades ligadas a construção (Problema de Planejamento de Projetos) devem ser desenvolvidas. Uma vez encontrada uma alternativa de projeto que passe pela fase de análise de viabilidade e que seja a melhor alternativa para o

investidor, parte-se então para a fase implantação do projeto, neste cenário entram técnicas CPM (Critical Path Method)/PERT (Project Evaluation Review Techinique), as quais apresentam uma formalização bem semelhante à da programação de produção.

As técnicas de caminho críticos (CPM,PERT) são utilizadas em diversas áreas para o acompanhamento de projetos. As fases destas técnicas são:

- Planejamento: Estabelecimento e predeterminação das operações que devem estar envolvidas na execução do projeto em análise.
- Programação: Nesta fase é estabelecido o tempo de execução de cada uma das atividades estratificadas na fase de planejamento. Como os tempos são cruciais para o término de projeto é necessária cautela na estimativa desses tempos das atividades envolvidas.
- Controle: Nesta fase a execução das operações é acompanhada, principalmente aquelas que fazem parte do caminho crítico, pois se alguma atrasar o término do projeto estará comprometido. Estas técnicas não têm como corrigir eventuais atrasos, mas servem como ferramenta que mostra quais as atividades ou operações que não podem ser relaxadas, ou seja, devem ser executadas em regime de urgência para que o projeto seja executado o mais próximo possível do prazo estipulado.

As técnicas PERT/CPM para o gerenciamento de projetos são constituídas dos seguintes elementos:

- Programa: É um conjunto de operações ou atividades que concorrem para a realização de um determinado objetivo, de tal modo que se conhece o tempo de cada operação e as interconectividades de operações pertencentes à rede.
- Operação ou Atividade: É a execução de uma tarefa que consome tempo e recursos.
- Evento: Como uma operação que tem um ponto de início e de fim, estes pontos extremos de uma operação denominam-se eventos.

2.4.2. Estabelecimento de um Problema de Gerenciamento de Projeto

Como exemplo deste problema considere a tabela 2.3 , que estabelece as atividades que determinam um determinado objetivo.

Tabela 2.3: Atividades de Projeto

Atividade	Duração	Atividades Predecessoras
A	2	-
B	7	-
C	4	-
D	3	A
E	0	B
F	1	B
G	2	C
H	6	D,E
I	4	F,G

A terceira coluna da tabela 2.3 acima mostra as relações de precedência entre as atividades do respectivo projeto. Nota-se portanto que não existe grau de liberdade quanto a ordem de execução das atividades e o objetivo nestes problemas não é encontrar uma programa de execução que otimize o tempo total de execução das atividades, mas uma vez estabelecido o tempo em que projeto deve ficar pronto, as técnicas que são aplicadas para gerir estes tipos de problemas (PERT/CPM) procuram estabelecer uma prioridade na distribuição dos recursos entre as atividades da melhor forma, procurando satisfazer a data de término do projeto. Esta prioridade é estabelecida através do conceito de Caminho Crítico, o qual determina quais as operações que não podem sofrer atrasos durante a execução do programa, pois isto comprometeria o seu término, consequentemente elevaria os custos operacionais. A figura 2.3 mostra a construção da rede de interconectividades das atividades.

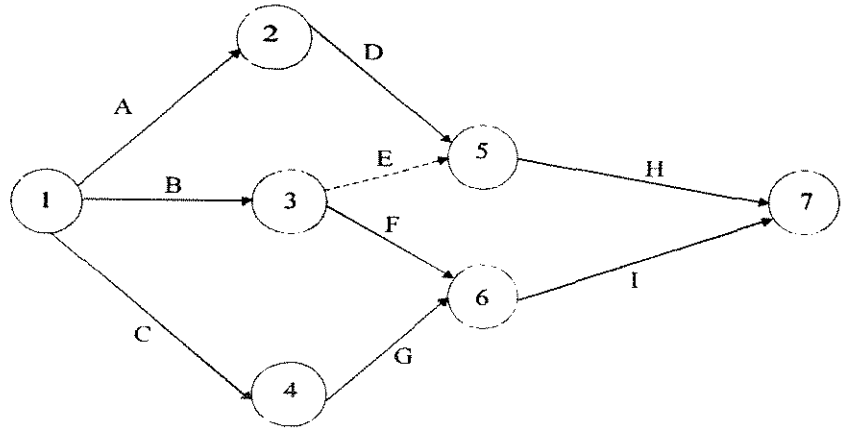


Figura 2.3: Relação entre as atividades de projeto

O problema de gerenciamento de atividades pode ser formulado como mostrado em Ming et al. (1994), onde o objetivo é encontrar o tempo de início e de término das atividades. O modelo proposto é então conveniente quando está envolvido um grande número de atividades e várias restrições de precedência, eliminando assim a elaboração da rede.

2.5. Programação de Produção em Sistemas Flexíveis

2.5.1. Introdução

Em períodos de baixos índices inflacionários, onde a estipulação de preços aos produtos para cobrir eventuais aumentos de custo de produção é bastante restrita ou porque não dizer quase impraticável, a modernização da máquina produtiva, a aplicação de otimização e controle de processos, a adoção de metodologias para aumento da produtividade, controle e planejamento de produção são essenciais para qualquer indústria que busca sua sobrevivência num mercado que em tempos de "globalização" se torna cada vez mais competitivo. Podemos dizer que o ambiente de negócios pesquisado e implantado na fase de projetos, seja em qualquer área, atualmente é caracterizado em tempos de "globalização econômica" por:

- Alta competitividade
- Preços determinados pelo mercado
- Exigência de Programas de Qualidade e Produtividade
- Necessidade constante de monitoração de atividades envolvidas na elaboração de produtos e serviços
- Necessidade de melhoria da Informação Gerencial
- Busca Contínua de Oportunidades

A exigência de programas de qualidade e produtividade, terceira característica de um ambiente competitivo apresentada acima, justifica o estudo e a pesquisa de métodos de planejamento, controle e programação de produção que venham auxiliar o desenvolvimento de atividades nos diversos setores produtivos.

Embora o modo de operação contínuo seja desejável em muitos processos químicos por causa de sua alta produtividade, o processamento não-contínuo ainda tem

seu lugar de destaque na indústria de processos químicos. Devido à necessidade de manufatura de vários produtos nas áreas da química fina e biotecnologia, os processos não-contínuos passaram a despertar maior interesse industrial e o espaço merecido na literatura especializada. Além disso, a crescente demanda de produtos de alta tecnologia e a satisfação de restrições rígidas impostas pelo mercado competitivo favorecem o processamento em modo batelada, dada a sua flexibilidade em produzir múltiplos produtos em quantidades baixas, através do compartilhamento de equipamentos. No entanto, para que essa flexibilidade possa ser bem aproveitada são necessárias ferramentas de planejamento sofisticadas que permitam maximizar a utilização dos diversos recursos compartilhados disponíveis.

A procura dos setores industriais por processos batelada deve-se em parte pelo aumento de interesse no desenvolvimento de pesquisa acadêmica na área de planejamento e programação de produção. Vários trabalhos de revisão abordando este modo de processamento são apresentados na literatura (Reklaitis,1982 e 1992; Rippin, 1983 e Ku et al, 1987). Reklaitis (1982) faz uma revisão progressista no sentido de apontar a evolução obtida no tratamento de problemas da área. Neste trabalho, também são evidenciadas as metodologias empregadas na solução dos problemas, as quais em sua maioria abrange procedimentos heurísticos, técnicas de programação matemática, técnicas de busca ("Branch and Bound").

2.5.2. Estruturas de Processamento Químico

As plantas batelada são operadas em dois modos:

- Modo contínuo
- Modo descontínuo

No que se refere à produção descontínuo, a coordenação dos trabalhos nestas plantas 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 de modo racional durante a execução das diversas tarefas.

As plantas químicas batelada são classificadas em:

- Plantas Multiproduto
- Plantas Multipropósito

Em plantas multiproduto as tarefas são executadas em uma mesma seqüência de produção. A principal característica das plantas multiproduto é a relação de precedência linear entre os estágios de produção (etapa de produção), como mostra a figura 2.4, onde vários equipamentos semicontínuos estão presentes entre os equipamentos batelada.

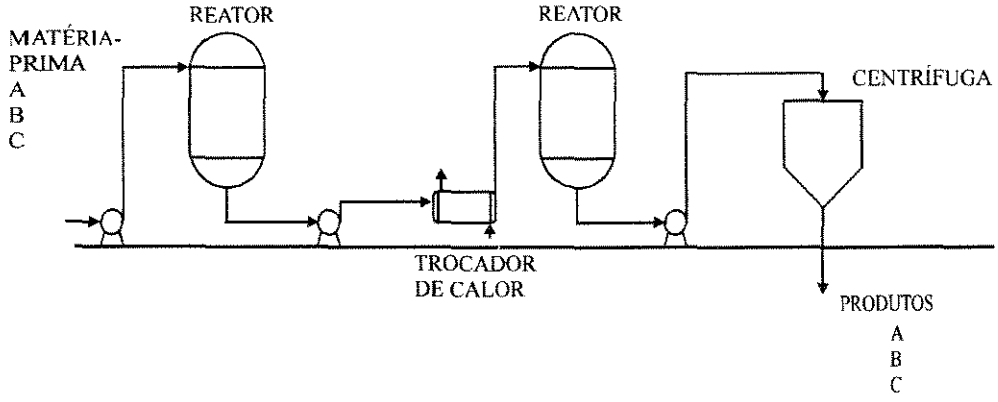


Figura 2.4: Planta Multiproduto Típica

Uma idealização feita para a solução de problemas envolvendo estas plantas é considerar que somente os equipamentos batelada constituem a estrutura de processamento (esta hipótese também é comum no problema de projeto, como visto anteriormente), mas em uma situação real equipamentos semicontínuos são muitas vezes intercalados entre os equipamentos batelada. Quando a planta batelada multiproduto é caracterizada pela ausência de semicontínuos a planta recebe o nome de "flowshop" puro, onde somente tanques batelada são considerados. Esta estrutura é esquematizada na figura 2.5, a qual contém M equipamentos onde N tarefas são desenvolvidas. O fluxo de atividades é unidirecional seguindo a ordem natural dos equipamentos. Os equipamentos são numerados de j = 1 a M e as operações da tarefa i são correspondentemente enumeradas a cada equipamento, sendo cada tarefa constituída por M operações.

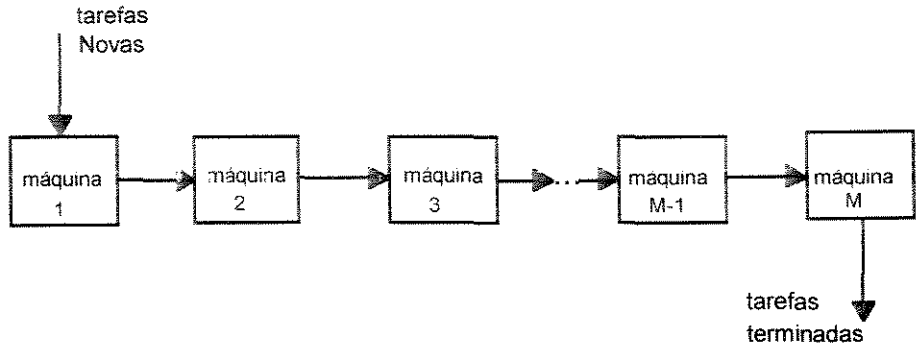


Figura 2.5: Linha "Pura" de Processamento

Em plantas multipropósito um conjunto de diferentes operações pode estar presente no mesmo tempo na planta e o mesmo produto pode ser processado por rotas diferentes.

Em plantas multiproduto, determinar um programa de produção detalhado envolve principalmente uma distribuição de tarefas no tempo, desde que não existe liberdade na escolha de equipamento. Em plantas multipropósito uma determinada tarefa pode ser executada em diferentes equipamentos, consequentemente a escolha da tarefa e do equipamento devem ser feitas previamente.

Egli e Rippin (1986) desenvolveram um programa "SRSBP" ("Short-Range Scheduling of Batch Plants") para a programação de produção em plantas multiproduto/multipropósito. O código determina um programa detalhado para a manufatura de produtos, visando minimizar o custo produtivo (custos de estocagem, custos de mudança de bateladas e custos de utilidades). Kondili et al. (1993) apresentam uma formulação baseada na representação de estado-tarefa de sistemas de manufatura, bastante conveniente para a programação de produção em plantas multipropósito e multiproduto.

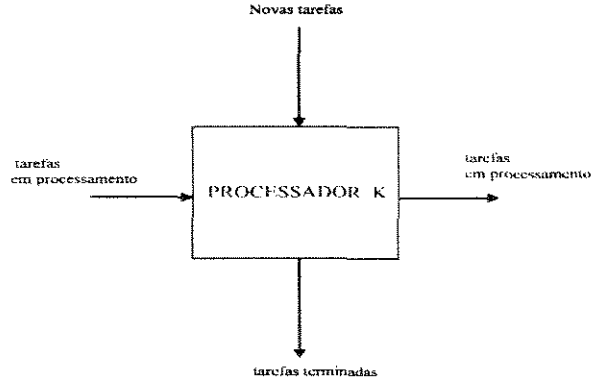


Figura 2.6: Planta Multipropósito típica

A figura 2.6 mostra a generalização de uma planta multipropósito e o desenvolvimento de tarefas neste tipo de estrutura.

2.5.3. Caracterização das Unidades Químicas

As unidades químicas podem ser definidas como aquelas em que as matérias-primas podem ser transformadas em produtos em unidades de processamento, onde são realizadas operações físico-químicas.

As principais características destas unidades são:

- Ocorrência de reações químicas
- Entradas e saídas quimicamente diferentes
- Os materiais têm natureza "contínua"
- Fluxo de material
- Ocupação simultânea das unidades de processamento
- Conectividade das unidades de processamento

Na operação contínua a produção é medida em geral pela produção volumétrica ou mássica chamada taxa de produção (por exemplo: ton/ano), havendo entrada e saída contínua de material. Na operação descontínua há entrada e saída intermitente de material, sendo a produção medida em bateladas/tempo e cada batelada possui um tamanho expresso em massa ou volume. Na tabela 2.4 são apresentados exemplos de processos contínuos e não-contínuos.

Tabela 2.4: Exemplos de Processos Contínuos e Descontínuos

Processos Contínuos (Entradas e saídas sem interrupção)	Processos Não-contínuos (Entradas e saídas intermitentes)
Petroquímica	Bioquímica
Fertilizantes,	Polímeros
Químicos (Amônia)	Cosméticos
Celulose	Alimentos

Em ambos os casos, processos contínuos ou descontínuos, as plantas podem apresentar um perfil de produção flexível no que diz respeito à capacidade de processar diferentes matérias-primas e/ou produzir diferentes produtos.

Existem ainda as plantas contínuas ou não de processamento dedicado, isto é, processam uma só categoria de matérias-primas produzindo uma só categoria de produtos. Estas plantas são chamadas de "dedicadas".

As plantas contínuas flexíveis são aquelas que periodicamente sofrem mudanças nos pontos de operação e mesmo no fluxo mássico, sem que haja interrupção na entrada e saída de material. Há então, neste caso, um período de transição em que se tem produtos ou intermediários fora de especificação de produção (refugo ou material de baixo valor agregado). Um dos objetivos do planejamento e programação de produção, neste caso específico, é a minimização deste período de transição (por exemplo: indústria petroquímica processando óleos crus com especificações diferentes, a quantidade de refugo gerado é função da qualidade dos óleos).

As principais diferenças entre os processos contínuos e não-contínuos são apresentadas na tabela 2.5.

Tabela 2.5: Diferenças entre os processos contínuos e não contínuos

Unidades Contínuas	Unidades batelada ou semi-contínuas
Baixa armazenagem intermediária	Alta armazenagem intermediária
Alta estocagem	Baixa estocagem
Regime estacionário	Regime não-estacionário
Unidades dedicadas (não há alocação de tempo)	Unidades flexíveis (alocação no tempo)
Larga escala de produção	Pequena escala de produção
Produtos de baixo valor agregado	Produtos com alto valor agregado

A tabela 2.6 mostra as diferenças entre as unidades descontínuas e semicontínuas.

Tabela 2.6: Diferenças entre as unidades descontínuas e Operações semi-contínuas

Unidades Descontínuas	Operações Semi-contínuas
• Entradas e saídas intermitentes	• Operação intermitente (partidas e paradas)
• Especificadas por volume ou dimensão dos equipamentos	• Especificadas por taxas
	• Taxas de operação variáveis
• Opera em ciclos	• Etapas de operação
• Etapas de Operação	- Processamento.
-Transferência de matéria-prima para o equipamento	- Limpeza/preparação
	-Período ocioso
- Processamento de material	
-Transferência de produto	
-Limpeza e preparação ("setup")	• Exemplo de Equipamentos:
• Exemplo de Equipamentos	(Bombas, Filtros, trocadores de calor, centrífuga)
(Reatores, Secadores, Cristalizadores, Colunas de destilação)	

Nas últimas duas décadas tem-se observado uma demanda crescente por produtos de alta tecnologia ou tecnologia de ponta que são normalmente fabricados em unidades químicas não-contínuas e com isso engenheiros químicos começaram a se preocupar com aspectos novos deste tipo de unidade. Dentre estes pode-se distinguir:

- Dinamismo de alocação das operações;
- As descontinuidades devidas ao início e término das etapas individuais de processamento.

Os processos descontínuos (batelada e semicontínuos) são mais complexos que os contínuos, no entanto a busca de processos descontínuos vem crescendo, este fato é justificado basicamente por diversas razões:

- Viabilizam processos com tempos de residência longa;
- Viabilizam os processos de síntese complexos;
- Exigem conhecimento mais limitado dos processos;
- Quando as informações sobre o "scale-up" são inadequadas.
- Flexibilidade de Produção

- Natureza multipropósito dos equipamentos permitem a execução de múltiplas tarefas e múltiplos produtos;
- São adequados para a produção de baixos volumes;
- São adequados quando a demanda de produtos é sazonal ou incerta;
- Permitem absorção de incertezas do processo.

Por exemplo, em função das altas temperaturas empregadas na indústria siderúrgica e da indisponibilidade de integrar seus equipamentos através de fluxo contínuo, a produção de aço é feita em unidades descontínuas (restrição tecnológica). Por outro lado, em função da sazonalidade da produção de leite e demanda de seus derivados, a indústria de leite opera, em geral, em regime semicontínuo, garantindo assim uma menor ociosidade de seus equipamentos.

2.5.4. Caracterização das Operações Químicas

As operações descontínuas são caracterizadas por (Rippin, 1983)

- Especificação da operação que deve ser executada
- Tempo necessário para executar a tarefa
- Capacidade necessária

Uma batelada de produto é produzida através da execução de uma seqüência de operações, sendo que o rendimento e a quantidade de produto são determinados pelo grau de conversão das operações envolvidas. A freqüência de execução das diferentes bateladas é determinada pelos tempos necessários para a execução das operações e as quantidades que podem ser produzidas são determinadas pelas dimensões dos equipamentos disponíveis.

A extensão em que uma operação é executada é definida pelo estado das correntes que deixam o equipamento em relação às correntes de entrada ou alguma especificação externa. O desempenho é definido em termos de conversão de uma reação química, grau de separação, ou alcance de uma temperatura, dentre outros. Se existem meios de especificar "a priori" estas medidas de desempenho, então o desempenho global da seqüência de operações pode ser calculado através de balanços de massa e energia, da mesma maneira que realizados para processos contínuos.

O tempo necessário para a execução das operações é normalmente especificado antecipadamente. Se a integridade da batelada é preservada através de todos os equipamentos, então o intervalo de tempo no qual as bateladas sucessivas podem ser produzidas será controlado pelo equipamento que demanda o maior tempo de execução de uma operação.

A quantidade a ser produzida determinará a capacidade necessária para operação. A capacidade necessária por unidade de massa é chamada de "size factor", e é normalmente expressa em termos de volume por unidade massa.

Uma tarefa de processo é um conjunto de operações, cada uma delas realizada em um tipo de equipamento. As tarefas de processo são caracterizadas por:

- Seqüência de operações: nos problemas de projeto o equipamento onde a tarefa deverá ser realizada pode ainda não estar especificado, mas tanto para a atividade de projeto quanto para a de planejamento, a seqüência de operações é fixada "a priori". Uma seqüência de operações define um processo completo de produção.
- Modo de execução das operações de uma tarefa (contínuo, semicontínuo ou batelada)
- Desempenho da operação: em geral é fixado "a priori", mas em função de modificações que podem ocorrer no processo, tais como decréscimo da atividade catalítica, incrustações em trocadores de calor e qualidade das matérias-primas podem necessitar revisões periódicas através do balanço de massa e energia com as devidas às alterações de parâmetros de sistemas.
- Tempo de operação das atividades: depende da forma de execução da operação, e também está sujeito as alterações em função de alterações do nível de desempenho da tarefa.
- Capacidade necessária: a relevância das mudanças na capacidade para o desempenho de uma operação está relacionada com a dimensão do equipamento disponível para executá-la. Esta característica é especialmente crítica se o equipamento é gargalo do processo.
- Incerteza na especificação das operações: são devidas especialmente às variações na qualidade de matéria-prima, degradação antecipada do desempenho dos equipamentos ou efeito "scale-up".

2.5.5. Caracterização de produto

O produto pode ser definido em quantidade ou tipo, além disso devem também ser especificadas as seguintes características:

- Estrutura do produto: determinada pela receita de produção, alguns produtos, por exemplo, dependem não somente de matérias-primas mas também de materiais intermediários produzidos durante o processamento.
- Flexibilidade de demanda: em alguns casos os níveis de produção podem ser especificados em um intervalo, viabilizando a oportunidade de manufaturar produtos mais lucrativos.
- Prazo de entrega: determina datas em que um determinado produto devem estar pronto para ser entregue ao cliente. Perfis de demanda, construídos a partir da carteira de pedidos da empresa, assim como um perfil de demanda contínua ou discreta no tempo.

2.5.6. Especificação dos Equipamentos

As especificações dos produtos da seqüência de operações, são normalmente rígidas. A complexidade dos processos descontínuos reside na variedade de configurações possíveis de equipamentos e a maneira como as tarefas podem ser alocadas, originando um problema de natureza combinatória. Em geral as configurações possíveis dependem da seqüência de operações de tarefa e do tipo de operações a serem realizadas, enquanto que possibilidades de alocação refletem restrições globais do sistema. Os equipamentos são especificados através de:

- Tipo: a especificação mais simples de tipo de equipamento é restringir cada tipo de operação a um único equipamento. No entanto em certas estruturas, principalmente nas plantas descontínuas, existem mais de um equipamento habilitado a desenvolver uma determinada operação. Neste caso pode ser necessário especificar se há alguma preferência no uso equipamentos que tem a mesma finalidade (por exemplo vários reatores).
- Número de equipamentos: No caso de Planejamento de Produção este parâmetro é fixo enquanto que na atividade de projeto o que se pretende muitas vezes é determinar o número de equipamento a instalar.

- Dimensões: No caso de planejamento e programação de produção é um parâmetro de entrada do problema, na fase de projeto as dimensões são variáveis.
- Possibilidade de interconexões: No caso geral todas as interconexões possíveis entre equipamentos poderiam ser estabelecidas, de forma que não há qualquer impedimento das operações aos equipamentos. No caso em que as interconexões são limitadas, são criadas restrições de alocação que restringem o espaço de soluções de problemas de projetos e de planejamento da produção.
- Grupos de equipamentos em paralelo operando em fase ou fora de fase.

2.6. Estabelecimento de um Problema de Programação de Produção

2.6.1. Introdução

A Programação de produção pode ser definida como uma técnica que possibilita o desenvolvimento de um programa de produção que mostre como executar as diversas atividades industriais com uso racional de recursos compartilhados (energia, vapor, mão-de-obra) com diminuição de estoque de materiais intermediários.

Tomando como exemplo unidades químicas que operam em modo batelada (onde planejamento e controle de produção tornam-se ferramentas imprescindíveis), nas quais são manufaturados diversos produtos através do compartilhamento de diversos recursos existentes na estrutura de processamento, torna-se imperativo a adoção de técnicas de programação de produção para se obter um plano de produção que otimize um determinado critério de desempenho, tais como o tempo total de produção ("makespan"), maximização do lucro, minimização dos atrasos. A crucialidade do planejamento de produção muitas vezes é justificada pelas interações existentes entre indústria e mercado, assim como flutuações na demanda, além disso, há limitação na oferta de recursos, o que muitas vezes pode evitar o cumprimento de obrigações contratuais.

O problema de Programação de operações está mais associado às plantas batelada, já que esta atividade praticamente inexiste em plantas contínuas, visto que a grande parte das indústrias de processamento contínuo é dedicada à produção de poucos produtos e não se necessita de alocação de material durante o horizonte de produção. Já em plantas batelada ou semicontínuas, por condicionantes tais como sazonalidade de produtos, restrições tecnológicas, flutuações na demanda

impossibilitam a operação em modo contínuo. Com isso, a quantidade exigida por forças de mercado é obtida geralmente pela combinação de diversas bateladas de produto. A decisão de começar e em que período de tempo a desenvolver uma determinada batelada de produto para atender a uma demanda futura com um custo de produção mínimo é uma tarefa bastante difícil, ou porque não dizer impossível se alguma ferramenta para programar a produção não for utilizada. Em plantas que operam em modo não-contínuo existem diversas possibilidades para se alcançar uma determinada demanda, mas para se obter um programa de produção ótimo de manufatura de produtos, de modo que se otimize uma determinada função objetivo (como por exemplo, "makespan") e racionalizando os recursos existentes é necessário a utilização de simulação.

O problema de programação de produção em uma única máquina é tido como o caso mais simples de programação de produção. Graves (1981) cita vários trabalhos envolvendo esta estrutura de processamento. Baker (1974) também mostra alguns resultados relacionados a esta estrutura.

No que se relaciona ao processamento em um único estágio com equipamentos em paralelo, a literatura apresenta o procedimento de McNaughton (Baker,1974) visando à minimização do makespan em problemas de seqüenciamento. Além desses procedimentos específicos, a literatura fornece algumas técnicas heurísticas para o seqüenciamento de tarefas em plantas multiproduto. Algumas destas são mostradas na no capítulo 3.

Segundo Graves (1981) três informações básicas devem ser consideradas na classificação de problemas de programação de produção.

- Necessidade de produção
- Complexidade de processamento
- Critério de desempenho ou função objetivo

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

- Produção por encomenda ("Open shop"): onde todas as ordens de produção são diretas do mercado.

- Produção para estoque ("Closed shop"): as ordens são indiretas, frutos da necessidade de manutenção de estoques.

A complexidade de solução de problemas de programação de produção está associada ao número de equipamentos envolvidos no processo. Os elementos básicos presentes nestes problemas são:

- Um conjunto de N tarefas a serem desenvolvidas;
- Um conjunto de M equipamentos disponíveis para processar as tarefas;
- Tempos de processamento das tarefas em cada processador;
- Política de armazenagem da planta;
- 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.

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, ou pela adoção de hipóteses simplificadoras ou pela utilização de metodologias não exatas (métodos heurísticos).

A classificação de problemas de programação encontra-se ligada ao critério utilizado para avaliar o desempenho da planta, pela estrutura de processamento e pelas regras de produção que governam o processo (Reklaitis, 1982). As funções de custo são geralmente de dois tipos:

- Critério baseado em custo real;
- 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 com facilidade. Já em plantas com grau de

processamento mais complexo é comum se utilizar critérios de desempenho de sistema, dentre os quais estão:

- Minimização do tempo total de execução de todas as tarefas ou "makespan";
- Minimização do máximo tempo de residência ou "flowtime";
- Minimização do tempo de residência médio;
- Minimização dos atraso das tarefas (tardiness");
- Minimização do tempo de atraso ou "mean tardiness";
- Minimização de custo sobre-mudança de produtos;

2.6.2. Definições das Funções de Desempenho mais utilizadas

A análise das principais variáveis e funções de desempenho básicas no tratamento de problemas de programação /seqüenciamento é vista em Baker (1974). Um problema de programação de produção tem com dados principais:

- Tempo de processamento da tarefa i no processador j
- Instante em que a tarefa i está pronta para ser processada ("ready time")
- Instante estabelecido para a entrega da tarefa i ("due date")

Os critérios de desempenho definidos a seguir são funções das variáveis: tempo de processamento da tarefa i no processador j, instante em que a tarefa i está pronta para ser processada ("ready time") e instante estabelecido para a entrega da tarefa i ("due date").

a) Tempo de Residência ($F_i=C_{iM}-r_i$): avalia o tempo em uma tarefa i permanece no processo. Onde C_{iM} equivale ao tempo de término da tarefa i no processador M (último processador habilitado para desenvolver a tarefa i) e r_i (data em que a tarefa está pronta para ser desenvolvida).

b) "Lateness" ($L_i = C_{iM}-d_i$): avalia o tempo que excede a data de entrega estabelecida para a tarefa i. Onde d_i equivale a data de entrega estipulada da tarefa i.

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

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.

CAPÍTULO 3

METODOLOGIAS DE SOLUÇÃO DE PROBLEMAS DE PROGRAMAÇÃO DE PRODUÇÃO

3.1. Introdução

Este capítulo estabelece algumas metodologias para o tratamento de problemas de programação de produção sem o intuito de explorar detalhadamente cada uma delas.

Dentre as metodologias para o tratamento de problemas de programação de produção destacam-se a programação matemática, busca em árvore e o métodos heurísticos. Nos últimos anos técnicas heurísticas como algoritmos genéticos e lógica "fuzzy" vêm ganhando espaço de utilização nesta área.

As atividades de planejamento e controle de produção ganham cada vez mais espaço entre as atividades do processamento industrial. O panorama econômico mundial concatenado às perspectivas de um mercado cada vez mais exigente justificam a elaboração e implementação de sistemas que executem estratégias eficientes de controle e programação de produção de sistemas de manufatura, visando com isso alcançar as metas estipuladas pela organização de modo menos oneroso.

O rápido desenvolvimento da tecnologia computacional e o surgimento de pacotes otimizadores cada vez mais próximos às necessidades do usuário, tal como a interface GAMS ("General Algebraic Modeling System"), têm permitido a solução dos problemas de programação com maior nível de detalhamento e tornando exoráveis novas estratégias de solução.

3.2. Metodologia "Branch and Bound" aplicada a problemas de Programação de Produção

A enumeração implícita compreende uma busca controlada por uma função objetivo estabelecida adequadamente para cada problema. A natureza combinatorial do espaço de solução dos problemas de programação de produção inviabiliza a análise do conjunto global de todas as soluções possíveis. Nilson (1971) aborda alguns métodos de redução de espaço, dentre eles a pesquisa "branch and bound" em profundidade ("Depth first"), onde os Nós com maior nível de profundidade na árvore são preferidos, e a pesquisa em largura ("Breadth first"), onde os Nós mais promissores para otimizar a função de custo são escolhidos para a ramificação. A Metodologia "branch and bound" consta basicamente das seguintes etapas:

- Etapa ramificação: toma o problema original e então gera um conjunto de subproblemas;
- Etapa de cálculo do limitante: calcula o menor limitante da função objetivo de cada subproblema.

Na apresentação de busca em árvore, considere um problema original representado por V^0 e seus subproblemas como mostrados na figura 3.1. 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 os Nós V_1^1, V_2^1 até o Nó 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 escolhidas. Cada subproblema gerado é novamente decomposto, gerando o segundo nível. O subproblema V_2^1, por exemplo, é tomado como o próximo Nó a ser ramificado, originando os Nós $V_{21}^2, V_{22}^2, \ldots, V_{2N}^2$. Quando se adota a estratégia de busca em profundidade esta ramificação prossegue até a base da árvore (onde não existe mais possibilidade de ramificação), gerando uma solução completa ou limitante superior. Depois de alcançada a base a etapa de retrocesso ("backtraking") é colocado em prática a busca de subproblemas não solucionados e que são promissores à otimização da função objetivo.

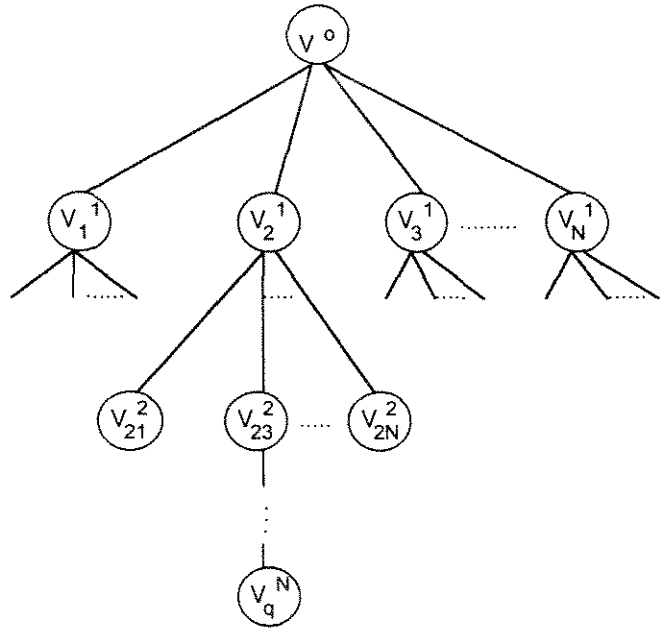


Figura 3.1: Árvore "Branch and Bound"

O procedimento de cálculo de limitante é crucial para que a estratégia de enumeração implícita forneça uma solução ótima. Para a redução do espaço de busca é necessário que esse limitante da função objetivo seja estimado com garantia, evitando assim que regiões factíveis não sejam eliminadas durante a busca.

Uskup e Smith (1975) propõem um algoritmo que se utiliza da busca "branch and bound" para o sequenciamento de tarefas onde tempos de preparação dos equipamentos são considerados. O custo de estabelecimento envolvido na mudança de tarefas é considerado diretamente proporcional aos tempos de preparação, os quais são necessários para a minimização do tempo total de produção das tarefas ("makespan"). Lomnicki (1976) propõe um algoritmo exato para o programação de tarefas em problemas envolvendo três máquinas. Seguindo esta diretriz, Santos (1994) também propõe uma função de cálculo de limitante em tempo total de produção ("makespan") para o sequenciamento de tarefas em plantas multiproduto, envolvendo tempos de preparação de equipamentos.

Utilizando esta técnica Rodrigues (1992) desenvolve um algoritmo de sequenciamento e alocação de operações em sistemas flexíveis (estrutura "flowshop") com restrições sobre recursos compartilhados.

3.3. Programação Matemática aplicada a problemas de Programação de Produção

A modelagem matemática exige o devido estabelecimento do problema e da manipulação eficiente das restrições e variáveis que determinam as características do problema. Com o surgimento de situações modernas envolvendo otimização (em setores como: economia, gerenciamento industrial, processamento industrial), muitas vezes a adoção matemática tradicional não fornece modelos condizentes às situações práticas.

Quando utilizada para o tratamento de problemas, a programação matemática é dividida em: Programação Linear, Programação Não-Linear, Programação Inteira, Programação Linear Inteira Mista e Programação Não-Linear Inteira-Mista. Esta classificação é feita basicamente de acordo com a natureza linear ou não linear das restrições envolvidas e da presença ou não de variáveis inteiras.

A programação linear foi uma das primeiras técnicas de otimização a transpor os conceitos teórico- matemáticos e adquirir um caracter prático, principalmente em problemas na área petroquímica (Dantzig, 1963).

A programação linear inteira mista constitui segundo a divisão anteriormente feita, uma técnica que propicia o estabelecimento de modelos de problemas de programação de produção em sistemas flexíveis.

A utilização de uma abordagem de programação matemática para o tratamento de problemas de programação de produção em plantas polivalentes exige a representação matemática adequada da estrutura de processamento, como será visto no capítulo 4. Uma vez que os problemas sejam devidamente modelados, os pacotes otimizadores destinados à solução dos respectivos modelos utilizam a metodologia "branch and bound" para a satisfação das variáveis definidas como inteiras. Quando somente variáveis binárias (0 e 1) estão presentes, tem uma árvore binária, onde cada Nó escolhido para a ramificação pode gerar no máximo até dois novos Nós. A maioria dos pacotes comerciais destinados à solução de problemas via programação matemática incorporam esta técnica, como é caso do OSL.

3.4. Programação Inteira-Mista

3.4.1. Estabelecimento da formulação

A formulação geral da programação inteira-mista de um determinado:

$$\text{min ou max} \quad Z = f(x,y)$$

$$\text{sujeito a} \quad \begin{cases} h(x,y) = 0 \\ g(x,y) \leq 0 \\ x \in R^n, y \in N_+^m \end{cases}$$

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

No caso da programação linear inteira mista ("MILP"), a função objetivo Z e as restrições h e g são lineares em x e y. Em muitos problemas práticos as variáveis inteiras são pertencentes ao conjunto de valores inteiros e binários, isto é, $y \in (0,1)^m$.

No caso da programação não linear inteira-mista ("MINLP"), a função objetivo e/ou as restrições são não-lineares. Na prática assume-se comumente a forma linear para as variáveis binárias e não-lineares para as variáveis contínuas.

Quando se utiliza a programação matemática, uma das dificuldades que surge na solução dos problemas é a natureza combinatorial do espaço de solução desses problemas (Nemhauser e Wolsey,1988).

3.4.2. Metodologia "Branch and Bound" aplicada à Programação Linear Inteira-Mista

A técnica de programação linear inteira-mista consiste basicamente de duas fases:

- Resolver o problema relaxando as restrições de integralidade (não satisfação das variáveis binárias);
- Proceder à busca da solução ótima procurando satisfazer as restrições de integralidade.

A metodologia "branch and bound" aplicada à programação inteira-mista usa o processo de geração e análise de subproblemas através da fixação de variáveis inteiras. Cada Nó da árvore pode ser dividido em dois novos Nós (um representando a ramificação "up", onde a variável escolhida para a ramificação é fixada no número inteiro mais próximo acima do valor corrente da variável, e o outro Nó a variável é fixada no número inteiro logo abaixo do valor corrente, tem-se então a ramificação "down"). O limitante da função objetivo de cada Nó é calculado resolvendo um Problema Linear (PL), isto para qualquer que seja a função de custo do problema.

O número de Nós gerados na busca "branch and bound" depende do número de variáveis binárias envolvidas. Esta relação é dada por 2^N, onde N é número de variáveis inteiras.

O procedimento "branch and bound" é incorporado por diversos pacotes comerciais destinados à otimização de problemas envolvendo variáveis inteiras. O OSL (Otimization Subroutine Library) é um dos que utilizam a metodologia acima descrita, assim como o ZOOM, LINDO, SCICONIC, OSL.

Uma alternativa para diminuir a complexidade de desenvolvimento da metodologia "branch and bound" é reduzir a diferença entre a solução relaxada (equivalente ao limitante da função objetivo no Nó zero) e a solução ótima. Esta ação interfere na profundidade que a árvore pode atingir, consequentemente diminuindo a enumeração de Nós. Para tanto são empregados métodos complementares que

melhoram a solução relaxada, o que reduz o tempo computacional. Os principais métodos que seguem esta abordagens são classificadas em:

- Métodos de Decomposição (Benders, 1962);
- Métodos de Planos de Corte (Applegate e Cook, 1991);
- Métodos baseados em Lógica (Hooker, 1994).

Alguns pacotes otimizadores, como por exemplo OSL (IBM, 1992), possuem estratégias que interferem no processo de enumeração da árvore de busca, como é caso de uso de conjuntos ordenados de variáveis inteiras ("Special Ordered Sets") e técnicas de preprocessamento de Nós, as quais se utilizam de "heurísticas" na busca de fixar variáveis em um determinado Nó. O uso de técnicas de preprocessamento não garante a diminuição de complexidade de solução de problemas e o tempo computacional de solução dos mesmos pode aumentar.

3.4.3. Análise da Programação Linear Inteira-Mista no cenário da Programação de Produção de Sistemas Flexíveis

O planejamento e programação de produção em sistemas flexíveis de produção são considerados uma área em que as técnicas de programação inteira-mista ganham um caráter de aplicabilidade, em especial a programação linear inteira-mista. Isto porque a representação matemática adequada das estruturas flexíveis exige a definição tanto de variáveis discretas (variáveis inteiras, em especial binárias) quanto de variáveis contínuas.

Uma revisão mostrando a aplicação das técnicas de otimização via programação linear inteira-mista e programação não linear inteira mista em sistemas de produção não-contínuos é apresentada por Grossmann et al. (1992). Os autores procuram mostrar a crescente utilização destas técnicas e como a programação matemática vêm desempenhando um papel relevante na solução de problemas ligados ao processamento em sistemas que operam em modo batelada e síntese de processos.

A programação de produção via formulação matemática exige duas restrições que são básicas para a modelagem:

- A restrição de ordenamento das operações de diferentes tarefas em cada equipamento;

- A restrição de precedência tecnológica, determinando assim que uma operação de uma determinada tarefa só pode ser desenvolvida depois do término da operação anterior.

Na literatura um grande número de formulações para o tratamento de problemas de planejamento e programação são propostas. No cenário de Engenharia Química um dos principais modelos direcionados à programação de produção em planta multipropósito foi proposto por Rich e Prokopakis (1986). Entre outras considerações, a formulação resolve conflitos potenciais quando dois ou mais produtos necessitam do mesmo equipamento, através da adição de restrições representadas por disjunções, as quais permitem o ordenamento das tarefas através de relações de precedência. O modelo é resolvido com o pacote LINDO (Linear Interactive Discrete Optimizer).

Uma outra formulação que representa de modo mais adequado os sistemas produtivos em plantas químicas é apresentada por Kondili et al. (1993). Os autores desenvolvem uma representação do processo através da definição da rede estado-tarefa ("State Task Network" ou "STN"). A principal característica dessa modelagem é a aplicação de balanço de material nos diversos estados (matérias-primas e materiais intermediários) para o ordenamento tecnológico das operações. Esta modelagem é utilizada neste trabalho e melhor detalhada no capítulo 4.

A formulação rede-estado-tarefa representa de forma relativamente simples várias características de processamento industrial, assim como:

- Divisão de bateladas de materiais durante o processo
- Mistura de materiais de acordo com a receita de produção do produtos
- Armazenagem intermediária
- Produção simultânea de vários produtos finais e/ou intermediários
- Compartilhamento de recursos
- Tempos de preparação de equipamentos

A formulação matemática do problema baseada na representação "STN", com características "MILP" ("Mixed Integer Linear Programming"), é desenvolvida adotando-se a discretização uniforme do tempo, em que o horizonte de planejamento é dividido em um número de intervalos (períodos de tempo) de igual duração. Shah et al. (1993) apresentam várias técnicas matemáticas para melhorar a eficiência computacional da formulação rede-estado-tarefa. Uma reformulação das restrições de

alocação é proposta para reduzir a diferença entre a solução ótima e a solução relaxada ("gap").

Pinto e Grossmann (1994) apresentam uma formulação direcionada ao programação de plantas batelada com múltiplos estágios, os quais podem conter equipamentos operando em paralelo. Baseada em uma representação contínua do tempo, contrapondo-se à formulação rede-estado-tarefa. O horizonte de tempo de produção é também composto por períodos de tempos, os quais não possuem duração previamente definida, como estabelecido na modelagem de Kondili et al. (1993).

3.4.4. Elementos Básicos de um Modelo direcionado à Programação de Produção

O objetivo da formulação matemática é escrever o conjunto de equações e/ou inequações (denominadas de hiperplanos de restrições) que descrevam adequadamente o processo produtivo. As variáveis envolvidas no problema, as quais determinam as restrições do modelo, podem ser definidas como variáveis de alocação ou como variáveis de precedência. Sobre o problema que está sendo resolvido, uma variável de alocação oferece mais informações quando comparada a uma variável de precedência. Para verificação da potencialidade de uma variável de alocação considere primeiramente a variável de precedência utilizada na formulação proposta por Rich e Prokopakis (1986).

$$Z_{ikj} = \begin{cases} 1 \; sea \; tarefa \; i \; precede \; a \; tarefa \; \mathrm{k} \;\; no \; equipamento \; j \\ \\ 0 \; caso contrario \end{cases}$$

Considerando o desenvolvimento de uma árvore de busca "branch and bound", quando a alguma variável Z_{ikj} assume valor 1, não se pode utilizando esta informação, por exemplo, avaliar o consumo de recurso em um determinado intervalo de tempo sem saber os tempos de início e de fim desta alocação. Pois a única informação dada por uma de precedência é se uma determinada operação antecede uma outra em seu processamento.

$$W_{ijk} = \begin{cases} 1 & \textit{se a tarefa i inicia o processamento no equipamento j no periodo de tempo k} \\ 0 & \textit{caso contrario} \end{cases}$$

Contrapondo-se à variável de precedência Z_{ikj}, a variável de alocação W_{ijk}, como utilizada na formulação Kondili et al. (1993), fornece mais informações mais relevantes quando a mesma por exemplo assume valor 1 (W_{ijk}=1). Nesta situação, pode-se por exemplo avaliar o consumo de recurso a partir do período de alocação k da tarefa i no equipamento j, ou seja, conclui-se então que uma variável de alocação é forte para a construção parcial do programa de produção, pois se sabe exatamente onde a operação está alocada, o que não acontece com uma variável de precedência. Esta característica é explorada no desenvolvimento da abordagem de tese.

O custo associado ao se utilizar variável de alocação W_{ijk}, como é o caso da modelagem proposta por Kondili et al. (1993), é a necessidade de definição de um número de períodos de tempo, onde as operações são alocadas, tornando o problema dependente desta discretização.

Quando se parte para o solucionamento de problemas de programação de produção, seja qual for a metodologia a ser adotada para resolvê-los, algumas simplificações são geralmente feitas, o que leva, no caso de se adotar a programação matemática, às formulações mais próximas ou menos próximas dos problemas reais. Analisando algumas formulações que aparecem na literatura, as simplificações que são geralmente adotadas são:

- Todos os dados são determinísticos sobre o horizonte de tempo de produção;
- Uma vez iniciada uma determinada operação, esta não pode ser interrompida até o seu término;
- Não existência de recursos compartilhados;
- Os tempos de preparação dos equipamentos ("setup times") são independentes da seqüência e podem ser incluídos nos tempos de processamento.

Assim como todo o problema de otimização com satisfação de restrições possui uma função objetivo, no problema de programação de produção não é diferente, a função de desempenho econômico da planta também deve ser estabelecida. Esta

função deve ser composta, em princípio, por todos os custos relevantes do sistema. No entanto, alguns destes custos são difíceis de serem avaliados na prática, como por exemplo o custo de estoque de intermediários. Desta forma, medidas de desempenho são bastante utilizadas na literatura ("Mean Flow Time", "Tardiness", "Earliness ", "Makespan").

3.5. Heurísticas de Seqüenciamento de tarefas

Na literatura são citados alguns procedimentos heurísticos para o tratamento de problemas de sequenciamento, ou seja, problemas que não levam em consideração o compartilhamento de recursos. Baker (1973, 1974) e Mah (1990) apresentam heurísticas clássicas, sendo algumas apresentadas abaixo:

a) Alocar Operações com Tempo de Processamento mais Longo ("Longest Processing Time"): No caso de operações sem pré-ocupação, os problemas com equipamentos idênticos podem ser tratados pela heurística "LPT", onde as operações com maior tempo de processamento são designadas para o equipamento que apresentar menor tempo de processamento.

b) Alocar as operações com Tempo de Processamento mais Curto ("Shortest Processing Time"): Esta heurística apresenta um raciocínio contrário ao da heurística "LPT" e também pode ser utilizada visando a minimização do tempo de execução total. Neste procedimento heurístico, o equipamento com menor tempo de processamento é escolhido para receber a tarefa com menor tempo de manufatura.

Além desses procedimentos citados acima, nos últimos anos um número de estratégias heurísticas para o solucionamento de problemas de programação de produção vem ganhando importância na literatura, sendo justificado este estudo pela extrema complexidade de solução que permeiam esses problemas.

Morton e Pentico (1993) apresentam novas direções para o tratamento de problemas de programação. Dentre os métodos citados pelos autores que surgem para se incorporarem aos métodos já existentes estão: Busca Tabu, "Simulated Annealing" e Algoritmos Genéticos. Os métodos clássicos como: "Branch and Bound", Programação Dinâmica, Pesquisa de vizinhança também são desenvolvidos pelos autores.

3.6. Metodologia "Branch and Cut"

A metodologia "Branch and Cut" não difere muito da metodologia "branch and bound". A principal diferença é que análise de proposições lógicas é realizada durante a pesquisa.

Segundo Raman e Grossmann (1992), dois problemas surgem quando se propõe o uso de análise lógica em problemas de programação matemática, que são a representação das proposições lógicas (forma de disjunção ou na forma de conjunção) de uma expressão lógica e a incorporação destas proposições durante o processo de solução.

Hooker (1994) mostra como uma expressão lógica pode ser colocada na forma matemática e posteriormente adicionada ao modelo. O autor também relata que uma inequação pode gerar várias proposições lógicas, no entanto, o autor afirma que não é necessário obter todas, pois isto é semelhante a gerar todos os planos de corte para um problema de programação inteira.

A metodologia "Branch and Cut" combina três ferramentas: geração de expressões lógicas, análise de proposições lógicas e cálculo de limitante da função objetivo. As etapas deste procedimento são apresentadas abaixo:

a) Relaxação do problema: Obtido pelo relaxamento das variáveis inteiras do problema.

b) Regra de Ramificação: Entre as regras de ramificação existe aquela que escolhe a variável binária mais próxima de 0.5 (Ming and Sahindis, 1995) e portanto mais distante de ser satisfeita em 0 ou 1. Outras regras incorporam penalidades ao fixar uma determinada variável inteira em seu limitante superior ou inferior. Esta última é mais utilizada no pacotes otimizadores, como é o caso do OSL.

c) Regra de seleção do subproblema: Dois procedimentos são bastante citados na literatura. A pesquisa em profundidade ("Depth first") e a pesquisa em largura (Breadth first). Vantagens e desvantagens destes dois procedimentos são apresentadas por Nilson (1971). Uma alternativa de busca é utilizar os dois procedimentos, ou seja, aplicar a pesquisa em profundidade para obter de imediato uma solução do problema, e em seguida partir para a busca em largura. Tal estratégia é utilizada pelo solver ZOOM.

d) Geração de proposições lógicas: Nesta fase proposições lógicas que são violadas pelo problema relaxado são transpostas para a forma de inequações(como mostrado no capítulo 5) e adicionadas ao problemas.

e) Análise Lógica: Proposições na forma lógica auxiliam na fixação de variáveis e/ou na eliminação de variáveis candidatas à ramificação.

A eficiência da "branch and bound" é determinada pelo número de Nós gerados durante a busca implícita. Este número depende do "gap" (diferença entre a solução inteira e o limitante inferior no Nó raiz), do cálculo do limitante da função objetivo e do processo de seleção do subproblema. Na fase de análise lógica é necessária a avaliação de expressões (proposições) lógicas provenientes do conhecimento da estrutura lógica do problema.

No capítulo 5 é apresentado um detalhamento de uso de proposições lógicas em problemas que envolvem variáveis inteiras, exemplificando o uso da metodologia "branch and cut". Estas exploram a estrutura lógica presente em problemas diminuindo com isso a complexidade de solução.

CAPÍTULO 4

MODELAGEM DE SISTEMAS FLEXÍVEIS VIA REPRESENTAÇÃO REDE-ESTADO-TAREFA

4.1. Introdução

Neste capítulo, é apresentada a modelagem de sistemas flexíveis de manufatura através da representação Rede-Estado-Tarefa ("STN"-State Task Network) proposta por Kondili et al. (1993).

A formulação via Rede-Estado-Tarefa é aplicada na programação de produção de curto prazo em plantas flexíveis. Uma das características dessa formulação é a utilização da representação da estrutura de processamento da planta através do detalhamento das receitas de manufatura dos produtos. Os elementos utilizados nesta representação são mostrados na figura 4.1, complementados pelas descrições que seguem.

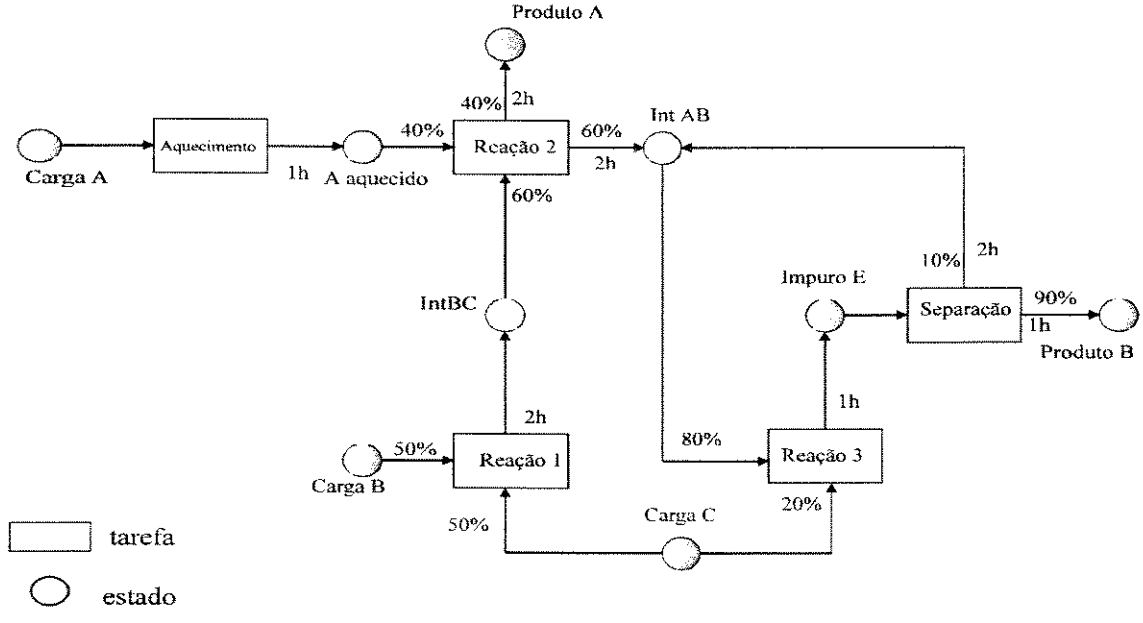


Figura 4.1: Representação da Estrutura de Produção via Rede-Estado-Tarefa (Kondili et al., 1993)

4.2. Descrição dos Elementos Simbólicos utilizados na Representação Rede-Estado-Tarefa

Na planta flexível mostrada na figura 4.1, dois produtos são manufaturados (Produto A e Produto B). Os nós que simbolizam os estados podem representar: matérias-primas, materiais em processo (produtos intermediários) e produtos finais; já os nós que simbolizam das tarefas identificam a seqüência das tarefas (as quais são

habilitadas em diferentes equipamentos, determinando assim as operações de produção) necessárias para a manufatura dos produtos (intermediários ou finais). Nota-se que por meio desta representação todos os materiais necessários para o desenvolvimento de uma operação ficam perfeitamente determinados.

A representação Rede-Estado-Tarefa dada na figura 4.1 é composta das seguintes tarefas: Aquecimento (simbolizada pelo número 1), Reação 1 (simbolizada pelo número 2), Reação 2 (simbolizada pelo número 3), Reação 3 (simbolizada pelo número 4) e uma Separação (simbolizada pelo número 5). Para cada tarefa são habilitados equipamentos, onde as mesmas sofrem modificações físico-químicas. Neste exemplo, a planta é provida dos equipamentos descritos na tabela 4.1.

Tabela 4.1: Dados estruturais da planta

Equipamentos	Tarefas Convenientes	Capacidade de Produção (kg)
Trocador de Calor	1	100
Reator 1	2,3 e 4	80
Reator 2	2,3 e 4	50
Separador	5	200

Como mostra a figura 4.1, a receita de produção determina a quantidade em termos percentuais de matérias-primas utilizadas no desenvolvimento de cada tarefa. Cada uma produz materiais em um ou mais estados e cada estado é formado por parte (percentual) do material que está em processamento no equipamento. Tomando a tarefa 2 (Reação 2), como exemplo, esta necessita dos seguintes materiais para iniciar o seu processamento: 40 % de A aquecido (proveniente do estado intermediário HotA), 60% de intermediário BC (proveniente do estado IntBC). Por sua vez, esta mesma tarefa gera os seguintes materiais: 40% do produto A depois de 2h de processamento (no estado Produto 1), 60% de intermediário AB depois de 2h de processamento (no estado IntAB).

constituirá um novo estado. Já a tarefa 1 gera material em um só estado, o qual é consumido pelas tarefas 2 e 3.

A figura 4.3b mostra uma alternativa de produção, desprovida de qualquer ambigüidade. A tarefa 1 gera produtos intermediários em dois estados diferentes, os quais são utilizados pelas tarefas 2 e 3. O material reciclado da tarefa 5 tem natureza físico-química diferente da outra matéria-prima também consumida pela tarefa 4, constituindo portanto um novo estado.

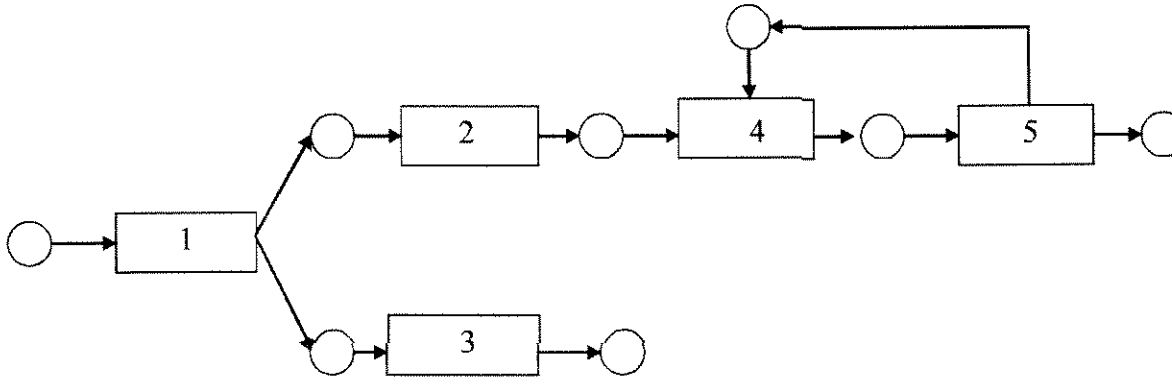


Figura 4.3b: Representação de processos via Tarefa-Estado (caso b) (Kondili et al., 1993)

Nesta seção é utilizada a descrição da representação Rede-Estado-Tarefa para uma planta específica, mas isto não limita o seu uso para outras estruturas mais ou menos complexas.

4.3. Modelagem de Sistemas Flexíveis de Produção via conceito Rede-Estado-Tarefa.

Quando se parte para a solução de problemas de programação de produção via formulação matemática, duas restrições são básicas para a modelagem:

a) A restrição de ordenamento das operações de diferentes tarefas em cada equipamento;

b) A restrição de precedência tecnológica, determinando assim que uma operação de uma determinada tarefa só pode ser desenvolvida depois do término da operação anterior. De acordo com a figura 4.1, a reação 2 só pode ser iniciada depois que

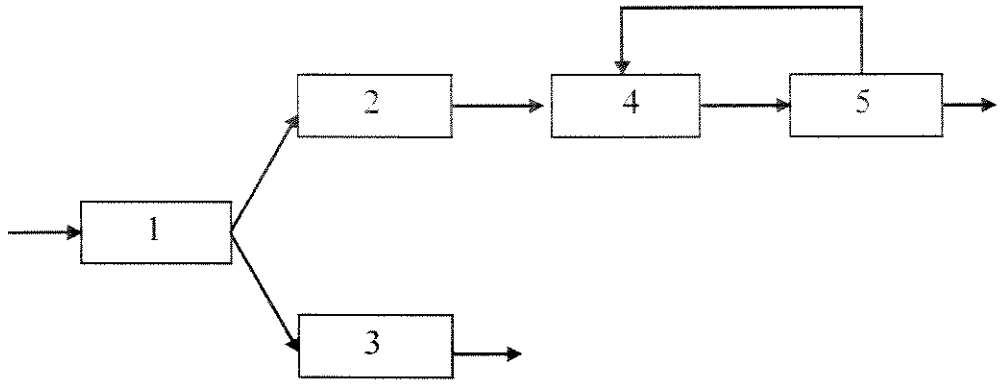


Figura 4.2: Representação de processos de manufatura via tarefas (Kondili et al., 1993)

A utilização dos conceitos estabelecidos pela representação Rede-Estado-Tarefa evita ambigüidades na descrição gráfica de estruturas de processamento. Observando a figura 4 2, não se pode afirmar se a tarefa 1 gera um ou dois materiais intermediários, o qual ou os quais são consumidos pelas tarefas 2 e 3. Outra deficiência nesta representação, é estabelecer se a tarefa 4 consome uma ou duas matérias-primas, ou seja, se o material reciclado da tarefa 5 tem a mesma natureza físico-química do material produzido pela tarefa 2. Estas incertezas só podem ser esclarecidas de posse das receitas detalhadas de produção de cada tarefa. Graficamente, a representação Rede-Estado-Tarefa evita tais deficiências.

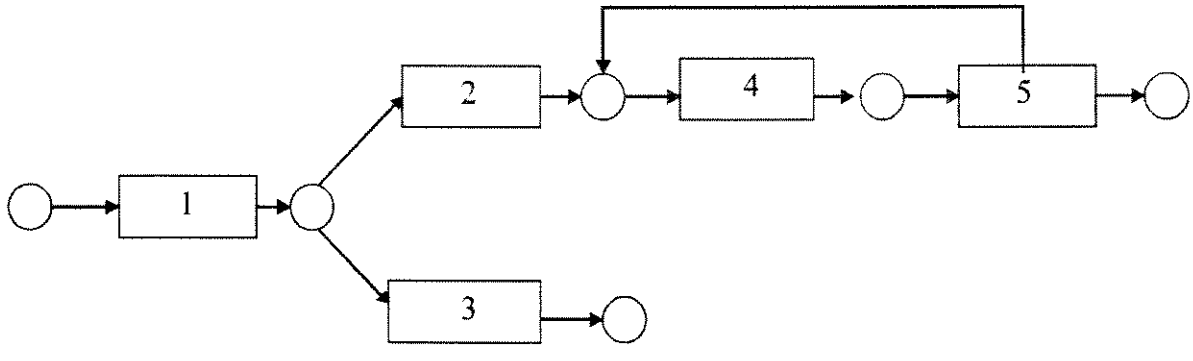


Figura 4.3a: Representação de processos via Rede-Estado-Tarefa (caso a) (Kondili et al., 1993)

Através da figura 4.3a, nota-se que o material reciclado da tarefa 5 é do mesmo estado daquele produzido pela tarefa 2, então podem ser misturados. Caso este material reciclado não tenha a mesma natureza físico-química, ter-se-á duas matérias-primas distintas que serão consumidas pela tarefa 4, ou seja, o material reciclado

alguma quantidade de HotA for produzida, ou seja, depois que a operação de Aquecimento for desenvolvida.

Uma consideração básica em qualquer formulação direcionada à programação de produção refere-se à representação do tempo. Este pode ser modelado de modo contínuo (Pinto e Grossmann, 1994,1995) ou por meio de uma representação discreta. A modelagem proposta por Kondili et al. (1993) utiliza a discretização uniforme do tempo. Neste caso, o horizonte de produção (H) estipulado para o término de toda a produção é dividido em um número de intervalos de igual duração, denominados de período de tempo ou "slots". A duração de cada período de tempo é equivalente ao máximo divisor comum dos tempos de processamento das operações, e por consideração do modelo cada operação só pode iniciar ou terminar seu processamento no início de cada período de tempo. A figura 4.4 mostra a representação desta discretização.

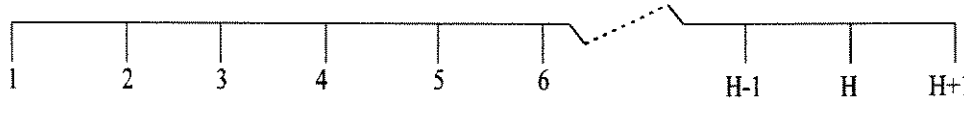


Figura 4.4: Discretização do horizonte de produção

O final do horizonte de produção é tomado como H+1, neste período de tempo é impossilibitado qualquer início de alguma operação e todas as operações devem ter seu processamento finalizado, no máximo no período de tempo H+1.

Nesta modelagem, a restrição que garante a não sobreposição de operações (não-coexistência de operações) em um determinado equipamento é proposta por Shah et al. (1993). Esta restrição envolve variáveis binárias de alocação.

$$\sum_{i \in I_j} \sum_{k'=k}^{k-TP_i+1} W_{ijk'} \leq 1 \qquad \forall\, j,k \tag{4.1}$$

Onde:

i: índice para as tarefas

j: índice para os equipamentos

k e k': índices de período de tempo

TP_i: tempo de processamento da tarefa i

I_j: conjunto de tarefas habilitadas ao equipamento j

$$W_{ijk} = \begin{cases} 1 & \textit{se a tarefa i inicia o processamento no equipamento j no periodo de tempo k} \\ 0 & \textit{se caso contrario} \end{cases} \qquad (4.2)$$

A variável binária $W_{i\,j\,k}$ determina ou não a alocação de uma tarefa i em um equipamento j no início do período de tempo k.

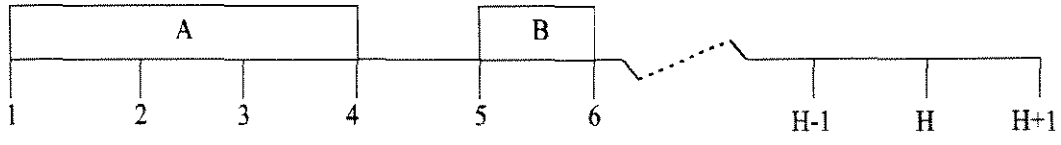


Figura 4.5: Representação da alocação das tarefas A e B

A figura 4.5 mostra uma representação resultante da equação 4.1, onde a tarefa A (com tempo de processamento p_A=3), em um determinado equipamento, começa seu processamento no período de tempo k=1 e termina no período de tempo k=4. Neste período, por restrições físico-químicas nenhuma outra operação pode desenvolvida antes que a mesma termine o seu processamento.

A restrição de precedência tecnológica, que determina o ordenamento das diversas operações de uma tarefa, é garantida nesta formulação por meio de uma equação de balanço material em torno de cada estado em cada período de tempo, como é mostrado a seguir.

$$S_{sk} = S_{sk-1} + \sum_{i \in To_s} qo_{is} \sum_{j \in J_i} B_{ij(k-TP_{is})} - \sum_{i \in Ti_s} qe_{is} \sum_{j \in J_i} B_{ijk} \qquad \forall s,k \qquad (4.3)$$

Onde:

s: índice de os estados.

S_{sk}: quantidade de material no estado s estocada no início do período de tempo k.
B_{ijk}: quantidade de material da tarefa i em processamento no equipamento j no início do período de tempo k..
qo_{is}: percentagem de material da tarefa i que gera material no estado s
qe_{is}: percentagem de material da tarefa i proveniente do estado s.
TP_{is}: tempo de processamento da tarefa i que produz material no estado s.
TP_i: tempo de processamento da tarefa i (max Tp_{is})
J_i: conjunto de equipamentos habilitados para o desenvolvimento da tarefa i.
Ti_s: conjunto de tarefas que consomem material no estado s.
To_s: conjunto de tarefas que geram material no estado s.

A equação 4.3 considera que nenhum produto é vendido e nenhuma matéria-prima é fornecida uma vez iniciada o programa e produção. Para contemplar estas características pragmáticas no cenário industrial, deve-se proceder uma modificação na equação 4.3, adicionando os parâmetros ou variáveis R_{sk} e D_{sk}.

$$S_{sk} = S_{sk-1} + \sum_{i \in To_s} qo_{is} \sum_{j \in J_i} B_{ijk-TP_{is}} - \sum_{i \in Ti_s} qe_{is} \sum_{j \in J_i} B_{ijk} + R_{sk} - D_{sk} \quad \forall s,k \qquad (4.4)$$

Onde:

R_{sk}: Quantidade de material no estado s recebida de fornecedores no período de tempo k.
D_{sk}:Quantidade de material no estado s vendida no período de tempo k

As condições iniciais de estoque de cada estado s no início da produção devem ser conhecidas.

A representação gráfica da equação 4.3 pode ser vista na figura 4.6. Nota-se que a operação A-misturador (com duração de 3h) é iniciada no período de tempo k=2, com isso a segunda operação da tarefa A, que é a reação 2 só pode ser iniciada depois do término da operação A-reação 1 (no período de tempo k=5). A mesma análise pode ser extrapolada para as demais tarefas e suas operações. O horizonte de produção considerado nesta representação foi de 9h.

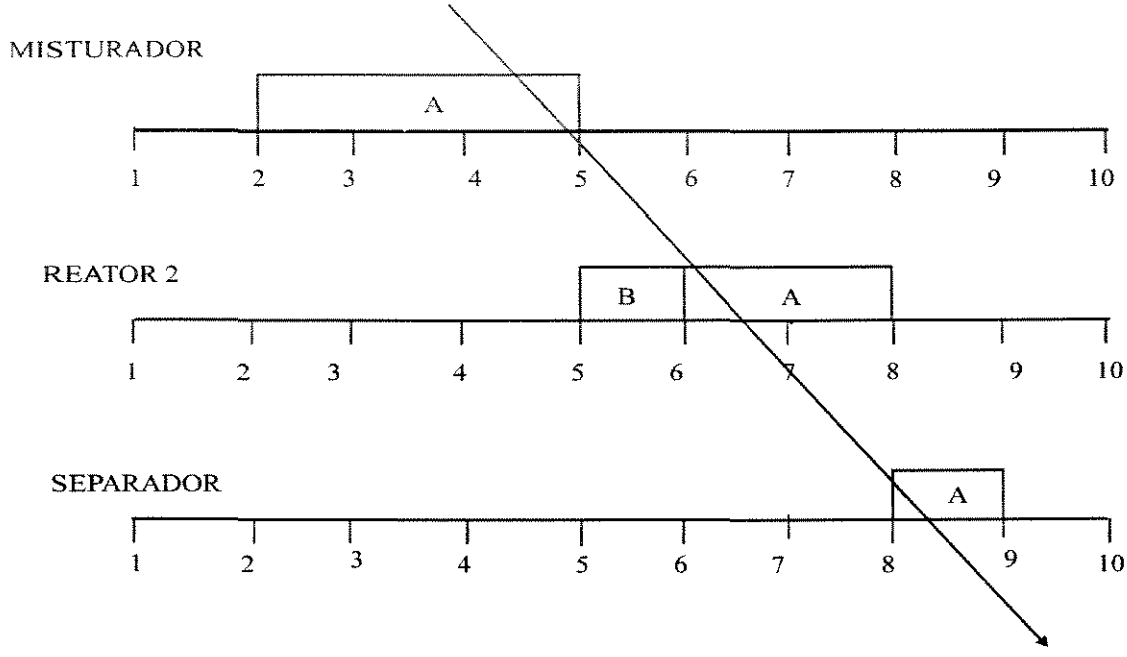


Figura 4.6: Restrição de precedência tecnológica

A quantidade de material a ser processada em um determinado equipamento é limitada pela capacidade deste equipamento.

$$W_{ijk}Vmin_{ij} \leq B_{ijk} \leq W_{ijk}Vmax_{ij} \qquad \forall\, i, j \in J_i \tag{4.5}$$

A quantidade de material a ser processada em um determinado equipamento deve observar a disponibilidade de armazenagem para o respectivo material.

$$0 \leq S_{sk} \leq C_s \qquad \forall\ s, k \tag{4.6}$$

Onde:

C_s: Capacidade de armazenagem disponível para o estado s

Além dos equipamentos, as tarefas compartilham outros recursos, tal como utilidades (vapor, eletricidade, água de refrigeração) e operadores ("*manpower*"), os quais são denominados de recursos compartilhados. Neste caso, deve-se acrescentar ao modelo restrições que contemplem este compartilhamento de recursos durante o

horizonte de produção. O consumo de recursos em um determinado período k é dado por:

$$Q_{kr} = \sum_{i} \sum_{j \in J_i} \sum_{k'=k-TP_i+1}^{k} CR_{ijr} W_{ijk'} \qquad \forall\ k, r \qquad (4.7)$$

Onde:

Q_{kr}: quantidade total de recurso r necessária no período de tempo k;
CR_{ij}: quantidade de recurso r exigida pela tarefa i quando é desenvolvida no equipamento j;

Em muitas estruturas de produção, os recursos existem em quantidade limitada e, com isso é necessário a formalização desta restrição. A limitação de consumo é dado pela a expressão abaixo:

$$Q_{kr} \leq Of_r \qquad \forall\ k, r \qquad (4.8)$$

Onde:

of_r: quantidade de recurso r disponível na planta

Como veremos posteriormente, esta restrição sobre recursos compartilhados contribui de modo significativo para o aumento de complexidade de solução dos problemas, e com isso o problema de programação que já é considerado como "hard problem" torna-se muito mais árduo de ser resolvido. Este aspecto é analisado quando da apresentação do sistema desenvolvido neste trabalho.

As restrições envolvendo recursos compartilhados podem ser melhor vistas pela figura 4.7 para um determinado programa de produção. Nota-se que em nenhum instante da produção a oferta de mão-de-obra (9 Homens) é violada, isto é garantido pela inequação 4.8.

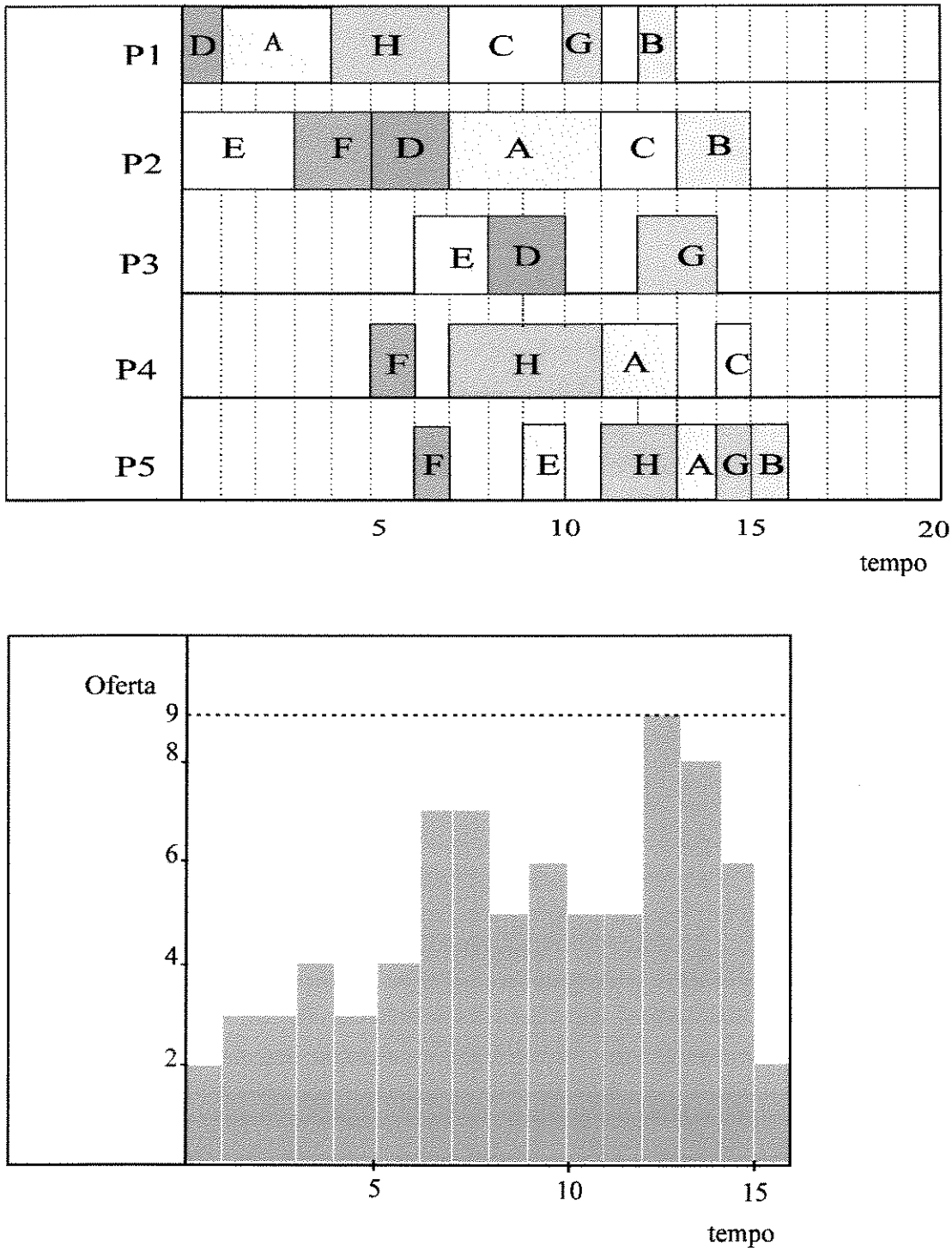


Figura 4.7: Distribuição de necessidade de mão-de-obra

Em muitos problemas no cenário industrial, o tempo para preparar um equipamento depende da ordem em que as operações são executadas, influenciando assim no custo de execução do programa de produção. O seqüenciamento com tempos de estabelecimento é comum em plantas batelada, as quais por natureza manufaturam

vários produtos, como por exemplo: indústrias de tintas, detergentes, óleos e indústria têxtil. Nestes setores de produção os tempos de preparação de máquina são significantes e devem ser considerados no tratamento do problema. Como mostrado em Egli e Rippin (1986) estes tempos tem uma parcela muito grande no custo de produção de uma planta.

Quando a estrutura da planta possui um único processador e os tempos de estabelecimento dos equipamentos dependem da sequência de produção, a minimização do tempo total de execução das tarefas equivale a minimização do tempo total de preparação (Baker 1974, Santos 1994), uma vez que o somatório dos tempos de processamento das tarefas é constante. Com esta característica, o problema torna-se semelhante ao problema clássico da otimização que é o Problema do Caixeiro Viajante ("TSP", "Traveling Salesman Problem").

Na formulação proposta Kondili et al. (1993) duas alternativas são apresentadas para o tratamento desta característica de produção. A primeira delas estabelece que, uma vez alocada uma operação, o tempo para preparar o equipamento para o desenvolvimento de outra operação fique garantido.

$$\sum_{i' \in I_j} \sum_{k'=k+TP_i}^{k+TP_i+TE_{ii'j}-1} W_{i'jk'} \leq G(1 - W_{ijk}) \qquad \forall\ j,i \in I_j,k \tag{4.9}$$

Onde:

$TE_{ii'j}$: tempo de preparação do equipamento j necessário para iniciar a tarefa i' depois da produção da tarefa i.
G: número positivo grande.

Esta restrição só se torna ativa quando ocorre alocação de uma operação, ou seja, quando algum W_{ijk} é igual a 1. Se durante a busca em árvore nós são gerados a partir de ramificações que fixam valores de W_{ijk}=0, a restrição 4.9 constitui uma restrição pouco eficiente na construção do hiperplano de restrições. Quando variáveis binárias são fixadas em 0, a referida restrição é transposta para:

$$\sum_{i' \in I_j} \sum_{k'=k+TP_i}^{k+TP_i+TE_{ii'j}-1} W_{i'jk'} \leq G \qquad \forall\ j,i \in I_j,k \tag{4.10}$$

Na expressão 4.10, tem-se no lado esquerdo um somatório de variáveis que podem assumir valores 0 ou 1 e no lado direito tem-se um número inteiro positivo grande, que mesmo podendo ser ajustado de acordo com os dados do problema, não contribui para a solução do subproblema.

No problema representado pelos dados apresentados nas tabelas 4.2 e 4.3 que busca a minimização do "makespan", a restrição 4.9 pode ser vista na figura 4.8, que representa um programa de produção parcial com alocação da tarefa A no misturador no período de tempo k=7.

Tabela 4.2: Tempos de Preparação das Operações

Tarefas	Misturador	Reator	Separador
A	4	2	2
B	2	3	7
C	3	1	1

Neste exemplo, os tempos de preparação são considerado os mesmos para os 3 equipamentos da planta.

Tabela 4.3: Tempos de Preparação das operações

Tarefas	A	B	C
A	-	-	3
B	1	-	-
C	-	-	-

Uma vez alocada a tarefa A no misturador (alocação em k=7), todas as variáveis binárias que representam a alocação da tarefa de C no misturador do período de tempo k=11 (que é o período de alocação de A, k=7, mais o tempo de processamento de A no misturador, $TP_{A1} = 4$) até o período $k = 11 + TE_{AC} = 13$ são levadas a 0 (onde $TE_{AC}=3$), ou seja, não fazem mais parte do conjunto de variáveis candidatas à ramificação (são levadas ao valor 0), garantindo assim que a tarefa C só poderá ser desenvolvida a partir do período de tempo k=14, que é quando o equipamento estará pronto para recebê-la.

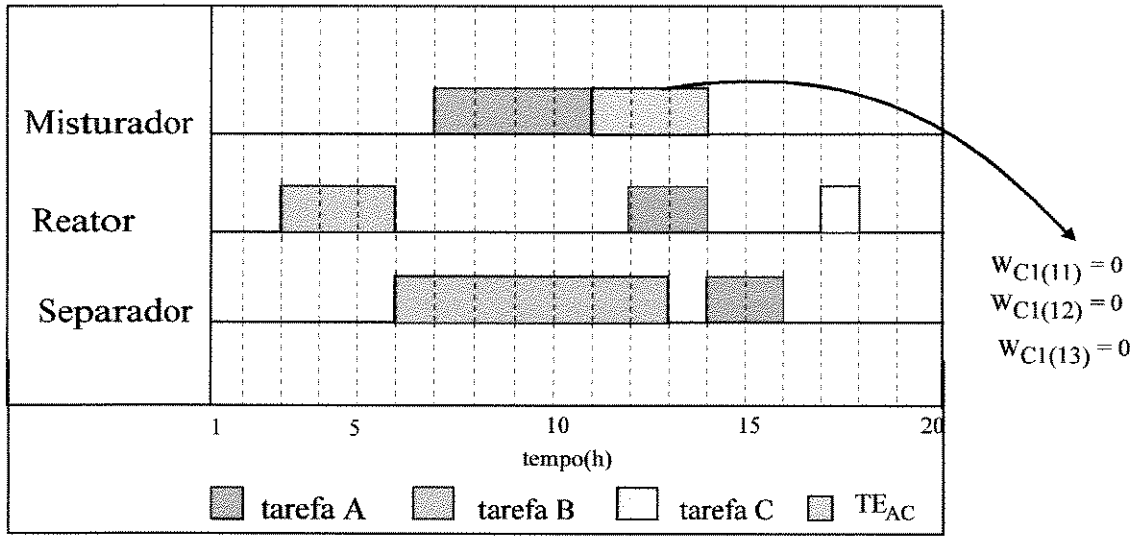


Figura 4.8: Representação da restrição de tempo de preparação

A segunda alternativa apresentada por Kondili et al. (1993) para o tratamento de tempos de estabelecimento leva em conta que uma atividade improdutiva (como por exemplo uma operação de limpeza de um equipamento) necessita de utilidades (como por exemplo água) e de mão-de-obra para o seu desenvolvimento. Com isso é necessário um tratamento diferenciado daquele visto anteriormente, que procura tão somente garantir o tempo de preparação necessário entre duas operações, não contemplando os recursos necessários para desenvolvê-las. Para levar em conta esta característica de consumo de recursos por parte das atividades de preparação de equipamentos, estas atividades são tratadas como uma operação produtiva, ou seja, suceptível de alocação, e com isto pode-se considerar as quantidades de recursos exigidas por tais atividades. A expressão 4.11 contempla este detalhe de processamento.

$$\sum_{k=k_1+1}^{k_2-1} W_{jF_1F_2k} \geq \sum_{i \in I_j^{F_1}} W_{ijk_1} + \sum_{i \in I_j^{F_2}} W_{ijk_2} - \sum_{k=k_1+1}^{k_2-1} \sum_{i \in I_j} W_{ijk} - 1 \qquad \forall\, j, F_1, F_2, k_1, k_2 > k_1 \tag{4.11}$$

Onde:

F_1: conjunto 1 de tarefas com características em "setup" semelhantes

F_2: conjunto 2 de tarefas com características em "setup" semelhantes

$W_{jF_1F_2k}$: variável de alocação da operação ("setup") entre a família de produtos F_1 e F_2

O exemplo mostrado a seguir é solucionado aplicando o modelo descrito anteriormente (expressões 4.1 a 4.3, 4.5 e 4.6). A receita de produção da planta é mostrada na figura 4.1, onde se pode obter as percentagens de materiais necessários para o desenvolvimento de cada tarefa, assim como os tempos de processamento das operações. O problema consiste em maximizar o lucro de produção, dado pela equação abaixo:

$$Lucro = \sum_{s} PP_{sH+1} S_{s,H+1} - \sum_{s} PI_{sH+1} S_{sH+1} \tag{4.12}$$

Onde:

PP_{sH+1}: Preço de produtos no final do horizonte de produção.
PI_{sH+1}: Preço de estoque de material intermediário no final do horizonte de produção.
S_{SH+1}: Estoque de estado s no final do horizonte de produção.

As operações e seus tempos de processamento em cada equipamento são fornecidos pela tabela 4.4.

Tabela 4.4: Tempos de Processamento das Operações

Tarefas	Trocador	Reator1	Reator2	Separador
Aquecimento	1	-	-	-
Reação 1	-	2	2	-
Reação 2	-	2	2	-
Reação 3	-	1	1	-
Separação	-	-	-	2

A tabela 4.5 mostra os equipamentos disponíveis na planta, as tarefas que podem ser desenvolvidas nestes equipamentos e a capacidade de cada um.

Tabela 4.5: Dados estruturais da planta

Equipamentos	Tarefas Convenientes	Capacidade de Produção (kg)
Trocador de Calor	aquecimento	100
Reator 1	reação 1,2 e 3	80
Reator 2	reação 1,2 e 3	50
Separador	separação	200

Existe limite na capacidade de armazenagem para alguns estados intermediários, ou seja, a armazenagem não é ilimitada durante o processo produtivo. A tabela 4.6 mostra estes dados.

Tabela 4.6: Limite de capacidade de armazenamento dos estados

Estados	Capacidade de armazenamento (kg)
Cargas A, B e C	*
A aquecido	100
Intermediário AB	200
Intermediário BC	150
Impuro E	100
Produtos A e B	*

(*) armazenagem ilimitada

Neste exemplo, é considerado que recursos existem ilimitadamente, isto equivale a não utilização das expressões 4.7 e 4.8. O Preço dos produtos (PP) A e B é de 10 R$/kg e o preço de armazenamento de matérias intermediários (PI) no final do horizonte de produção é de 1R$/kg.

O programa de produção ótimo para este problema é mostrado na figura 4.9. O número que aparece acima de cada operação é a quantidade de material em processamento no equipamento (B_{ijk}).

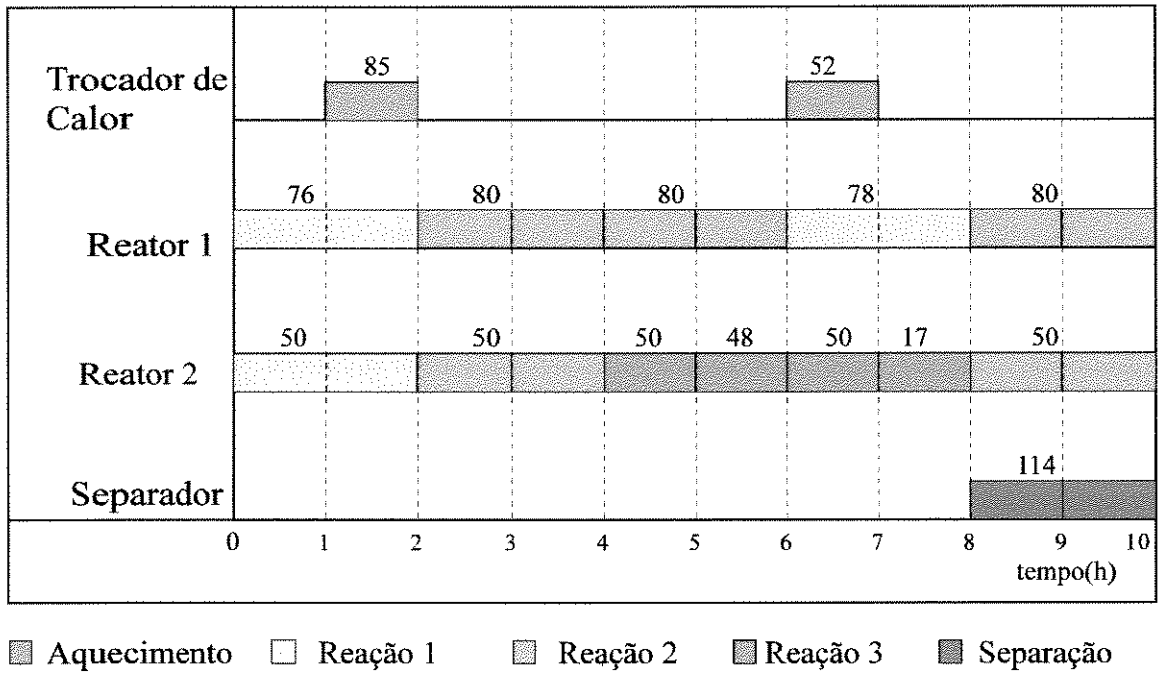


Figura 4.9: Programa de produção ótimo do exemplo proposto

O problema é solucionado utilizando o GAMS/OSL e o valor ótimo do lucro é de R$ 2744,4. As quantidades ótimas de produtos A e B encontradas foram de 136,0 kg e 47,4 kg respectivamente.

4.4. Sequenciamento de tarefas em Plantas Multiproduto via representação Rede-Estado-Tarefa

O modelo a seguir é uma aplicação do conceito Rede-Estado-Tarefa no sequenciamento de produção em plantas multiproduto, ou seja, nesta adaptação busca-se somente a melhor seqüência de produção das tarefas, nenhuma referência às quantidades de materiais em processamento nos equipamentos é considerada. No exemplo aqui tratado não é considerada limitação na oferta de recursos e a armazenagem de intermediários é ilimitada. O horizonte de tempo considerado neste problema é de 30h e o objetivo é minimizar o tempo de execução ("makespan") das tarefas. Os dados de tempos de processamento das tarefas são mostrados na tabela 4.7.

Tabela 4.7: Tempos de Processamento das tarefas

Tarefas	Misturador	Reator	Separador
A	4	5	2
B	2	3	7
C	3	5	4

A representação Rede-Estado-Tarefa para esta estrutura de processamento é mostrado na figura 4.10. Nota-se neste exemplo, que os produtos seguem linhas de produção distintas no que diz respeito às receitas de produção. Neste caso, o único recurso da planta compartilhado pelas operações são os equipamentos. Os materiais intermediários gerados pelas operações de cada tarefa não são utilizados pelas outras tarefas, assim como as matérias-primas também não são compartilhadas.

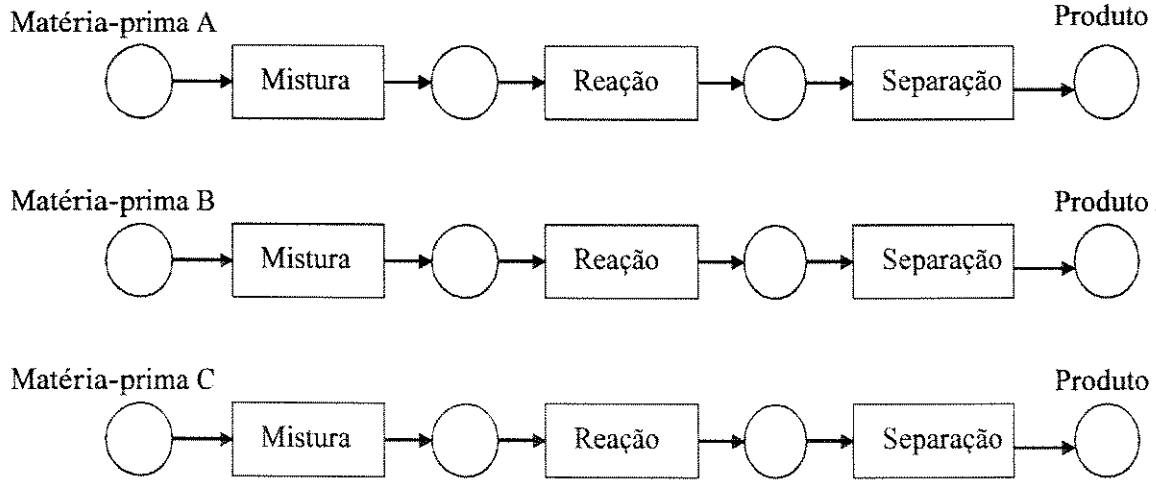


Figura 4.10: Representação Rede-Estado-Tarefa de uma Planta Multiproduto

Nesta transposição da modelagem anteriormente apresentada, adequada para a programação de produção em plantas com estruturas de processamento mais complexas, para o sequenciamento em plantas multiproduto, a restrição que garante a não sobreposição de operações em um determinado equipamento é dada por:

$$\sum_{i} \sum_{k'=k}^{k-TP_{ij}+1} W_{ijk'} \leq 1 \qquad \forall\, j,k \qquad (4.13)$$

Onde:

Tp_{ij}: tempo de processamento da tarefa i no equipamento j

A restrição 4.14 determina que só haja uma alocação de cada tarefa em cada equipamento.

$$\sum_{k=1}^{H+1} W_{ijk} = 1 \qquad \forall i, j \tag{4.14}$$

A restrição dada pela expressão 4.3, de precedência tecnológica, é transposta para este problema de sequenciamento segundo as expressões 4.14 a 4.16. As bateladas de materiais, B_{ijk}, são substituídas pelas variáveis binárias de alocação W_{ijk}. As percentagens de entrada e saída de material dos estados são consideradas iguais a 1 (100%).

$$S_{ijk} = S_{ijk-1} + W_{i(j-1)(k-TP_{ij-1})} - W_{ijk} \qquad \forall i, j, k \tag{4.15}$$

No início da produção (k=1) para o primeiro equipamento da planta (Misturador), a equação 4.15 reduz-se a:

$$S_{i(Misturador)1} = 1 - W_{i(Misturador)1} \qquad \forall\, i \tag{4.16}$$

As expressões 4.15 e 4.16 consideram que a quantidade de material processada (batelada) de cada operação tem valor unitário, ou seja, neste problema a quantidade de material processada não é relevante, pois procura-se somente sequenciar as tarefas, sem preocupação de atender a uma demanda de mercado.

Completando as restrições de ordenamento de tarefas, o balanço de matéria-prima de cada tarefa é dado pela equação 4.17. Nota-se que esta equação há somente saída de material.

$$S_{i(Misturador)k} = S_{i(Misturador)k-1} - W_{i(Misturador)k} \qquad \forall\, k > 1 \tag{4.17}$$

O consumo de recursos em um determinado período de tempo é dado por:

$$Q_{kr} = \sum_{i} \sum_{j} \sum_{k'=k-TP_{ij}+1}^{k} CR_{ijr} W_{ijk'} \qquad \forall\ k, r \tag{4.18}$$

Onde:

Q_{kr}: quantidade total de recurso r necessária no período de tempo k;
CR_{ijr}: quantidade de recurso r exigida pela tarefa i quando é desenvolvida no equipamento j;
Tp_{ij}: tempo de processamento da tarefa i no equipamento j

A limitação de consumo é dado pela a expressão abaixo:

$$Q_{kr} \leq Of_{r} \qquad \forall\ k, r \tag{4.19}$$

Considerando como função de desempenho o tempo total de execução das tarefas (Mk), tem-se:

$$Mk_{i} \geq \sum_{k} W_{ijk} \cdot k + TP_{ij} - 1 \qquad \forall\ i, j = M \tag{4.20}$$

Onde:

Mk_i: tempo de produção da tarefa i

M: número de equipamento

s existentes na planta

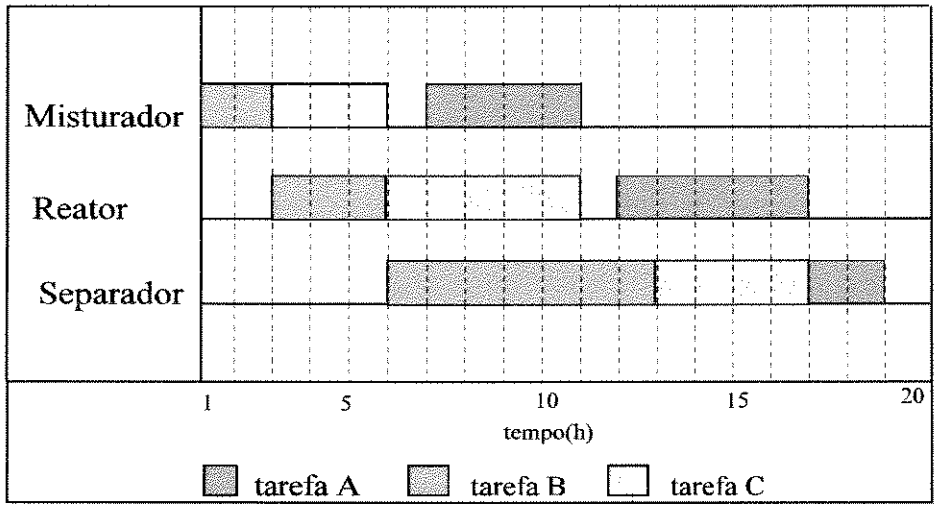


Figura 4.11: Programa de produção ótimo

Utilizando as expressões 4.13 a 4.17, 4.20 e a tabela 4.7, o programa de produção que minimiza o tempo de execução das tarefas é mostrado na figura 4.11. O modelo foi solucionado utilizando o GAMS/OSL.

4.5. Sequenciamento de tarefas em Plantas Multiproduto com Processadores em Paralelo

No modelo apresentado anteriormente que é usado para o sequenciamento de tarefas em plantas multiproduto, considerou-se que todas as tarefas são habilitadas em todos os equipamentos. No caso de planta com processadores em paralelo e que nem todas as tarefas sejam habilitadas a todos equipamentos, é necessário algumas reformulações nas restrições vistas anteriormente. Em cada estágio de produção (que pode comportar vários equipamentos em paralelos), cada operação só pode ser processada em um dos equipamentos. Esta característica de produção é dada abaixo:

$$\sum_{\substack{j \in E_j \\ j \in J_i}} \sum_{k} W_{ijk} = 1 \qquad \forall \quad i, E \tag{4.21}$$

Onde:

E: índice de estágios de processamento

E_j: conjunto de equipamentos em paralelo no estágio E

Adotando-se o tempo de execução total das tarefas como função, tem-se:

$$Mk_i \geq \sum_{\substack{j \in E_j \\ j \in J_i}} \sum_k W_{ijk} \cdot k + TP_{ij} - 1 \qquad \forall \; i, E = NE \tag{4.22}$$

Onde:

NE: número de estágios de processamento

As outras restrições necessárias à modelagem segue as expressões 4.1 (de não coexistência de operações em um determinado equipamento), 4.2 (de definição da variável de alocação), 4.3 (de ordenamento de operações de cada tarefa). Na equação 4.3 os B_{ijk} são substituídos pelos W_{ijk}, pois o que se busca é apenas o sequenciamento das tarefas. As percentagens de entrada (qo_{is}) e saída dos estados (qe_{is}) são consideradas 1 e o tempo TP_{is} (tempo de processamento da operação i para produzir material no estado s) é equivalente ao tempo da operação (TP_{ij}). As condições iniciais dos estados das matérias-primas assumem valores iguais a 1.

4.6. Considerações Finais sobre a formulação Rede-Estado-Tarefa

A formulação apresentada por Kondili et al. (1993) contempla algumas outras restrições que são abordadas nesta seção.

a) Indisponibilidade Temporária de equipamentos

Durante o desenvolvimento do programa de produção, alguns equipamentos podem ficar indisponíveis em certos períodos de tempo, devido à manutenção programada da planta. Esta característica de processamento é facilmente incorporada nesta formulação fixando todas as variáveis de alocação de operações naquele período de tempo em 0, ou seja, impossibilita que durante o processo de busca em árvore tais variáveis adquiram valor 1.

$$W_{ijk} = 0 \qquad \forall\, i \in I_j\ ,\ k = k_1 - p_i + 1,\ldots,k_2 - 1 \tag{4.23}$$

Por exemplo, considere o problema representado pela tabela 4.7, se fosse dado no problema que na sétima hora até a oitava (k_1=7 até k_2=8) de produção seria feita uma manutenção no misturador, variáveis binárias mostradas na tabela 4.8 deveriam ser fixadas em 0, segunda a equação 4.23.

Tabela 4.8: Variáveis de alocação fixadas em 0

Variáveis de alocação
$W_{A(misturador)\,4}=0$
$W_{A\,(misturador)5}=0$
$W_{A\,(misturador)6}=0$
$W_{A\,(misturador)7}=0$
$W_{B(misturador)6}=0$
$W_{B(misturador)7}=0$
$W_{C(misturador)6}=0$
$W_{C(misturador)7}=0$

Neste caso a operação (A,misturador), por exemplo, não poderia estar sendo desenvolvida no período de tempo k=7, como mostra a figura 4.8.

b) Consumo de recursos dependente da quantidade de material em processamento

As operações a serem desenvolvidas durante a programação de produção necessitam consumir recursos compartilhados. A demanda de recurso exigida por cada operação pode variar ao longo do processamento de acordo com a quantidade de material em processo, assim como a oferta deste recurso. As equações 4.24 e 4.25 representam a quantidade total de recurso r exigida no período de tempo k e a limitação desta quantidade demandada respectivamente.

$$Q_{rk} = \sum_{i} \sum_{j \in J_i} \sum_{\theta=0}^{TP_i - 1} (\alpha_{ri\theta}\, W_{ijk-\theta} + \beta_{ri\theta}\, B_{ijk-\theta}) \qquad \forall\ r,k \tag{4.24}$$

$$Q_{rk} \leq Of_{rk} \qquad \forall\ r,k \tag{4.25}$$

Onde:

Q_{rk}: quantidade de recurso r consumida no período de tempo k
J_i: conjunto de equipamentos convenientes para processar a tarefa i
$\alpha_{ri\theta}$: indica quanto de recurso r é consumido no período de tempo θ quando a tarefa i está sendo processada.
$\beta_{ri\theta}$: indica quanto de recurso r é consumido no período de tempo θ por unidade de material processado da tarefa i.
Of_{rk} : Quantidade de recurso r disponível no período k.

A equação 4.24 expressa que o consumo do recurso r por todas as operações em processamento no período de tempo k. O somatório de θ de um até o período de tempo TP_i - 1 garante em qualquer período de tempo k são consideradas todas as operações em processamento neste período de tempo.

No caso de recursos discretos, cujo consumo independe das quantidades que estão sendo processada, tal como mão-de-obra, os parâmetros $\beta_{ri\theta}$ são tomados como 0 e os $\alpha_{ri\theta}$ são inteiros.

4.7. Conclusão

A principal característica dos problemas de programação de produção é a natureza combinatorial do espaço de soluções factíveis e à medida em que detalhes de processamento, tais como política de armazenagem intermediária, tempos de preparação, tempos de transferência de material, datas de entrega, limitação de recursos compartilhados vão sendo contemplados na modelagem, esta pode tornar-se de difícil solução.

Este capítulo teve, portanto, como objetivo mostrar as restrições de produção consideradas nos exemplos solucionados neste trabalho. A formulação via discretização uniforme de tempo apresentada para o tratamento de problemas de programação de produção tem como principais características a não complexidade de representação das características de processamento e a linearidade das expressões propostas, mesmo para características de produção consideradas de extrema

complexidade, como é o caso das restrições que garantem o sequenciamento dependente do tempo de preparação ("setup time") e as restrições sobre recursos compartilhados. Como no trabalho apresentado por Pinto e Grossmann (1994), as restrições sobre recursos compartilhados foram formuladas de modo não-linear, havendo necessidade de uma linearização das mesmas, através de introdução de novas variáveis binárias. No entanto, até o momento, no cenário de utilização de programação matemática como metodologia para solucionar problemas de programação de produção, todo e qualquer modelo traz suas vantagens e desvantagens, alguns são mais representativos e menos complexos em sua formulação que outros. Por exemplo, uma desvantagem da formulação com horizonte de produção discretizado, como apresentada neste capítulo, é o elevado número de variáveis binárias de alocação necessário, e portanto a exigência de um horizonte longo pode inviabilizar o uso desta pela quantidade de variáveis binárias geradas no modelo. Portanto, se não for utilizada nenhuma estratégia que auxilie os procedimentos padrões dos pacotes destinados à solução desses problemas, o tempo computacional para se alcançar a solução pode ficar comprometido.

A utilização de uma abordagem de programação matemática linear-mista no tratamento de problemas de programação de produção exige a formulação rigorosa do problema. A principal dificuldade associada ao uso desta metodologia é a necessidade, de representar as características do problema de forma linear. A presença de expressões não-lineares no modelo nos obrigam a utilizar técnicas de linearizações, o que pode comprometer a solução do problema.

CAPÍTULO 5

ANÁLISE LÓGICA NO CONTEXTO DA PROGRAMAÇÃO MATEMÁTICA

5.1. Introdução

A pesquisa de métodos que venham diminuir a complexidade de solução de modelos matemáticos é extremamente importante, pois favorece a solução de problemas mais realísticos, que no caso de programação de produção equivale dizer que se pode considerar no problema um maior número de operações (habilitação de uma tarefa em um determinado equipamento), além de possibilitar a solução de problemas com maior nível de detalhamento do processo produtivo (consideração de armazenagem intermediária, limitação de recursos compartilhados, tempos de preparação de equipamentos).

No cenário de problemas de otimização com satisfação de restrições, dois parâmetros de avaliação do procedimento de solução devem ser sempre analisados:

- Qualidade do valor da função objetivo;
- Tempo computacional para obter a solução do problema.

Qualquer que seja a metodologia de solução a ser utilizada, sempre deve-se ter em mente estes dois parâmetros. Neste capítulo, é formalizado o uso de análise de proposições lógicas em problemas de programação matemática, constituindo assim a metodologia de programação linear inteira-mista baseada em lógica (Hooker, 1994). A essência desta estratégia é de combinar elementos lógicos dentro do contexto de programação matemática e para melhor compreendê-la faz-se necessário a análise de elementos pertencentes à análise de lógica clássica.

5.2. Representação de Proposições Lógicas via Conjunção e Disjunção

Duas formas podem ser utilizadas para a representação de uma expressão lógica e que posteriormente são utilizadas no processo de busca "Branch and Cut". Na apresentação destas formas considere a expressão abaixo:

$$x_1 \vee (x_2 \wedge x_3) \tag{5.1}$$

Aplicando a propriedade distributiva (apêndice A) nos termos da expressão 5.1, tem-se:

$$(x_1 \vee x_2) \wedge (x_1 \vee x_3) \tag{5.2}$$

No apêndice A é apresentada uma síntese das principais propriedades envolvendo proposições lógicas. Com isso tem-se a Forma Normal via Conjunção ("Conjuctive Normal Form", "CNF") da expressão 5.1, que é uma conjunção das cláusulas $(x_1 \vee x_2)$ e $(x_1 \vee x_3)$. A forma geral de uma expressão lógica na forma de conjunção é dada pela expressão 5.3.

$$P_1 \wedge P_2 \wedge ... \wedge P_n \tag{5.3}$$

Onde P_i é uma disjunção de literais P_i (i=1,2..., n)

A forma geral de P_i é $\wedge_{j \in J(i)} \pm C_j$,

Onde:
i: índice de cláusulas
j: índice de literais
n: número de cláusulas
s: número de literais
C_j é a j-ésima literal de P_i.
$J(i) \subset \{1,...,s\}$ é o conjunto de literais pertencentes a cláusula i.

Na representação da Forma Normal via Disjunção ("Disjunctive Normal Form", "DNF"), considere a seguinte expressão lógica

$$x_1 \wedge (x_2 \vee x_3) \tag{5.4}$$

Aplicando novamente propriedade distributiva nesta expressão tem-se:

$$(x_1 \wedge x_2) \vee (x_1 \wedge x_3) \tag{5.5}$$

A proposição 5.5 representa a expressão 5.4 na sua forma normal via disjunção. A generalização de uma expressão nesta forma é dada como segue.

$$P_1 \vee P_2 \vee ... \vee P_n \tag{5.6}$$

A forma geral de $P_i = \vee_{j \in J(i)} \pm C_j$

As definições de índices e conjuntos seguem as anteriormente feitas.

Blair et al. (1986) apresentam um procedimento de redução de expressões lógicas à sua forma normal via conjunção. Hooker (1994) também mostra como obter uma inequação a partir de uma proposição na Forma Normal Conjuntiva (FNC).

5.3. Representação Matemática de Proposições Lógicas

No translado de uma expressão lógica para sua forma matemática é necessário primeiramente considerar os operadores lógicos envolvidos na respectiva expressão. A transformação de cada operador em sua representação matemática conduz à forma matemática da proposição lógica. O primeiro passo relacionado ao tratamento de um problema baseado no conhecimento lógico é relacionar restrições na forma matemática a proposições lógicas. Tomando como exemplo a inequação:

$$3x_1 - 2x_2 + x_3 \geq 2 \tag{5.7}$$

Sendo x_1, x_2 e x_3 variáveis binárias. A proposição lógica abaixo é satisfeita pela mesma solução que satisfaz a inequação 5.7.

$$x_1 \wedge (\neg x_2 \vee x_3) \tag{5.8}$$

Este resultado é mostrado na tabela 5.1, onde o valor "1" indica que a expressão lógica é verdadeira e "0" quando a proposição é falsa. Fazendo todas as combinações possíveis para as variáveis binárias, pode-se fazer uma comparação entre a inequação 5.7 e a expressão 5.8.

Tabela 5.1: Equivalência entre uma inequação e uma expressão na forma lógica

x_1	x_2	x_3	$3x_1 - 2x_2 + x_3$	$x_1 \wedge (\neg x_2 \vee x_3)$
0	0	0	0	0
0	0	1	1	0
0	1	0	-2	0
0	1	1	-1	0
1	0	0	3	1
1	0	1	4	1
1	1	0	1	0
1	1	1	2	1

Raman e Grossmann (1991) apresentam equivalências entre relações lógicas e expressões matemáticas, dadas pela tabela 5.2.

Tabela 5.2: Equivalências entre Expressões Lógicas e Expressões matemáticas

Relação Lógica	Designação	Proposição Lógica	Representação Matemática
"OR"	"ou" simbólico	$P_1 \vee P_2 \vee ... \vee P_r$	$y_1+y_2+...+y_r \geq 1$
"AND"	"e" simbólico	$P_1 \wedge P_2 \wedge ... \wedge P_r$	$y \geq 1; y_2 \geq 1; ...; y_r \geq 1$
$P_1 \Rightarrow P_2$	"$\Rightarrow$"Implicação	$\neg P_1 \vee P_2$	$1-y_1+y_2 \geq 1$
$(P_1 \Rightarrow P_2) \wedge (P_2 \Rightarrow P_1)$	Equivalência	$(P_1 \Rightarrow P_2) \wedge (P_2 \Rightarrow P_1)$	$y_1-y_2 \leq 0; y_2-y_1 \leq 0$
$\oplus$	"Ou" exclusivo	$P_1 \oplus P_2 \oplus P ... \oplus P_r$	$y_1 + y_{2+} + yr = 1$

Hooker (1994) apresenta o seguinte procedimento para obter uma expressão lógica na forma normal via conjunção à forma matemática.

$$a_j y \geq 1 + \eta(a) \tag{5.9}$$

Onde cada $a_j \in \{0,1,-1\}$.

O coeficiente a_j é negativo se a literal aparecer com o operador negação ($\neg$). A parcela $\eta(a)$ é calculada pela equação:

$$\eta(a) = \sum_{\substack{j \\ a_j < 0}} a_j \tag{5.10}$$

Como exemplo considere a seguinte expressão lógica apresentado por Raman e Grossmann (1991).

$$(\neg y_1 \vee \neg y_2 \vee y_4 \vee y_5) \wedge (\neg y_3 \vee y_4 \vee y_5) \tag{5.11}$$

Aplicando as equações 5.9 e 5.10 no primeiro termo da expressão 5.11, tem-se:

$$y_1 - y_2 + y_4 + y_5 \geq -1 \tag{5.12}$$

ou

$$y_1 + y_2 - y_4 - y_5 \leq 1 \tag{5.13}$$

enquanto o segundo termo equivale a:

$$y_3 - y_4 - y_5 \leq 0 \tag{5.14}$$

5.4. Considerações sobre Metodologias para aceleramento do processo de solução de problemas com Programação Inteira-Mista.

A busca "branch and bound" está incorporada na maioria dos pacotes destinados à solução de problemas envolvendo variáveis inteiras. A depender das regras heurísticas estrategicamente inseridas em cada pacote pode-se obter uma solução mais rápida. Por exemplo, o OSL incorpora a definição de conjuntos especiais ordenados ("Special Ordered Sets"), que pode acelerar o processo de "branch and bound", a depender das características do modelo. A proposta de utilização de conjuntos ordenados é sugerida por Beale e Tomlin (1970), na qual o processo de ramificação um conjunto de variáveis inteiras é considerado na geração de um novo nó. Na metodologia "branch and bound" padrão somente uma variável inteira é fixada em um de seus limitantes para a geração de um nó. Seguindo esta abordagem Escudero (1988) usa uma extensão da definição apresentada por Beale e Tomlin (1970) em problemas de programação.

Beale e Tomlin (1970) apresentam restrições generalizadas em "upper bound" da forma:

$$\sum_{i \in I} y_i = 1 \tag{5.15}$$

Onde y_i é definida como variável binária. Para exemplo desta proposta considere o seguinte exemplo abaixo.

$$\begin{aligned} min \quad & Z = y_1 + 2y_2 + 3y_3 + 4y_4 \\ st. \quad & y_1 + y_2 - y_3 - y_4 \leq 0 \\ & y_1 + y_2 + y_3 + y_4 = 1 \\ & y_i = 0,1 \quad i = \{1,2,3,4\} \end{aligned} \tag{5.16}$$

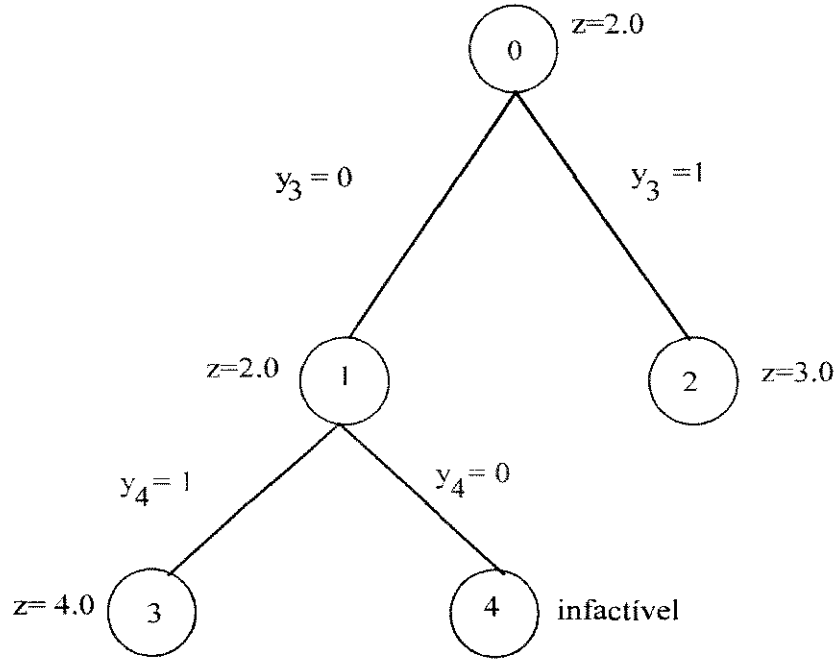


Figura 5.1: "Branch and Bound" com variáveis binárias

A figura 5.1 mostra o desenvolvimento da árvore de busca para o exemplo dado pelo modelo 5.16

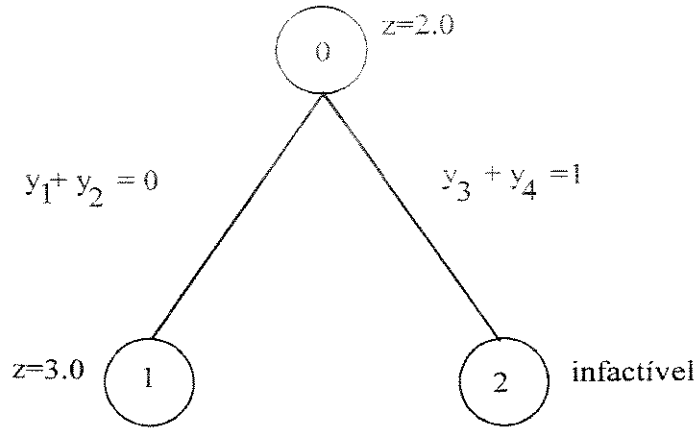


Figura 5.2: "Branch and Bound" com variáveis tipo SOS

Utilizando a definição de conjunto ordenado somente dois nós foram necessários para solucionar o problema, como é mostrado na figura 5.2. Nota-se que, por exemplo, o nó 1 é gerado pela análise simultânea de y_1=0 e y_2=0.

Na utilização de conjuntos ordenados, o problema deve apresentar características adequadas à definição dos mesmos. No manual do OSL (IBM,1992) são apresentados os tipos de conjuntos ordenados que podem ser utilizados como estratégia para o aceleramento de solução do problema. Alguns pacotes como o sistema OSL já permitem a definição de variáveis como conjunto ordenado.

Outra característica importante dos códigos "branch and bound" é a técnica de preprocessamento, a qual procura a fixação de variáveis e eliminação de restrições redundantes antes que o subproblema correspondente a um determinado nó seja resolvido. Uma das subrotinas do sistema OSL (EKKMPRE) têm como objetivo a preprocessamento da árvore "branch and bound" e a fixação de variáveis inteiras em seus limitantes, visando com isso a diminuição do tempo de solução do problema.

Alguns métodos têm sido sugeridos para melhorar o desenvolvimento do processo "branch and bound", tais como: métodos de planos de corte ("cutting planes") e métodos baseados em lógica. A idéia de métodos "cutting planes" é o desenvolvimento de restrições (planos de corte) que limitem o espaço de busca da solução ótima do problema (agem na diferença entre a solução relaxada e a solução inteira). (Applegate e Cook, 1991).

Os métodos baseados em lógica procuram relacionar as variáveis binárias com variáveis boleanas. Balas (1974) apresenta a Programação via Disjunção como alternativa de representação de problemas de Programação Linear Inteira-Mista. Outra

classe métodos baseados em lógica são os que usam inferência lógica durante o processo "branch and bound" (Hooker, 1994).

Raman e Grossmann (1991,1992,1993,1994) incorporam inequações provenientes de expressões lógicas e adiciona-as ao modelo (diminuindo a diferença entre a solução relaxada e a solução inteira do problema, a qual é denominada de "gap"). Outra estratégia sugerida pelo autor, é considerar a análise de expressões lógicas em cada nó da árvore "branch and bound" (inferência lógica). Nesta diretriz, Hooker et al. (1994) formalizam o uso do conhecimento lógico em problemas de síntese de processos.

As relações lógicas entre as variáveis binárias são denominadas de cortes lógicos ("logic cuts"). Na forma matemática um "logic cut" é representado por uma inequação que equivale à proposição na forma lógica. Quando cortes lógicos são incorporados no modelo "MILP" a qualidade da solução do problema relaxado tende a melhorar, além de eliminar do processo soluções inteiras impraticáveis, diminuindo assim o número de nós analisados.

Hooker et al. (1994) definem "Logic Cut" como uma a restrição relacionada com as variáveis inteiras, as quais quando incorporadas ao problema não mudam a solução deste, além de apresentar algumas alternativas de diminuir o custo computacional de problemas via programação linear inteira-mista, tais como:

- Verificação do comportamento de problemas utilizando cortes lógicos na forma de expressões matemáticas;
- Verificação comportamental dos problemas utilizando os cortes lógicos simbolicamente durante a busca "branch and bound" (integralização de conhecimento lógico e conhecimento matemático) - Metodologia "Branch and Cut".
- Adicionar as restrições lógicas violadas na solução relaxada no nó raiz.

No primeiro caso, proposições lógicas envolvendo variáveis binárias são transpostas para a forma matemática e inseridas no modelo original. Obviamente que o número de restrições no modelo aumenta e com isso o tempo computacional para a solução do problema pode também aumentar. No segundo caso, os autores sugerem a utilização de proposições lógicas externamente durante a solução do problema. Uma alternativa para suplantar este obstáculo é a construção de algoritmos de busca em árvore que permitam a análise lógica de proposições durante busca ou a utilização de

bibliotecas que permitam esta interferência externa. No terceiro caso, somente as proposições violadas pela solução relaxada (solução do nó raiz da árvore) são inseridas na forma de expressões matemáticas no modelo.

Buscando a formalização da utilização de lógica no contexto de programação matemática, Raman e Grossmann (1994) propõem o seguinte modelo para problemas de otimização discreta formulados como problemas lineares em que um subconjunto de restrições expressas via disjunção.

$$\begin{aligned} & min \quad z = \sum_i \sum_k c_{ik} + f(x) \\ & st \qquad g(x) \le 0 \\ & \bigvee_{i \in D_k} \begin{bmatrix} Y_{ik} \\ h_{ik}(x, c_{ik}) \le 0 \end{bmatrix}, \quad k \in SD \\ & x \in R^n, \; c \in R^m, \; Y \in \{verdadeiro, falso\}^m \end{aligned} \tag{5.17}$$

Onde:

Y_{ik}: são variáveis boleanas representantes das variáveis binárias que estabelecem que um termo da disjunção é verdadeiro se $[h_{ik}(x,c_{ik}) \le 0]$ ou falso $[h_{ik}(x,c_{ik}) > 0]$;
i: índice que representa o número de termos nas restrições na forma de disjunção;
k: índice que representa uma disjunção;
SD: Subconjunto de disjunções em no mínimo uma deve ser verdadeira.

Diante de todas as abordagens feitas acima, nenhuma levaria a um avanço no tratamento de problemas de Programação Inteira-Mista se recentemente códigos matemáticos não oferecem flexibilidade de utilização. No capítulo 6, o sistema OSL, que permite que subrotinas sejam implementadas pelo usuário com intuito de interferir durante o processo "branch and bound", é abordado. Além do OSL o sistema GAMS surge como uma excelente interface na geração de modelos os quais podem ser solucionados por "resolvedores" adequados ao problema.

Todas as técnicas descritas acima visam o aceleramento de solução de modelos que se utilizam da Programação Matemática Inteira-Mista e, consequentemente, problemas com maior complexidade e dimensão podem ser solucionados. Na seção que segue são mostrados alguns resultados de problemas de programação de produção em sistemas flexíveis.

5.5. Programação de Produção em uma Planta Batelada utilizando Análise Lógica.

5.5.1. Estabelecimento do Problema

O problema aqui abordado (Raman e Grossmann,1993) visa o programação de três tarefas em 3 estágios, tendo como objetivo a minimização do tempo total de todas as tarefas (makespan). A estrutura de processamento consta de três estágios, onde cada estágio é composto por um único equipamento, segundo descrição abaixo. A armazenagem de materiais intermediários é do tipo ZW ("zero wait"), ou seja, assim que uma batelada de substância é produzida em algum equipamento, esta deve ser imediatamente processada no próximo equipamento. Os tempos de processamento nos respectivos equipamentos são apresentados na tabela 5.3.

Tabela 5.3: Tempos de Processamento

Tarefas	Reator 1	Reator 2	Separador
A	5	-	3
B	-	3	2
C	2	4	-

Os equipamentos estão distribuídos nos estágios como mostrado abaixo:

Estágio E_1: formado por um reator (R1)
Estágio E_2: formado por outro reator (R2)
Estágio E_3: formado por um separador (S)

5.5.2. Modelagem e Solução do Problema

- Equação de término de processamento de cada tarefa i

$$tf_i = t_i + \sum_{j \in J(i)} tp_{ij} \qquad (5.18)$$

Onde:

tf_i: tempo de término da tarefa i
t_i: tempo de início da tarefa i

tp_{ij}: tempo de processamento de i no estágio j
J(i): conjunto estágios habilitados a processar a tarefa i

- Equação de tempo de início de i em j

$$tS_{ij} = t_i + \sum_{\substack{m \in J(i) \\ m<j}} tp_{im} \tag{5.19}$$

- Equação de tempo final de processamento de i em j

$$tF_{ij} = t_i + \sum_{\substack{m \in J(i) \\ m \leq j}} tp_{im} \tag{5.20}$$

Como o objetivo é minimizar o tempo de execução de todas as tarefas (função objetivo), T, o problema é formulado com as seguintes expressões:

$$\min T \tag{5.21}$$

$$T \geq t_i + \sum_{j \in J(i)} tp_{ij} \qquad \forall i \in I \tag{5.22}$$

$$t_i + \sum_{\substack{m \in J(i) \\ m \leq j}} tp_{im} \leq t_k + \sum_{\substack{m \in J(k) \\ m<j}} tp_{km} + M(1 - y_{ik})$$

$$\forall j \in C_{ik}, \forall i,k, i < k, i,k \in I \tag{5.23}$$

onde:

C(i,k):conjunto de estágios comuns a processar i e k

$$t_k + \sum_{\substack{m \in J(i) \\ m \leq j}} tp_{km} \leq t_i + \sum_{\substack{m \in J(k) \\ m<j}} tp_{im} + My_{ik}$$

$$\forall j \in C_{ik}, \forall i,k, i < k, i,k \in I \tag{5.24}$$

$$y_{ik} + y_{ki} = 1 \qquad \forall i,k, i < k, i,k \in I \tag{5.25}$$

$$T \geq 0, t_i \geq 0, i \in I \qquad \forall i,k, i < k, i,k \in I \tag{5.26}$$

Onde

$$y_{ik} = \begin{cases} 1 & \text{a tarefa i precede a tarefa k} \\ 0 & \text{a tarefa k precede a tarefa i} \end{cases} \tag{5.27}$$

As equações 5.16 e 5.17 representam a relação de antecedência de duas tarefas, i e k, ou melhor, em um processador j conveniente ao processamento das duas, ou a tarefa i antecede k, ou k antecede i. Uma das duas situações deve acontecer.

Discretizando o modelo segundo os dados apresentados na tabela 5.3, o seguinte modelo é gerado.

$$\begin{aligned}
&\min T \\
&\text{st} \\
&T \geq t_A + 8 \\
&T \geq t_B + 5 \\
&T \geq t_C + 6 \\
&\\
&t_A - t_B \leq -5 + M(1-y_{AB}) \\
&t_B - t_A \leq My_{AB} \\
&t_A - t_C \leq -5 + M(1-y_{AC}) \\
&t_C - t_A \leq -2 + My_{AC} \\
&t_B - t_C \leq -1 + M(1-y_{BC}) \\
&t_C - t_B \leq -6 + My_{BC} \\
&\\
&T, t_A, t_B, t_C \geq 0 \\
&y_{AB+}, y_{AC}, y_{BC} = \{0,1\}
\end{aligned} \tag{5.28}$$

A presença do número M (valor grande) neste modelo caracteriza a presença de relações representadas por disjunções. As inequações que estabelecem as restrições de precedência podem ser representadas na forma de disjunções.

$$D_1: (t_A - t_C \leq -5) \vee (t_C - t_A \leq -2)$$
$$D_2: (t_B - t_C \leq -1) \vee (t_C - t_B \leq -6)$$
$$D_3: (t_A - t_B \leq -5) \vee (t_B - t_A \leq 0) \qquad (5.29)$$

Onde:

$D_1, D_2, D_3 \in SD$

$i \in D_k$, $k \in SD$

k: índice das expressões na forma disjuntiva (k=1,2,3)

i: índice de cada termo da disjunção D_k (i=1,2 para toda disjunção D_k)

SD: subconjunto de restrições na forma lógica

As disjunções D_k são então utilizadas durante a pesquisa "branch and bound" na tentativa de eliminar variáveis candidatas à ramificação.

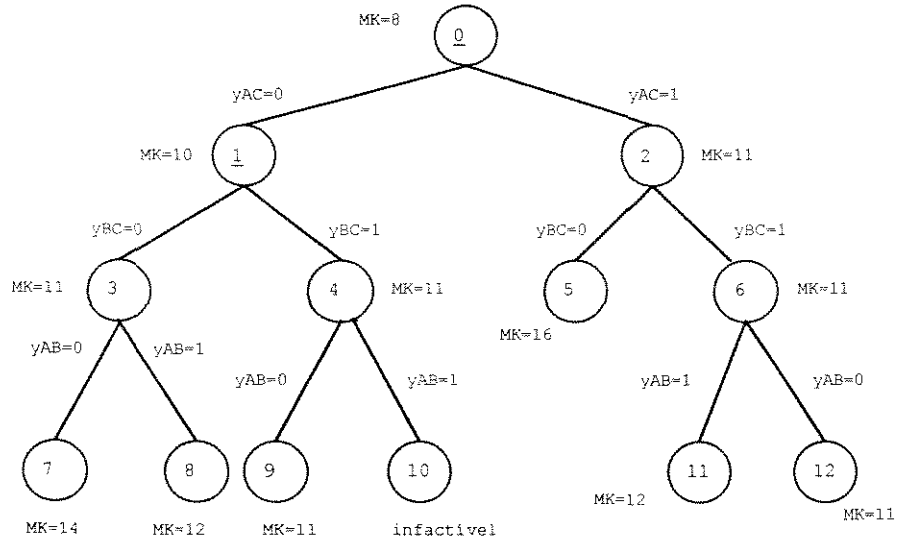


Figura 5.3: "Branch and Bound" padrão para o problema

A figura 5.3 mostra o desenvolvimento da árvore padrão para o problema apresentado. Nota-se que em cada nó são gerados dois nós, tal como apresentado na teoria de busca implícita.

Na solução do problema é considerada a seguinte terminologia:

N_N : {(i,k)/y_{ik} está fixada no nó N}

M_N : {(i,k)/y_{ik} está livre no nó N}

W_N : subconjunto de variáveis livres no nó, que satisfazem as disjunções.

$W_N = \{(i,k)/(i,k) \in M_N , \bigvee_{i \in D_k} Y_{ik} = \text{verdade}, k \in SD\}$

P_N: conjunto de variáveis candidatas à ramificação no nó

Y_{ik} variável boleana associada a cada termo (cláusula) da disjunção k.

a) Nó raiz (nó 0)

A solução relaxada neste nó tem como valor da função objetivo MS=8.0 ("makespan"), para $(t_A,t_B,t_C)= (0,0,0)$. Substituindo estes valores em 5.29 nenhuma das disjunções é satisfeita, consequentemente todas as variáveis são candidatas à ramificação.

$P_0 = M_0 = \{y_{AB}, y_{AC}, y_{BC}\}$ e $W_0 = N_0 = \varnothing$

A partir deste nó escolhe-se uma das variáveis a ramificar. Fixando $y_{AC}= 0$ gera-se o nó 1 e $y_{AC}=1$ gera-se o nó 2.

b) Nó 1

Valor da função objetivo no nó é igual a 10 para $(t_A,t_B,t_C)= (2,0,0)$. Substituindo estes valores nas sentenças representadas por disjunções, pode-se fazer a seguinte análise:

$Y_{12}=Y_{22}=$ falso

$Y_{13}=$falso

$Y_{23}=$ verdadeiro

Através desta análise, D_3 (expressão lógica 3 dada por 5.29) encontra-se satisfeita logicamente, logo a variável y_{AB} pode ser retirada do conjunto de variáveis candidatas à ramificação e com isso somente a variável y_{BC} pode ser ramificada.

$P_1 = \{y_{BC}\}$

c) Nó 2

O nó 2 é gerado fixando $y_{AC}=1$. Neste nó tem-se que T=11 e $(t_A,t_B,t_C)= (0,0,5)$, que satisfaz a todas as disjunções de 5.29. Com isso tem-se a primeira solução factível para o problema ("upper bound").

c) Nó 3

O nó é gerado a partir ramificação de y_{BC} ($y_{BC}=0$) do nó 1. A função objetivo neste nó é de 11 para $(t_A,t_B,t_C)= (2,6,0)$. Como a solução não é melhor que a encontrada anteriormente então este nó pode ser desprezado.

d) Nó 4

Este nó é criado a partir do nó 1 com a fixação de y_{BC} em 1. A solução do problema linear neste nó fornece um valor de 11 para $(t_A,t_B,t_C)= (3,0,1)$.

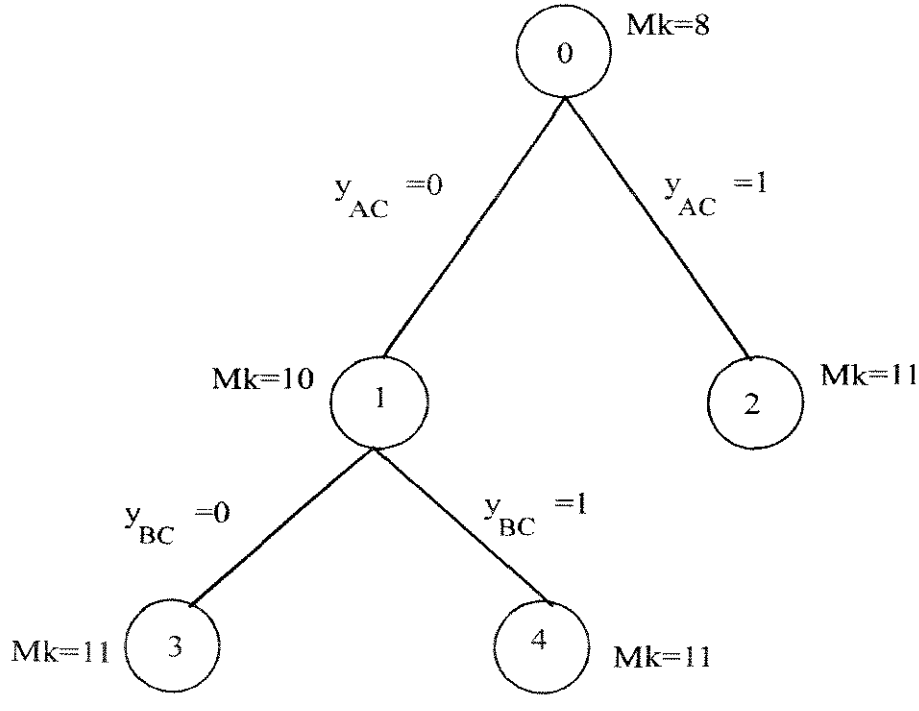


Figura 5.4: "Branch and Bound" utilizando análise lógica via disjunção

Então a solução ótima do problema apresenta um makespan de 11 unidades de tempo. A figura 5.4 mostra a árvore "branch and bound" gerada utilizando análise lógica.

Através da tabela 5.4, em todos os casos analisados, conseguiu-se uma aceleração de solução do problema quando se utilizou a formulação com análise lógica.

Tabela 5.4: Avaliação de Desempenho da Metodologia Proposta (Raman e Grossmann, 1994)

	Formulação Original	Formulação com Análise Lógica
5 tarefas, 10 estágios 165 restrições 10 var. binárias		
Nós	180	118
Iterações	378	277
Tempo CPU (s)	24.83	18.4
6 tarefas, 5 estágios 106 restrições 15 var. binárias		
Nós	499	366
Iterações	1058	813
Tempo CPU (s)	59.6	47.7
7 tarefas, 5 estágios 157 restrições 21 var. binárias		
Nós	2791	2000
Iterações	6186	4500
Tempo CPU (s)	339.7	256.4

5.6. Aplicação de Análise Lógica na Formulação com Tempo Contínuo

Como mostrada no capítulo 4, a formulação apresentada por Pinto e Grossmann (1994) utiliza a variável tempo formulado continuamente, em contraponto com a formulação proposta por Kondili et al. (1993) que, modela o tempo de forma discreta.

Uma das estratégias adotada por Pinto e Grossmann (1995), quando os problemas envolvem recursos compartilhados, é eliminar do modelo as restrições que determinam a limitação de recursos e interferir logicamente e externamente na busca em árvore para garantir tal restrição. Para tanto são propostas expressões lógicas sobre recursos compartilhados, as quais são utilizadas externamente durante a busca em árvore.

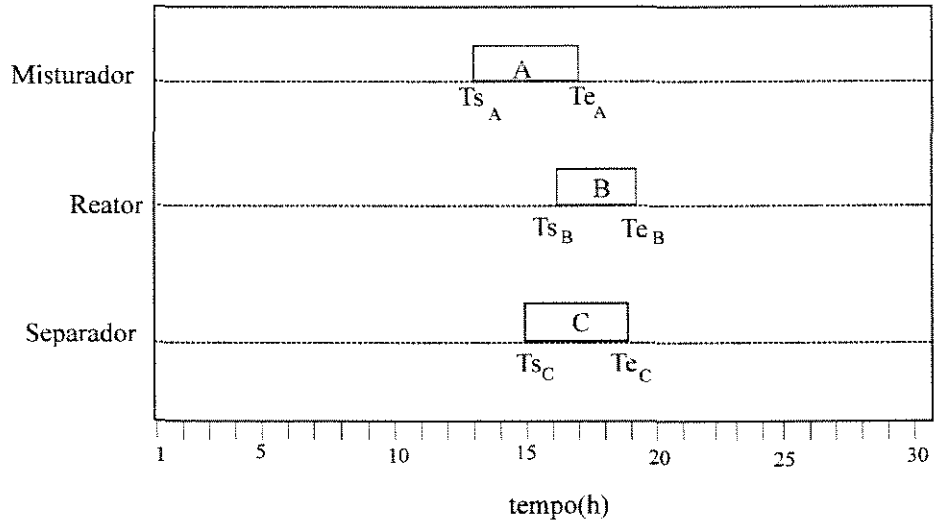


Figura 5.5: Programa de produção parcial equivalente ao Nó N

As variáveis T_S e T_e, mostradas na figura 5.5, são indicadores de tempo de início e de término das operações.

Considerando que o problema envolva limitação de recursos e que a demanda de consumo de recurso r das operações é dada na tabela 5.5, o programa parcial de produção dado na figura 5.5 é infactível, pois em determinados intervalos tem-se a violação da oferta de recurso. A limitação de recurso para o problema é 3 u.r.

Tabela 5.5: Demanda de recurso r das operações

Tarefas	Misturador	Reator	Separador
A	2	2	1
B	1	2	2
C	1	1	-

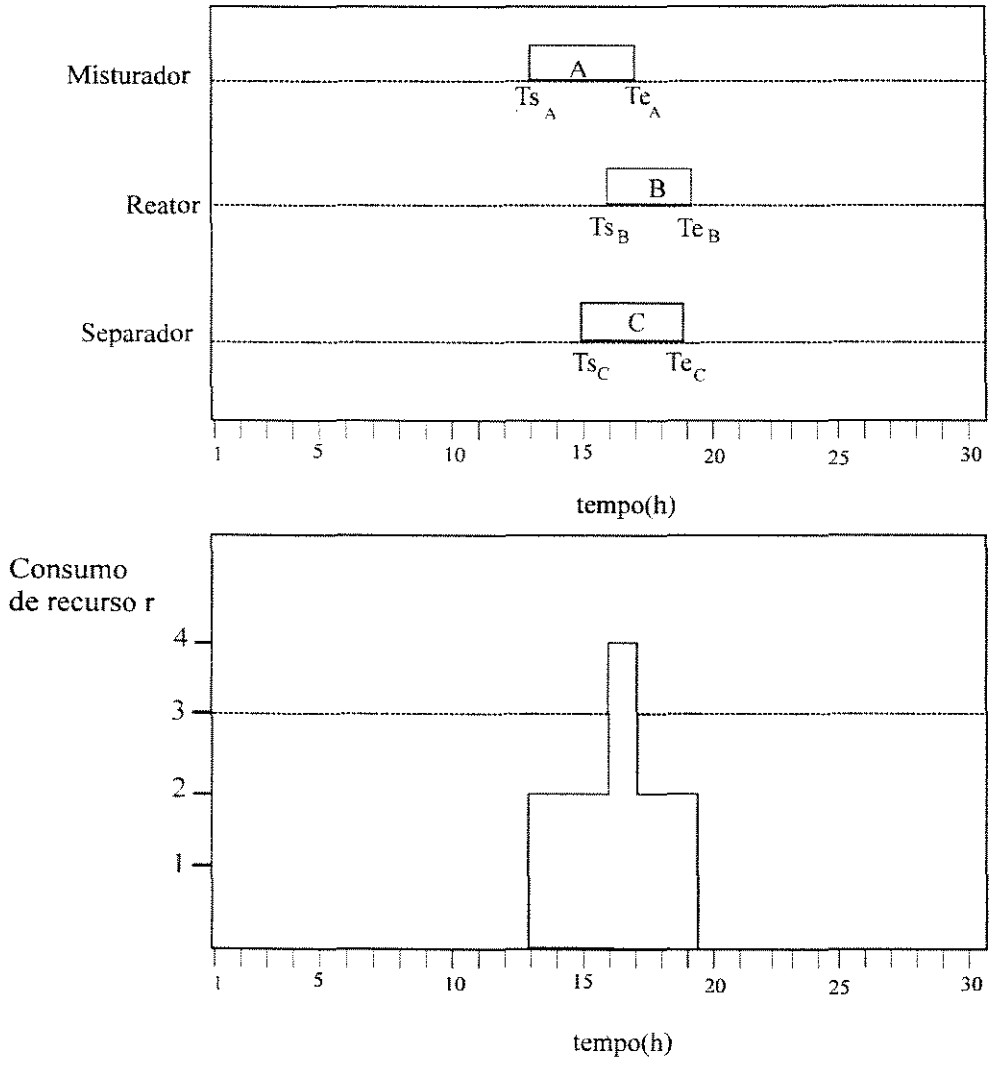


Figura 5.6: Programa de produção com consumo de recurso r

Se o modelo não apresentar as restrições que garantem a limitação de consumo de recursos compartilhados, o programa parcial de produção dado pela figura 5.6 pode perfeitamente acontecer durante a solução do problema. Este é o caso do trabalho proposto por Pinto e Grossmann (1995) que para diminuir a dificuldade de solução retiram as restrições de limitação de recurso do modelo, possibilitando soluções

parciais sem observância da oferta de recurso. Logo para invibializar o programa de produção os autores iterferem logicamente na busca em árvore adicionando restrições que garantem a não simultaneidade de operações que não podem ser desenvolvidas simultaneamente, por exemplo, das operações (A,misturador) e (B,reator). Para analisar o procedimento de inferência lógica utilizado pelos autores considere o programa dado pela figura 5.5 e que a operação (A,misturador) foi alocada antes que a operação (B,reator). Para invibializar um programa parcial a partir de um nó da árvore de busca, os seguintes procedimentos básicos são propostos.

- Selecionar dentre as operações cujo o início de processamento viola a limitação de recursos compartilhados a que tiver um maior tempo de início de processamento. Por exemplo a operação q_n, que no caso da figura 5.5 é a operação (B,reator).
- Determinar as operações que juntamente com a operação q_n contribuem para a violação de um determinado recurso em um determinado nó N, gerando assim o conjunto de pares operações (Io_N). Para o exemplo mostrado na figura 5.6, as operações (A,misturador) e (B,reator) sendo desenvolvidas simultaneamente violam a limitação de oferta de recurso

A partir do nó N, são adicionadas restrições que procuram resolver o problema de violação de recursos. seguindo as determinações dos itens acima. Para o exemplo considerado, como somente um par de duas operações viola a oferta de recurso, dois nós a partir do nó corrente (representante da situação mostrada na figura 5.6) são gerados. Um é gerado adicionando ao subproblema N a restrição $Te_A<Ts_B$ e o outro nó é gerado a partir da adição da restrição $Ts_A > Te_B$. A figura 5.7 mostra a partir do nó N a geração destes dois nós utilizando o conhecimento lógico sobre recursos compartilhados.

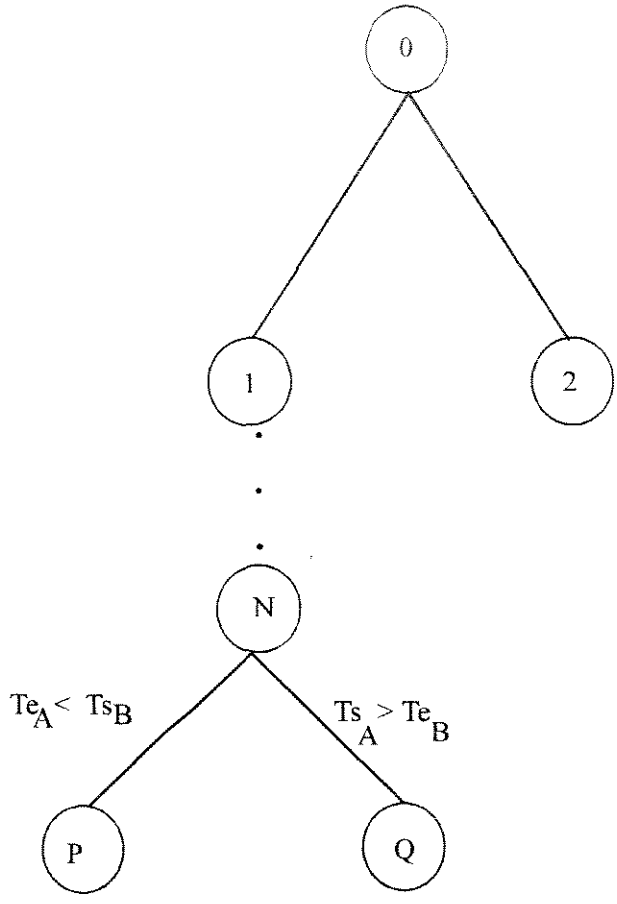


Figura 5.7: Processo de formação da árvore de busca segundo análise lógica proposta por Pinto e Grossmann (1995)

Nota-se portanto que no caso do nó possuir mais de um par de tarefas que violem a oferta de recurso, tentar-se-á resolver todas violações, por exemplo se (B,reator) e (C,separador) não pudessem ser também desenvolvidas simultaneamente, além dos nós P e Q gerados a partir de N, dois outros nós procurando tornar factível o programa de produção com relação às duas últimas operações citadas teriam que ser gerados, conclui-se portanto que a árvore não é necessariamente binária.

5.7. Conclusão

Neste capítulo procura mostrar algumas premissas da aplicação de análise lógica dentro do contexto de programação matemática. Esta aplicabilidade exige o conhecimento do problema que está sendo tratado, pois como mostrado é necessário decidir sobre quais restrições que podem ser satisfeitas de modo externo ao modelo. O modo de proceder o uso do conhecimento lógico envolvendo uma determinada característica do problema é fortemente dependente da forma que é modelado o

problema. Como mostrado no trabalho de Pinto e Grossmann (1995), ao se utilizar a não discretização do tempo e as variáveis binárias com características de variáveis de precedência, praticamente obriga a estratégia de adição de restrições segundo o conhecimento lógico sobre recursos compartilhados, como mostrado acima.

CAPÍTULO 6

O CONSTRUTIVISMO DO SISTEMA OSL E SUA APLICABILIDADE EM INTERFERÊNCIA LÓGICA EXTERNA

6.1. Introdução

A incorporação de análise lógica à metodologia de programação linear inteira-mista mostra-se bastante promissora para a diminuição da dificuldade de solução, sem contudo interferir na qualidade da solução. Hooker (1994) mostra algumas vantagens de se utilizar o procedimento de satisfação de proposições, o qual é denominado de programação linear lógica-mista ("MLLP"), que é uma extensão da programação linear inteira-mista, onde elementos discretos (em especial as binárias) são representados por proposições (expressões) lógicas. O autor também mostra a implementação de algoritmo em linguagem C que desenvolve regras de ramificação segundo análise de expressões na forma lógica.

No caso de se utilizar um pacote otimizador, tal como o LINDO, não se pode interferir em sua metodologia de busca, dentro do contexto de interferência mostrado no capítulo anterior, pois o arquétipo de construção deste pacote não permite o acesso aos procedimentos padrões por parte do usuário (como por exemplo na subrotina que desenvolve a busca "branch and bound"). No caso do sistema OSL, a sua filosofia de implementação e funcionalidade, que trazem extrema inovação frente a outros pacotes, são desenvolvidas praticamente em cima desta característica, que é permitir que o usuário, em determinados pontos do processo de busca, interfira durante o processo de busca, segundo o seu conhecimento sobre o comportamento do problema que está sendo tratado, obviamente que esta interferência é desenvolvida com algumas limitações. Portanto, com esta flexibilidade de utilização, o OSL se torna uma ferramenta que propicia análise de inferência lógica, tal como defendida pela metodologia de programação linear lógica-mista.

Este capítulo não tem o intuito descrever por completo o uso do sistema OSL para o tratamento de problemas via programação matemática, e sim mostra numa visão pragmática os pontos de interações possíveis entre o usuário e o sistema OSL, facilitando a compreensão do capítulo 7 direcionado ao desenvolvimento da abordagem deste trabalho.

Neste fato, sedimenta-se a proposta deste capítulo que visa mostrar como se desenvolve interferência externa pelo usuário no sistema OSL. Em suma, pretende-se mostrar como e onde se pode interferir nos procedimentos padrões do referido pacote, de modo que a partir de um conhecimento lógico previamente estabelecido se interfira

externamente na metodologia de busca implícita "branch and bound" do respectivo sistema.

Seguindo a abordagem deste trabalho, os exemplos aqui mostrados são direcionados aos problemas de programação de produção, o que não limita a descrição aqui desenvolvida em outros problemas que envolvam elementos discretos em sua formulação, tal como problemas de síntese de processos, como mostrado por Raman e Grossmann (1991).

6.2. Acessibilidade aos dados do Problema pelo Sistema OSL

O sistema OSL fornece duas formas para acessar os dados de um problema: uma é utilizando subrotinas que permitem a entrada de cada elemento do problema (número de linhas, número de variáveis, coeficientes das linhas, limitantes das linhas, etc) e a outra é através da leitura de arquivo no formato "MPS" ("Mathematical Programming System"). O formato "MPS" é uma forma de saída padronizada de alguns pacotes matemáticos (por exemplo GAMS), que facilita a solução de um determinado modelo por diversos pacotes otimizadores. Neste caso, o GAMS (Brooke et al., 1988) serve tão somente para discretizar as restrições do modelo, que um das etapas mais árduas quando se utiliza a programação matemática como metodologia de solução.

Na solução dos problemas abordados neste trabalho, primeiramente estes modelados compilados no GAMS gerando arquivos "MPS", como mostra a figura 6.1, o qual é utilizado pelo sistema OSL. No apêndice B é mostrado um exemplo de arquivo "MPS" para um determinado problema de programação de produção, nota-se que este formato "MPS" nada mais é que a matriz de elementos do problema numa forma sistematizada.

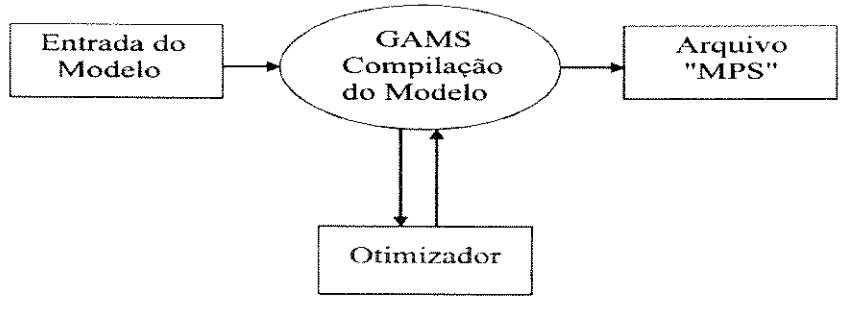


Figura 6.1: Geração do arquivo "MPS"

No arquivo "MPS", na seção colunas, as variáveis do problema são numeradas seqüencialmente. Para o arquivo apresentado no apêndice B, as colunas (ou variáveis) de 1 a 20 são representantes das variáveis binárias W_{ijk} do respectivo problema, ou seja, durante o desenvolvimento da árvore "branch and bound", o OSL designa as variáveis escolhidas para a ramificação por intermédio destes números, logo qualquer implementação que exija saber qual a variável W_{ijk} está sendo ramificada necessita-se de um procedimento que transforme as informações no formato "MPS" em linguagem do modelo (em variáveis W_{ijk}). Este procedimento de transformação é desenvolvido no capítulo 7. A estrutura abaixo mostra as seções de uma arquivo em formato "MPS".

- NAME: Nome do pacote gerador do arquivo "MPS"
- ROWS: Fornece nome de linha e coluna da matriz, nome da função objetivo
- COLUMNS: Fornece o nome de cada coluna da matriz (variáveis contínuas e binárias), o nome da linha em que um determinado coeficiente aparece e o seu respectivo valor .
- RHS: Fornece o nome e valor de cada limitante de cada restrição do modelo
- BOUNDS: Fornece o nome e limitantes de cada variável do modelo
- ENDATA: Indicador de fim de arquivo

A entrada de modelos no OSL é feita utilizando um arquivo no formato "MPS" gerado pelo sistema GAMS. A figura 6.2 esquematiza a seqüência de utilização dos sistemas GAMS OSL utilizados no presente trabalho.

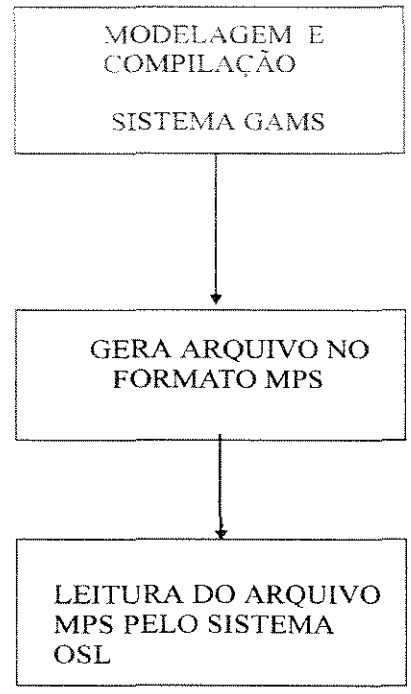


Figura 6.2: Seqüência de utilização dos Sistemas GAMS e OSL

6.3. Estrutura de Dados do Sistema OSL

O OSL mantém informações sobre cada conjunto de variáveis do modelo. No caso do modelo apresentar somente variáveis binárias, tem-se somente um conjunto de variáveis. Estes conjuntos possuem as seguintes informações:

a) Informações sobre as variáveis definidas

b) As variáveis são consideradas contínuas quando forem do tipo SOS 1 e 2 ("Special Ordered Sets")

- Variáveis são binárias quando forem do tipo SOS 3 ("Special Ordered Set" do tipo 3)
- Variáveis são consideradas inteiras para os demais casos

b) Informação de cada conjunto de variáveis: Se um modelo possui mais de um tipo de variável, estas são separadas em conjuntos distintos, segundo classificação acima.

- Prioridade do conjunto: Estabelece a ordem de ramificação dos conjuntos (Uma variável do conjunto de prioridade k_1 terá preferência de ramificação sobre uma variável do conjunto de prioridade k_2, onde $k_1 < k_2$.
- O número de variáveis pertencentes ao conjunto

- O tipo do conjunto (geral ou do tipo SOS:1, 2, 3)

c) As variáveis de cada conjunto são armazenadas com as seguintes informações:

- Número interno que identifica cada variável (O OSL atribui a cada variável inteira e que serve para identificar a variável durante o processo de ramificação).
- Valor da variável
- "Up pseudocost" (custo estipulado para ramificar uma variável binária em 1)
- "Down pseudocost" (custo estipulado para ramificar uma variável binária em 0)

6.4. Considerações sobre a Metodologia "Branch and Bound" desenvolvida pelo Sistema OSL

6.4.1. Introdução

Esta seção procura mostrar algumas diferenças entre metodologia "branch and bound" encontrada na literatura e a que é desenvolvida pelo sistema OSL, como por exemplo, uma diferença que é essencial para que se possa interferir corretamente sobre a sua busca "branch and bound" é a estratégia de escolha da variável a ser ramificada, a qual é imediatamente escolhida assim que um no nó é gerado.

As considerações do algoritmo "branch and bound" do OSL (desenvolvido pela subrotina EKKMSLV) apresenta várias características que necessitam de alguns esclarecimentos. No caso de problemas com variáveis 0 ou 1, a partir no nó 0 (zero), onde se tem a solução contínua (melhor solução não factível para o problema) o OSL segue os seguintes passos:

a) Passo 1: Escolha de um nó entre os nós ativos da lista. É selecionado o mais promissor segundo heurísticas incorporadas no sistema.

b) Passo 2: Soluciona o problema de programação linear (PPL), podendo fornecer uma solução factível, se não houver mais variáveis binárias com valores fracionários, ou uma solução infactível, se as variáveis inteiras persistirem com valores fracionários e/ou restrições não puderem ser satisfeitas.

A depender da solução encontrada, uma das decisões é tomada pelo OSL:

- Elimina o nó do processo de busca: Neste caso, ou a solução é infactível (não satisfaz às restrições) ou é pior que a melhor solução factível encontrada até o momento.

- Registra o valor corrente como uma nova solução: Neste caso a solução do nó corrente é factível e melhor que a anteriormente encontrada, então tem-se uma nova solução para o problema.
- Escolhe uma variável a ser ramificada no respectivo nó: Neste caso a variável escolhida é fixada em 0 (ramificação "down") ou em 1 (ramificação "up").

c) Passo 3: Retornar ao passo 1

O processo de busca termina quando não houver mais nós ativos a serem analisados ou quando for atingido o critério de otimalidade da solução.

6.4.2. Escolha do nó a ser ramificado

Durante o processo de busca do OSL, para cada nó gerado as seguintes informações são fornecidas:

a) Valor da função objetivo

b) Degradação estimada do nó

c) Número de variáveis definidas como inteiras que apresentam valores fracionários (infactibilidades inteiras)

d) Somatório das variáveis fracionárias

e) Nível de profundidade do nó na árvore

f) Número inteiro que indica o nó pai

g) O tipo de variável, de acordo com a classificação abaixo

1.tipo SOS 1

2.tipo SOS 2

3.tipo SOS 3

4.inteira (inclusive binária)

h) Variável ramificada: indica o número interno de referência da variável, se a variável for do tipo 4 (variável inteira). O número de referência é um valor inteiro atribuído a cada variável. Para os demais tipos de variáveis é fornecido o número do conjunto que está sendo analisado.

i) Valor da variável

j) Direção de ramificação de cada variável (se 1 ou 0 para variável binária)

6.4.3. Regra de Escolha do nó

A escolha do nó a ser ramificado no sistema OSL é feita segundo os critérios apresentados abaixo. Esta escolha é feita logo após que a variável a ser ramificada no nó anteriormente gerado for escolhida.

a) Se nenhuma solução for encontrada, então os últimos nós são preferidos (características semelhantes ao procedimento "detph first").

b) Nós cujos os valores da função objetivo são melhores que a variável de controle são escolhidos. Onde Rtarget é uma variável interna do OSL definida como real e que estabelece um valor para a função objetivo.

c) O nó com a melhor solução estimada é escolhido

6.4.4. Escolha da Variável para Ramificação a partir de um nó gerado

Na escolha da variável que é ramificada, primeiramente o OSL escolhe o conjunto com o grau de prioridade igual a 1. Entre as variáveis deste conjunto é escolhida aquela que apresentar melhor degradação (menor entre a "up degradation" e "down degradation") é escolhida.

Nota-se que a implementação de algoritmo "branch and bound" é bastante complexa, principalmente no que diz respeito ao gerenciamento de memória. Por exemplo um problema com 50 variáveis binárias, o número de nós que podem ser analisados é de 2^{50} (Cada nó pode gerar dois novos nós). Obviamente que as características inerentes à "branch and bound" (como por exemplo eliminação de nós por análise de limitante da função objetivo em cada nó) viabilizam a metodologia. Mesmo assim, esse valor serve para mostrar a complexidade de gerenciar e implementar um algoritmo "branch and bound".

6.4.5. "Status" de uma determinada Variável estabelecida pelo OSL

O sistema OSL matem para cada variável armazenada uma indicação se a mesma é:

- Variável Básica
- Variável Não-básica
- Variável Fixada (0 ou 1 no caso de uma binária)

Estas informações são extremamente importantes tanto para o desenvolvimento da árvore, pois quando se parte para a solução subproblema referente ao nó é necessário saber quais as variáveis que foram anteriormente fixadas, assim como para o próprio desenvolvimento do método "simplex", onde em cada iteração tem seu estado inciático estabelecido pela escolha de qual variável não-básica entrará na base e qual a variável básica deixará a respectiva base.

6.4.6. Geração de nós Infactíveis na árvore "branch and bound"

Uma das etapas mais importantes da metodologia "branch and bound" é a escolha da variável a ser ramificada a partir de um determinado nó. Esta pode determinar a formação de um nó infactível devido a não satisfação de alguma restrição. O sistema OSL, permite que uma vez gerado um nó, linhas (ou restrições) que não puderam ser satisfeitas sejam detectadas através do indicador "**". Como é vista no capítulo 7, quando problemas envolvendo limitação de recursos existe uma grande tendência de formação destes nós, o que leva a um aumento significativo no número de nós gerados e consequentemente no tempo de solução desses. Como exemplo de formação destes nós que degradam a árvore de busca, considere a figura 6.3.

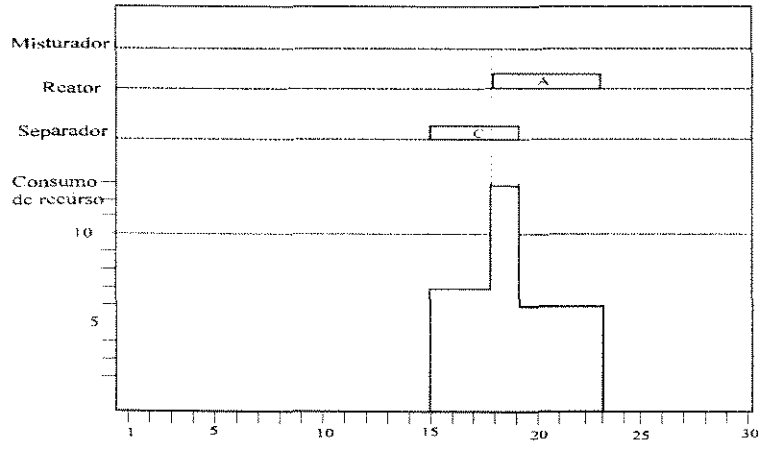


Figura 6.3: Distribuição temporal do consumo de recurso r

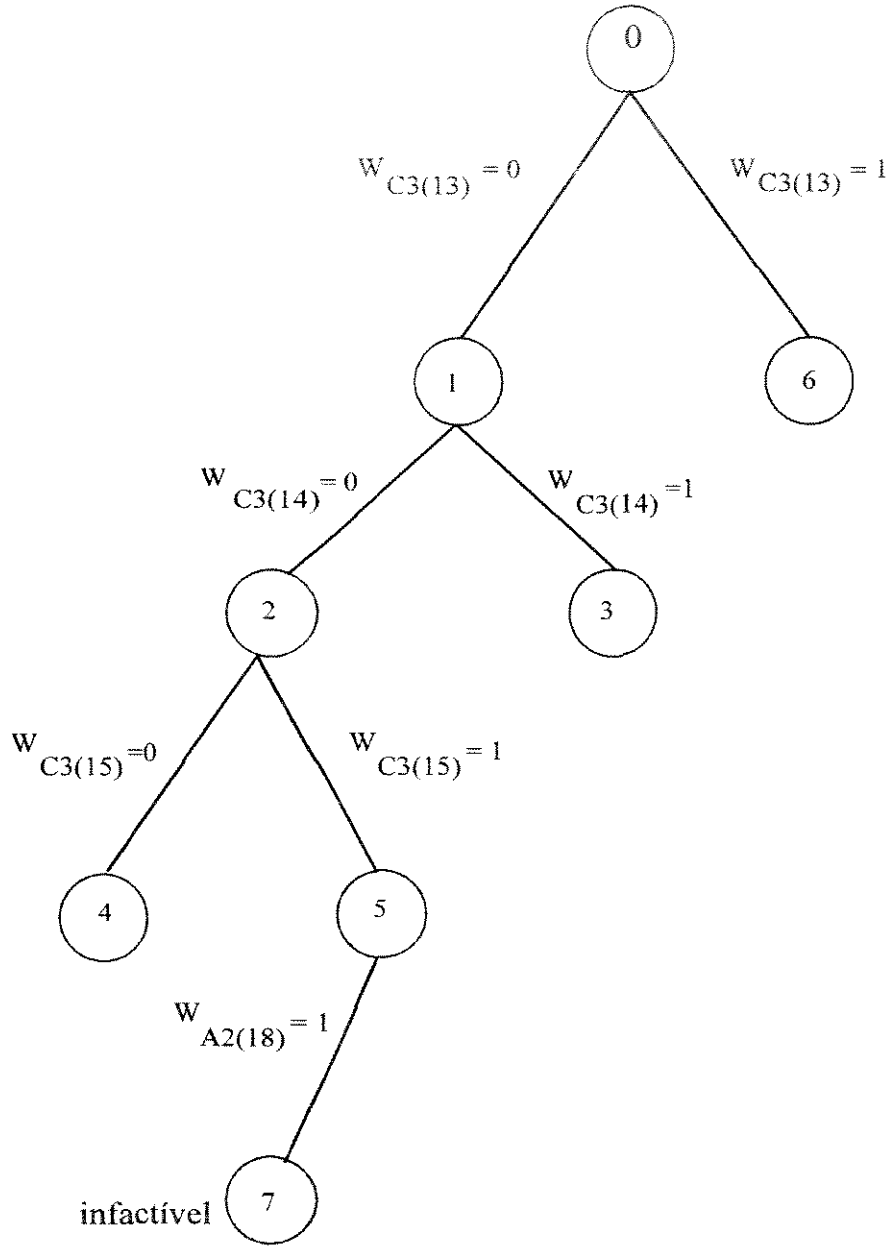


Figura 6.4: Representação via "branch and bound" da figura 6.3

Considerando que a operação (C,separador), a qual necessita de uma demanda de 7 u.r (unidades de recurso) foi alocada primeiro, nota-se que depois da alocação de A no reator (demanda de recurso de 6 u.r) não é possível garantir a limitação de oferta de recurso r (10 u.r) e para que esta violação ocorra basta que as variáveis que possibilitem a alocação de A no misturador ou no reator simultaneamente com a operação (C,separador) sejam diferentes de 0 e que o OSL escolha alguma destas variáveis para ramificar na direção 1. A representação desta situação pode ser visualizada via árvore "branch and bound", como é mostrada na figura 6.4. O nó 7 (gerado a partir do nó 5) é um inexorável devido a não satisfação da restrição que

limita o recurso compartilhado (restrição formulada pela expressão 3.8), ou seja, a situação de processamento neste nó é a mesma mostrada na figura 6.3, situação infactível.

6.4.7. Cálculo da Solução Estimada (SE)

Como visto na seção 6.4.2, uma das informações fornecidas em um nó é a degradação estimada. Esta informação é computada assim que o nó é gerado (nó ativo) e seu cálculo segue os seguintes passos:

a) Passo 1: Para cada variável (definida como inteira e/ou SOS) é calculado um custo de satisfação (CS_{var}), o qual é dado pela equação abaixo:

$$CS_{\text{var}} = Riweight + min\begin{bmatrix}(distancia\ para\ um\ valor\ inteiro\ acima\ do\ corrente).(up\ pseudocost)\\(distancia\ para\ um\ valor\ inteiro\ abaixo\ do\ corrente).(down\ pseudocost)\end{bmatrix}$$

Onde:

Rweight: Peso estipulado para cada infactibilidade inteira

O "up pseudocost" e "down pseudocost" podem ser visto como uma penalização ao se fixar uma variável em 0 ou 1. Quando a variável é fixada em seu limitante inferior ("lower bound") utiliza-se o "down pseudocost" e uma vez fixando a variável em seu limitante superior ("upper bound") utiliza-se "up pseudocost".

A degradação estimada de um conjunto de variáveis não-contínuas é calculada pela seguinte expressão:

$$DE_c = \sum_v CS_v$$

Onde:

v: índice de variáveis pertencentes ao conjunto de variáveis

c: índice de conjuntos

A degradação do nó (DE_{no}) é dada pela soma das degradações dos conjuntos.

$$DE_{no} = \sum_c DE_c$$

Este valor fornece um prognóstico de como a função objetivo degradar-se-á a partir do nó ramificado. Esta informação só é computada após esta escolha.

A partir da degradação do nó, o OSL calcula uma solução estimada da função objetivo

6.4.8. Estimativa da Função Objetivo

Durante o processo de busca, uma vez escolhido um nó para a ramificação, é calculada uma estimativa da função objetivo, a qual é computada pela seguinte equação:

$$SE = Rbestsol + R\deg scale.(DE_{no})$$

Onde:

Rdegscale (default=1): Fator de escala para o cálculo das degradações.
Rbestsol: Melhor solução encontrada até um determinado ponto da pesquisa em árvore.

6.5. Interferência Externa nos Procedimentos Padrões do Sistema OSL

A aplicação do OSL seguindo a abordagem desenvolvida neste trabalho, exige a análise de alguns procedimentos padrões do respectivo sistema. A utilização do arquétipo de implementação do OSL que permite a ação por parte do usuário em alguns pontos (em alguns de suas subrotinas) durante a solução de um determinado problema, passa por um estudo detalhado de sua implementação. Nesta seção, são fornecidas informações na conduta de solução de problemas via programação linear inteira-mista. Não se procura porém com esta descrição o entendimento por completo do OSL, e sim facilitar o entendimento do sistema desenvolvido neste trabalho.

O OSL é composto de quatro elementos básicos que são: as subrotinas de estruturas de dados, as ligadas aos algoritmos, as de controle de variáveis e mensagens. Composto de 60 subrotinas que podem ser chamadas a partir de um programa principal e/ou de suas próprias subrotinas. Estas subrotinas são classificadas dentro de quatro categorias.

a) Subrotinas relacionadas a dados: Estas incluem

- as subrotinas de entrada de dados (EKKMPS e EKKLMDL) e saída de resultados (EKKPRTS)
- as subrotinas de gerenciamento de dados internos (EKKDSCA, EKKDSCM) e de gerenciamento do vetor da matriz de elementos do problema (EKKSMAP).

b) Subrotinas relacionadas a algoritmos. Estas incluem:

- A subrotina que desenvolve a solução de problemas lineares (EKKSSLV),
- A subrotina que determina a melhor base inicial para o problema (EKKBASI)
- A subrotina que desenvolve preprocessamento (EKKPRSL)
- A subrotina que avalia a factibilidade de cada subproblema em cada nó da árvore (EKKSTAT) e a que desenvolve análise de sensibilidade de uma solução (EKKSOBJ).
- As subrotinas que permitem adição e eliminação de linhas e colunas durante a solução do problema (EKKCOL,EKKROW)

c) Subrotinas relacionadas ao Controle de variáveis. Nesta categoria estão incluídas as subrotinas que permitem o acesso às variáveis de controle definidas pelo sistema OSL, por exemplo o número de interações que devem ser desenvolvidas antes que o sistema interrompa o processo de busca (EKKIGET, por exemplo, permite que variáveis de controle definidas como inteiras sejam acessadas e EKKISET, por exemplo, permite que um novo valor seja atribuído a uma determinada variável).

- Subrotinas com funções diversas: Nesta categoria incluem-se as subrotinas que permitem fazer cópias dos dados do problema (EKKCOPY, EKKNWMT), as que permitem especificar nomes de linhas e colunas do problema (EKKNAME) e a que controla opções de mensagens (EKKMSET).

As principais subrotinas envolvidas diretamente na solução de problemas via programação inteira-mista são apresentadas abaixo. Além dessas, subrotinas de controle de variáveis (EKKNGET, EKKIGET, EKKRGET) e para estabelecimento de novos valores dessas variáveis (EKKRSET, EKKNSET) são utilizadas.

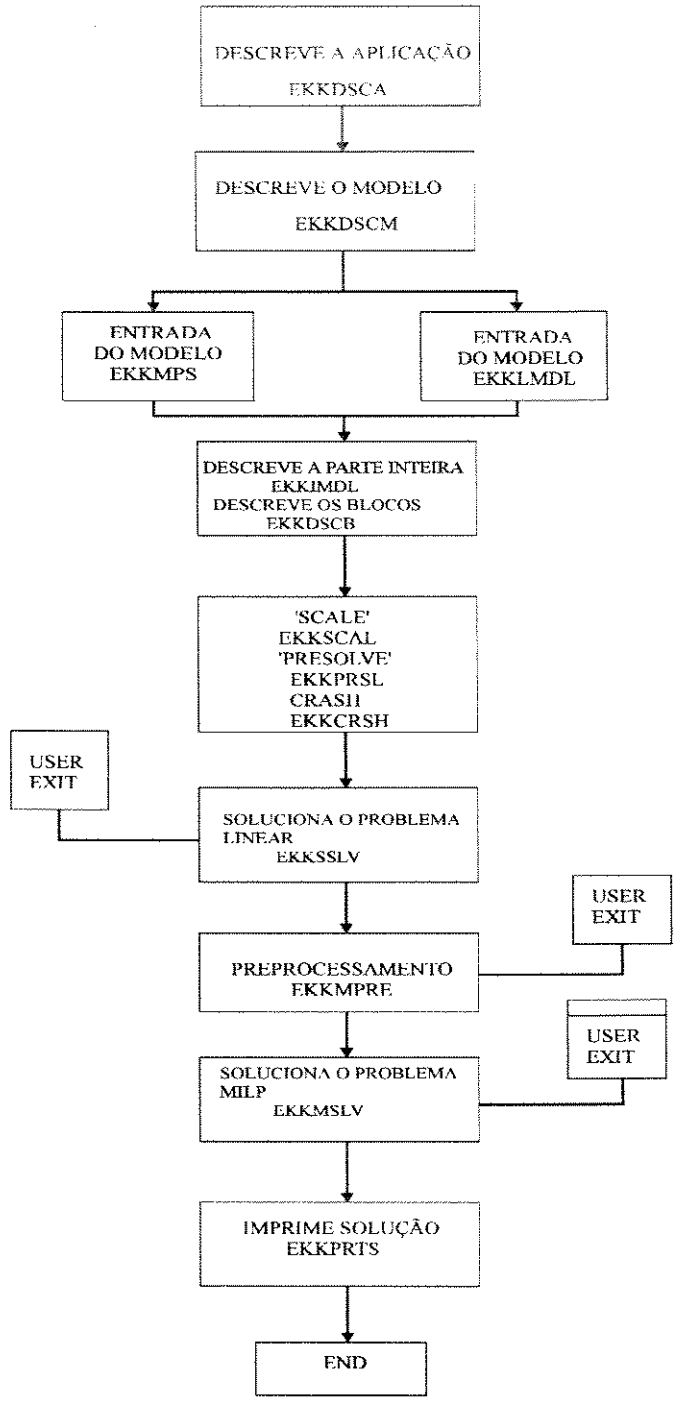


Figura 6.5: Subrotinas utilizadas na Programação Inteira-Mista via OSL

A descrição das principais subrotinas do OSL segue abaixo.

- EKKDSCA: Descreve a aplicação de Programação Matemática
- EKKDSCM: Descreve o modelo de Programação Matemática
- EKKMPS: Especifica a entrada de um modelo via formato "MPS"

- EKKLMDL: Especifica a parte linear do modelo
- EKKIMDL: Especifica a parte inteira do modelo
- EKKDSCB: Descreve um bloco da matriz do problema para o modelo
- EKKSCAL: Reduz os coeficientes da matriz de coeficientes
- EKKPRSL: Reduz o tamanho do problema
- EKKSSVL: Soluciona um problema de programação linear usando método simplex
- EKKMPRE: Preprocessa a árvore "branch and bound"
- EKKMSLV: Soluciona o problema via programação inteira-mista (desenvolve a árvore de busca "branch and bound").
- EKKPRTS: Imprime a solução do problema

A sequência de utilização destes procedimentos pode ser vista na figura 6.5. Observa-se por meio desta que em alguns pontos de execução de um programa, são chamadas algumas subrotinas denominadas "user exit". Essas subrotinas permitem que interferências externas possam ser feitas durante a solução do problema nos respectivos pontos indicados. Entre as subrotinas que permitem tal interferência estão:

- EKKBRNU: Escolhe a ramificação durante o processamento "branch and bound".
- EKKCHNU: Escolhe um nó durante o processamento "branch and bound".
- EKKCUTU: adiciona "cuts" durante o processamento "supernode".
- EKKEVNU: Avalia um nó durante o processamento "branch and bound".
- EKKHEUU: adiciona heurísticas durante o processamento "supernode" (EKKMPRE e EKKMSLV)
- EKKNODU: Salva informação de um determinado nó durante a busca "branch and bound".

O uso das subrotinas "user exit" pode aumentar o tempo de solução de um problema, exigindo com isso uma análise prévia de sua utilização. Contudo, elas fornecem um caminho de incorporação de conhecimento externo ao modelo (ausente na formulação do problema) durante o processo de solução, assim como possibilita

modificações em procedimentos padrões do Sistema OSL, como por exemplo, na regra de escolha de nós e da variável candidata à ramificação. Tais características estão ausentes em muitos pacotes destinados à solução de problemas via programação matemática. Entre as subrotinas "user exit" utilizadas no presente trabalho estão:

- EKKBRNU: Permite que regras de seleção da variável a ser ramificada sejam implementadas pelo usuário.
- EKKCHNU: Permite que a regra padrão de escolha do nó do sistema OSL seja modificada.
- EKKEVNU: Permite que sejam feitas diversas análises em cada nó da árvore "branch and bound". O acesso à informação da matriz de elementos de cada subproblema. É na respectiva subrotina que cada subproblema linear, correspondente a cada nó da árvore, é resolvido. O usuário portanto pode implementar nesta subrotina o procedimento de fixação de variáveis binárias, por exemplo.

A figura 6.6 mostra a seqüência de chamada destas subrotinas pelo OSL a partir de um determinado nó da árvore binária.

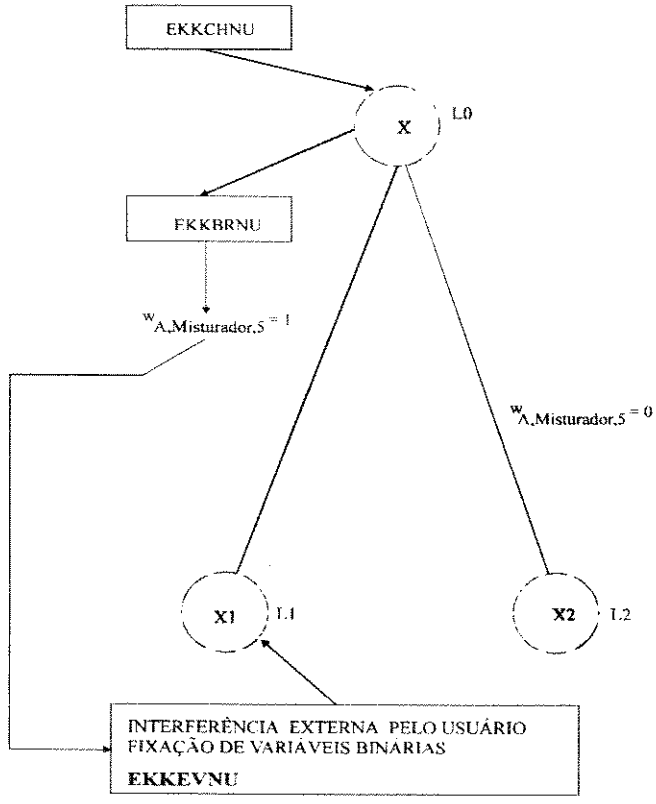


Figura 6.6: Sequência de chamada das "user exit" a partir de um nó X

6.6. Análise e Considerações sobre as subrotinas "User Exit"

As subrotinas "user exit" é chamada pela subrotina EKKMSLV (subrotina que desenvolve a busca "branch and bound") para avaliar o valor do limitante da função objetivo de um nó corrente. A subrotina EKKEVNU ("**EV**aluate a **N**ode **U**ser Exit") fornecida pelo OSL é mostrada abaixo, qualquer outra modificação na conduta padrão do OSL deve ser implementada pelo usuário, por exemplo, a fixação de variáveis segundo critério adotado pelo usuário pode ser feita nesta subrotina.

```
SUBROUTINE EKKEVNU (DSPACE,MSPACE,JRTCOD)
INCLUDE  (OSLI)
REAL*8 DSPACE (*)
INTEGER*4 MSPACE (*)
INTEGER*4 JRTCOD
C    Resolve o problema linear em cada nó da árvore
     CALL EKKSSLV (JRTCOD,DSPACE,2,1)
C    CALL EKKIGET (IRTCOD,DSPACE,OSLI,OSLILN)
C    Estabelece o retorno (JRTCOD=1) se a solução não for  encontrada
C    IPROBSTAT: variável tipo inteira que indica se a subrotina terminou
C    sua execução com sucesso
     IF (IPROBSTAT .NE.0) JRTCOD =1
     RETURN
     END
```

A subrotina EKKBRNU ("**BR**a**N**ch **U**ser Exit") é chamada pela subrotina EKKMSLV e tem como objetivo escolher a variável que vai ser ramificada a partir de um nó, qualquer modificação nos critérios adotados pelo do OSL deve ser implementada nesta subrotina. A estrutura de programa mostrada abaixo (pseudo-programa) exemplifica como o OSL, mais precisamente mostra a interação entre a subrotina EKKMSLV (subrotina que desenvolve a árvore "branch and bound") e a subrotina EKKBRNU.

```
C  Pseudo-Programa utilizado pelo sistema OSL no a EKKBRNU
C  EKKBRNU é chamada durante a escolha da variável a ser ramificada
C  Nenhum conjunto foi processado
        Call EKKBRNU (DSPACE,MSPACE,1,IARRAY,RARRAY,USERRTCD)
C  Para todos os conjuntos
        Do 200 I=1, "número de conjuntos"
        Call EKKBRNU (DSPACE,MSPACE,2,IARRAY,RARRAY,USERRTCD)
C
C  Para cada variável do conjunto
        Do 100 J=1, "número de variáveis do conjunto I"
        Call EKKBRNU(DSPACE,MSPACE,3,IARRAY,RARRAY,USERRTCD)
C  Aqui pode-se ser feita a consideração de ramificação
        100  CONTINUE
        Call EKKBRNU(DSPACE,MSPACE,4,IARRAY,RARRAY,USERRTCD)
C  Aqui pode-se fazer a mudança na escolha da variável a ser ramificada
        200  CONTINUE
         Call EKKBRNU(DSPACE,MSPACE,5,IARRAY,RARRAY,USERRTCD)
C  Aqui pode-se fazer a mudança na escolha do conjunto a ser  analisado
```

A subrotina EKKCHNU (**CH**oose **N**ode **U**ser exit) também de modo interno pela subrotina EKKMSLV e tem como objetivo modificar a escolha do nó feita pelo OSL.

```
C    Pseudo-Programa do sistema OSL para a subrotina EKKCHNU
C    EKKCHNU é chamada enquanto o OSL encontra-se no processo de
C.   Escolha dum nó ativo a ser ramificado.
C    Ponto em que nenhum nó foi processado
        CALL EKKCHNU(DSPACE,MSPACE,1,IARRAY,RARRAY,USERRTCD)
C  Para cada nó ativo
        Do 100 I=1, "número de nós ativos"
        CALL EKKCHNU(DSPACE,MSPACE,2,IARRAY,RARRAY,USERRTDC)
C    Neste ponto pode ser feita a regra de escolha do nó
  100  CONTINUE
        CALL EKKCHNU(DSPACE,MSPACE,3,IARRAY,RARRAY,USERRTCD)
C    Neste deve ser  estabelecido o nó escolhido.
```

6.7. Verificação da Árvore "Branch and Bound" desenvolvida pelo OSL

O modelo apresentado na seção 5.5.1 é solucionado pelo OSL e nesta seção é mostrado todas as etapas do desenvolvimento da árvore "branch and bound" utilizando a metodologia de satisfação de proposições lógicas associadas ao uso das "user exit". O desenvolvimento de uma árvore "branch and bound" em sua forma tradicional pode ser visto na figura 6.7.

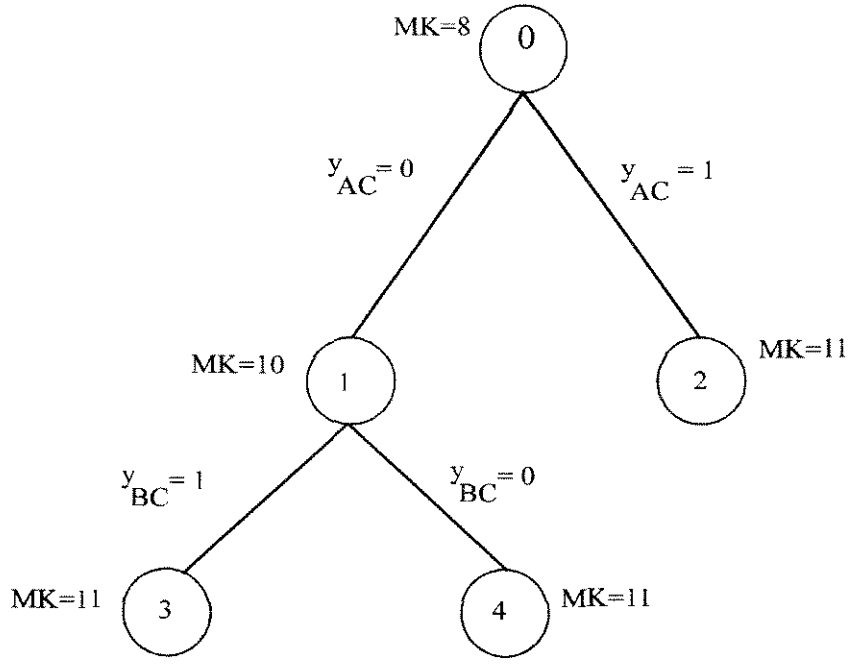


Figura 6.7: "Branch and Bound" utilizando análise lógica

Detalhando a árvore mostrada na figura 6.7, todas a etapas, inclusive mostrando os momentos em que as subrotinas que permitem interferência externa são mostrada na figura 6.8.

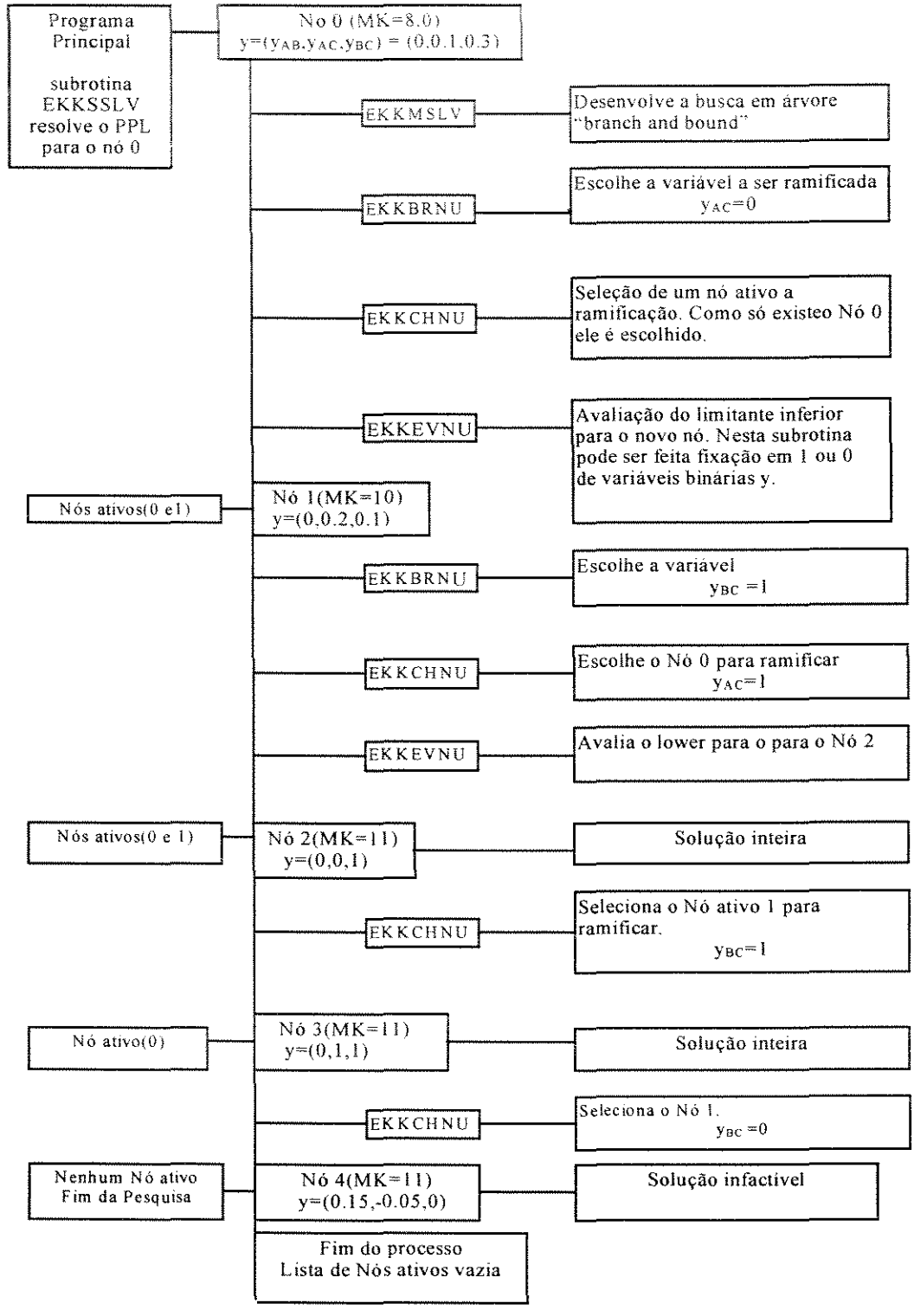


Figura 6.8: "Branch And Bound" desenvolvida pelo OSL

6.8. Considerações sobre o Nível de Interferência possível no Sistema OSL

Como dito anteriormente, o OSL fornece algumas vantagens na solução de problemas via programação matemática que é permissão de acessar alguns de seus procedimentos padrões durante a solução do problema. No entanto, este acesso encontra-se limitado não só nos pontos em que esta interferência é possível, como mostra figura 6.7, mas também no nível de interferência que pode ser realizado durante a solução do problema. Em Pinto e Grossmann (1994) é desenvolvido uma interferência onde restrições são insertas no modelo durante a solução dos problemas abordados pelos respectivos autores, determinando assim em um aumento no número de linhas na matriz de elementos do problema. Para isto o algoritmo a ser utilizado deve suportar tal nível interferência. A implementação do OSL, não permite por completo a sua utilização com este direcionamento (de eliminação e adição de restrições durante a busca). Caso o usuário deseje inserir e/ou eliminar restrições durante o processo de busca, estes procedimentos encontram-se limitados à ultima linha da matriz de elementos.

No caso de fixação de limitantes de variáveis e de linhas nos nós árvore, o OSL mostra-se totalmente flexível. É explorando esta característica que é desenvolvido este trabalho. A partir de relações lógicas sobre os problemas analisados é tomada uma ação sobre os limitantes das variáveis. Além dessas interferências, também qualquer regra de escolha da variável a ser ramificada e do próximo nó a ser analisado podem ser implementadas pelo usuário modificando assim a ação padrão do sistema OSL.

6.9. Conclusão

Neste capítulo, estabeleceu-se uma análise das possibilidades de interferências por parte do usuário sobre o sistema OSL, sem no entanto ter a pretensão de esgotar o assunto. O estudo aqui realizado apresenta as partes do sistema OSL que vão ser utilizadas, em especial as "user exit", assim como a sua funcionalidade, pois sem este conhecimento pormenorizado praticamente é impossível qualquer tentativa de aplicação das potencialidades do OSL.

No capítulo posterior é mostrado o desenvolvimento de relações lógicas para problemas de programação de produção que concatenado a este estudo, possibilitam o uso satisfatório das "user exit" no tratamento dos problemas analisados.

CAPÍTULO 7

APLICAÇÃO DE INTERFERÊNCIA LÓGICA EXTERNA EM PROBLEMAS DE PROGRAMAÇÃO DE PRODUÇÃO

7.1. Introdução

Neste capítulo, são investigadas alternativas para a diminuição do custo de solução de problemas de programação de produção. Primeiramente, a alternativa de uso do conhecimento lógico sobre o problema em questão durante a solução é desenvolvida, para tanto é necessário que a estrutura de implementação do pacote otimizador permita tais interferências, como ressaltado anteriormente. Neste ponto, justifica-se o uso do sistema OSL, cujo o arquétipo de implementação permite ao usuário interferir em determinados pontos dos seus procedimentos padrões. Em segundo lugar, a existência de relações lógicas entre variáveis binárias, as quais podem ser transpostas para a sua forma matemática, podem contribuir para diminuição da diferença entre a solução relaxada e a solução ótima do problema ("gap"). Consequentemente estas expressões insertas no modelo ocasionam um aumento no número de restrições, podendo com isso aumentar o custo de solução do mesmo.

As interferências desenvolvidas neste trabalho são direcionadas à fixação de variáveis binárias segundo uma determinada característica do problema analisado. Entre estas características encontram-se a análise sobre a disponibilidade de recursos compartilhados e tempos de estabelecimento entre operações ("setup times"). Além disso, um estudo detalhado no processo de solução via OSL do exemplo de programação de produção de uma planta multipropósito proposto por Kondili et al. (1993) é apresentado. Neste exemplo se verifica uma degenerescência na árvore "branch and bound" do pacote OSL e do GAMS/OSL.

Como estabelecido no capítulo 5, as metodologias utilizadas para a solução de problemas de otimização com satisfação de restrições sempre estão observando dois parâmetros, que são: a qualidade da solução e o tempo para se alcançar a solução de um determinado problema. Em cima destes parâmetros são propostas técnicas que venham diminuir a complexidade no tratamento destes problemas, sem contudo interferir na solução ótima dos mesmos. Isto é o que se pode chamar, no cenário de programação matemática, de "idealismo de solução", pois para problemas de grande dimensão a obtenção de um tempo de solução "satisfatório" sem comprometer a solução do mesmo ainda pode ser considerada uma situação não realista, pois à medida que os problemas aumentam de dimensão em termos de variáveis binárias e número de não-zeros o tempo de solução tende a aumentar exponencialmente com estas dimensões. No entanto, o que se pode definir como tempo satisfatório quando se

desenvolve uma mentalidade pragmática sobre o problema? Esta uma resposta que não pode ser fornecida se não quando a análise destes problemas ganham um contexto de aplicabilidade em seus respectivos campos.

A análise e aplicação de proposições lógicas no cenário da programação matemática vêm fornecendo resultados satisfatórios na redução do número de nós gerados e consequentemente no tempo de solução dos problemas, sem no entanto ganhar um "rótulo" de ser uma metodologia que venha solucionar problemas de larga escala.

A aplicação de interferência externa a partir de um conhecimento lógico sobre o problema não altera a solução ótima do mesmo, pois o que se busca é a fixação de variáveis binárias durante a solução, variáveis estas que podem assumir valores entre 0 e 1, mas que por uma análise lógica podem ser fixada em 0. Baseado nesta possibilidade de interferência, este capítulo mostra a implementação de um sistema que desenvolve interferência lógica em problemas de programação, concatenado com resultados promissores para o tratamento de problemas de larga escala.

Juntamente com a apresentação dos problemas a serem solucionados, as expressões que constituem os modelos dos mesmos, dentre aquelas apresentadas no capítulo 3, são referenciadas.

7.2 Análise e Desenvolvimento de Interferência Lógica sobre Recursos Compartilhados

7.2.1. Introdução

Entre as restrições que estabelecem um determinado modelo de um problema de programação de produção, encontram-se aquelas envolvendo a utilização de recursos compartilhados e a que limita o uso destes recursos durante o desenvolvimento das operações num horizonte de tempo. Isto significa que existe uma limitação na execução simultânea de operações. Com o estabelecimento de expressões lógicas envolvendo recursos compartilhados pretende-se impedir que alocações infactíveis sejam feitas durante o desenvolvimento da árvore "branch and bound". Por exemplo, se durante o processo de busca de algum modo o pacote otimizador tiver conhecimento que certas combinações de operações não podem coexistir durante a produção. Portanto, alocações infactíveis (devido à fixação de uma determinada variável W_{ijk} em 1) podem ser evitadas durante o processo de busca. Na exemplificação desta situação,

considere o seguinte problema de sequenciamento em uma planta de multiproduto, onde a oferta de recursos é limitada em 10 u.r (unidades de recurso) e o horizonte de produção H é de 24 h. Os dados do problema são apresentados nas tabela 7.1 e tabela 7.2.

Tabela 7.1: Tempos de Processamento

Tarefas	Misturador	Reator	Separador
A	6	5	2
B	5	3	7
C	3	5	4

Tabela 7.2: Demanda de recurso r das operações

Tarefas	Misturador	Reator	Separador
A	8	6	5
B	3	4	3
C	4	5	7

Tabela 7.3: Combinações envolvendo duas Operações

Conjunto de Operações	Consumo de recurso	Consumo Total de recurso
(A,misturador) (B,reator)	8 4	12
(A,misturador) (C,separador)	8 7	15
(A,misturador) (B,separador)	8 3	11
(A,reator) (C,separador)	6 7	13

Como a oferta de recursos está limitada em 10 u.r., é possível concluir que algumas combinações de operações não podem ser desenvolvidas simultaneamente durante a produção. Alguns pares de operações que inviabilizam a restrição de recurso são mostradas na tabela 7.3.

Analisando os resultados do consumo total das combinações acima, nota-se que tais duplas de operações não podem estar presentes simultaneamente na planta, pois não há recurso suficiente para satisfazê-las nos mesmos períodos de tempo (o limite é de 10 u.r.). Da mesma forma existem combinações de três operações que violam a oferta de recurso. Na tabela 7.4 são mostradas algumas destas combinações juntamente com consumo total de recurso r.

Tabela 7.4: Combinações de Operações que violam a limitação de recurso r

Combinações de Operações.	Consumo de recurso.	Consumo Total de recurso.
(A,misturador)	8	
(C,reator)	5	16
(B,separador)	3	
(A,reator)	6	
(B,misturador)	3	16
(C,separador)	7	
(A,separador)	5	
(B,reator)	4	13
(C,misturador)	4	

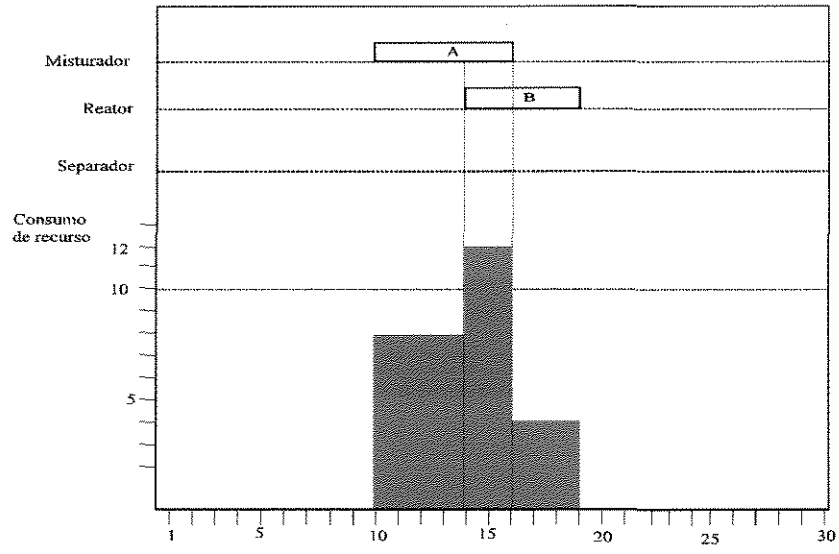


Figura 7.1: Distribuição temporal com violação da oferta de recurso

Através da figura 7.1 é possível visualizar que as operações (A,misturador) e (B,reator) não podem ser desenvolvidas simultaneamente. No período de tempo de k=14 a k=15 a restrição que limita o consumo total de recurso não é satisfeita e

portanto tais situações podem ser impossibilitadas de acontecer, pois leva à geração de nós infactíveis.

Diante destas impossibilidades, o que se pretende é uma vez alocada a ocupação (A,misturador), a operação (B,reator) seja impedida de ser desenvolvida simultaneamente com (A,misturador). Para isto todas a variáveis binárias W_{ijk} que possibilitem o desenvolvimento de (B,reator) e (B,separador) em períodos de tempo iguais ao processamento da operação (A,misturador) devem ser fixadas em 0, determinando assim que seja feita uma interferência no processo de busca "branch and bound", para determinar assim quais as operações já foram alocadas e proceder a fixação de variáveis binárias em 0, de acordo com a disponibilidade de recurso.

No exemplo dado, as variáveis de alocação W_{ijk} que representam a ocupação da operação (B,reator) nos períodos k=14 e k=15 poderiam ser eliminadas do conjunto de variáveis candidatas à ramificação, determinando assim um conjunto de variáveis não candidatas.

7.2.2. Análise de Conjuntos de Variáveis Binárias de Alocação Não-Candidatas à Ramificação

No processo de busca "branch and bound" dentro do contexto de programação matemática (a qual é desenvolvida de forma binária), são candidatas à ramificação todas as variáveis que apresentarem valores não inteiros, como abordado no capítulo 4. Por ineficiência dos pacotes otimizadores, algumas variáveis binárias podem apresentar valores diferentes de 0 (como visto em 7.2.1, as variáveis binárias W_{ijk} que representam o desenvolvimento das operações (B,reator) e (B,separador) em períodos de tempo iguais ao processamento da operação (A,misturador) devem estar fixadas em 0). Caso alguma destas variáveis binárias apresentem valor fracionário e seja ramificada, origina um nó em que a restrição de limitação de recurso não será satisfeita (gerando consequentemente um nó inexorável), como mostra a figura 7.1. Com a abordagem de interferência lógica se pretende eliminar do conjunto de variáveis candidatas à ramificação aquelas que por indisponibilidade de recurso compartilhado não podem ser mais fixadas em 1.

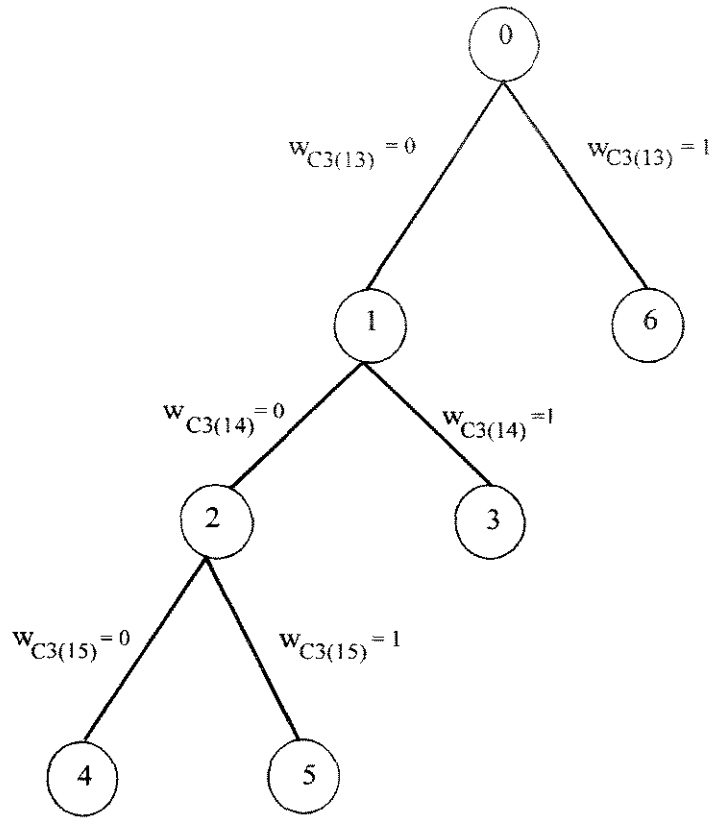


Figura 7.2: Árvore de busca parcial do problema 1

Procurando fornecer um melhor entendimento da abordagem apresentada neste trabalho, considere o problema de sequenciamento de tarefas (modelado pelas expressões) analisado na seção 7.2.1, o qual é resolvido pelo sistema OSL, e aqui para efeito de compreensão é mostrada parte da árvore "branch and bound" gerada, a qual é dada pela figura 7.2. Considerando, por exemplo, o nó 5, nota-se que este foi gerado pela fixação da variável $W_{(C)(3)(15)}$ em 1, ou seja, a operação (C,separador) foi alocada no separador no período 15 (o separador é representado pelo número 3 para efeito de simplificação, como encrespado anteriormente). A partir desta alocação, identifica-se operações que não podem ser processadas simultaneamente com a operação (C,separador) e, portanto variáveis binárias que violam a oferta de recursos devem ser eliminadas do processo de busca. A alocação desta operação leva o processamento à seguinte situação mostrada pela figura 7.3.

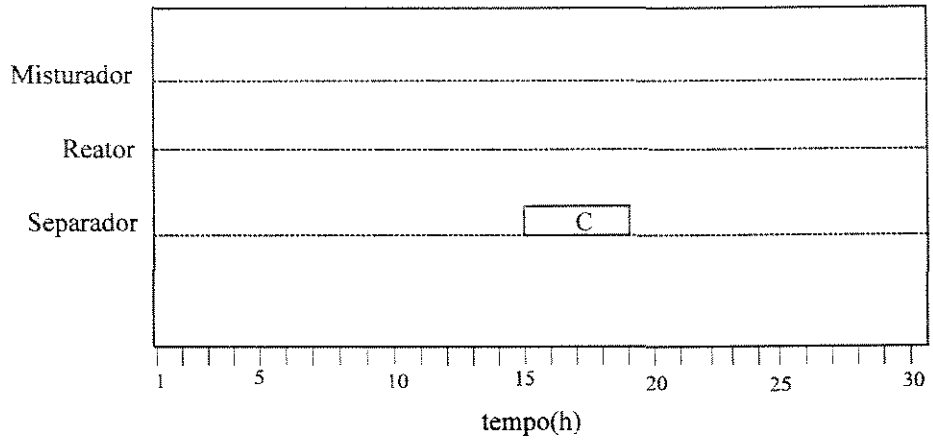


Figura 7.3: Situação da planta com a alocação da operação (C,Separador) no período 15

Na identificação de variáveis binárias que se fixadas em 1 podem gerar nós infactíveis por não satisfação da restrição de limitação de recurso, dois conjuntos de período de tempo(conjunto 1 e 2) que podem levar a este fato são constituídos.

a) Conjunto 1 de variáveis não-candidatas à ramificação

Analisando a figura 7.3, nos períodos de k=15 a k=18 (períodos 15,16,17,18), por exemplo, a operação (A,reator) não pode ser alocada, pois esta necessita de 8 u.r e a operação (C,separador) encontra-se consumindo 5 u.r. Logo é perfeitamente possível eliminar do processo de busca as variáveis binárias W_{ijk} que representem a ocupação da operação (A,reator) nos períodos de 15 a 18. A tabela 7.5 mostra os valores correspondentes a estas variáveis no nó 5, nota-se que tais variáveis por apresentarem valores fracionários são candidatas à ramificação (podendo ser alocadas, ou seja, fixadas em 1), isto porque não há como o resolvedor OSL detectar que tal escolha viola a oferta de recurso compartilhado.

Tabela 7.5: Valores das variáveis binárias da Operação (A,reator) no nó 5

Períodos	Variáveis binárias de alocação	Valores das variáveis no nó 5
15	$W_{A(2)(15)}$	0.0
16	$W_{A(2)(16)}$	0.052
17	$W_{A(2)(17)}$	0.035
18	$W_{A(2(18)}$	0.012

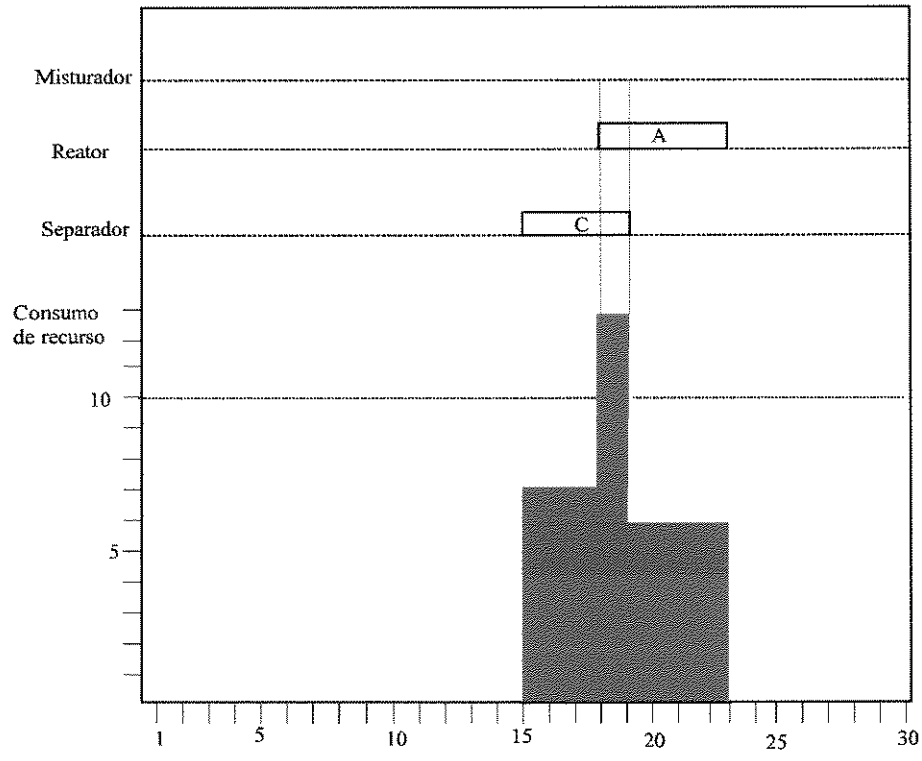


Figura 7.4: Programa parcial de produção infactível

As variáveis $W_{A(2)(16)}$, $W_{A(2)(17)}$ e $W_{A(2)(18)}$ apresentam valores fracionários e, portanto são candidatas à geração de um próximo nó, mesmo sabendo que, por análise de oferta de recurso, estas variáveis binárias deveriam apresentar valores iguais a 0 (não podendo ser alocadas ou seja, fixadas em 1). Digamos, por exemplo que a variável $W_{A(2)(18)}$ seja escolhida pelo OSL para gerar o nó 6 a partir do nó 5, isto leva o sistema de produção à situação mostrada na figura 7.4.

No período de tempo k=18 o consumo de recurso extrapola a oferta estipulada, consequentemente a operação (A,reator) não deve ser alocada nos períodos 15,16,17 e 18, significando com isso que as variáveis binárias de alocação dadas na tabela 7.5 devem ser fixadas em 0, evitando assim que o OSL as observe como variáveis candidatas à ramificação.

A mesma análise pode ser feita com outras operações que não podem ser simultaneamente desenvolvidas com (C,separador). Os valores das variáveis binárias W_{ijk} correspondentes a estas alocações no nó 5, são mostrados na tabela 7.6.

Tabela 7.6: Valores das variáveis binárias de alocação da Operação (B,reator)

Períodos	Variáveis binárias de alocação	Valores das variáveis no nó 5
15	$W_{B(2)(15)}$	0.0
16	$W_{B(2)(16)}$	0.09
17	$W_{B(2)(17)}$	0.07
18	$W_{B(2(18)}$	0.0

A determinação de períodos infactíveis a partir de uma ramificação "up" (fixação de uma variável binária em 1) pode-se calcular, tomando-se como base o período de tempo onde a operação foi alocada. Os períodos onde operações que necessitam de uma quantidade de recurso além da quantidade disponível não podem iniciar seu processamento. Considerando a alocação dada pela figura 7.4, onde a operação (C,separador) foi alocada no período k=15, a determinação de períodos infactíveis é mostrada a seguir.

$$S1_{i'j'} = \{k, k+1, \ldots, k+TP_{ij}-1\} \qquad \forall\, i' \neq i,\ j' \neq j \tag{7.1}$$

Onde:

$S1_{i'j'}$: Conjunto de períodos de alocação infactíveis para alocação das operação (i',j')
i: tarefa alocada
j: processador onde i foi alocada
k: período de alocação
i': tarefa não alocada
j': processador conveniente ao processamento da tarefa i'
TP_{ij}: tempo de processamento de i em j

Considerando a alocação de (C,separador) e uma vez que a operação (A,reator) não pode ser desenvolvida simultaneamente com a primeira operação, pode-se assim determinar um intervalo de tempo onde o desenvolvimento das duas simultaneamente não é possível. Este intervalo determina assim o conjunto de períodos de tempo infactíveis à execução de A no reator. Considerando o exemplo representado pela figura 7.3, onde a operação (C,separador) foi alocada no período de tempo k=15, o conjunto de períodos de tempo em que a operação (A,reator) não pode ser iniciada

pode ser avaliado pela expressão 7.1. O tempo de processamento da operação (C,separador) é dado na tabela 7.1 e é de 4h. O reator é considerado o segundo equipamento, simbolizado por 2.

$$S1_{A(2)} = \{15,16,17,18\}$$

As variáveis binárias de alocação correspondentes à ocupação da operação (A,reator) nestes períodos de tempo podem então ser eliminadas do processo de busca, ou seja, podem ser fixadas em 0. No entanto, a tabela 7.6 mostra que algumas destas variáveis apresentam valores fracionários.

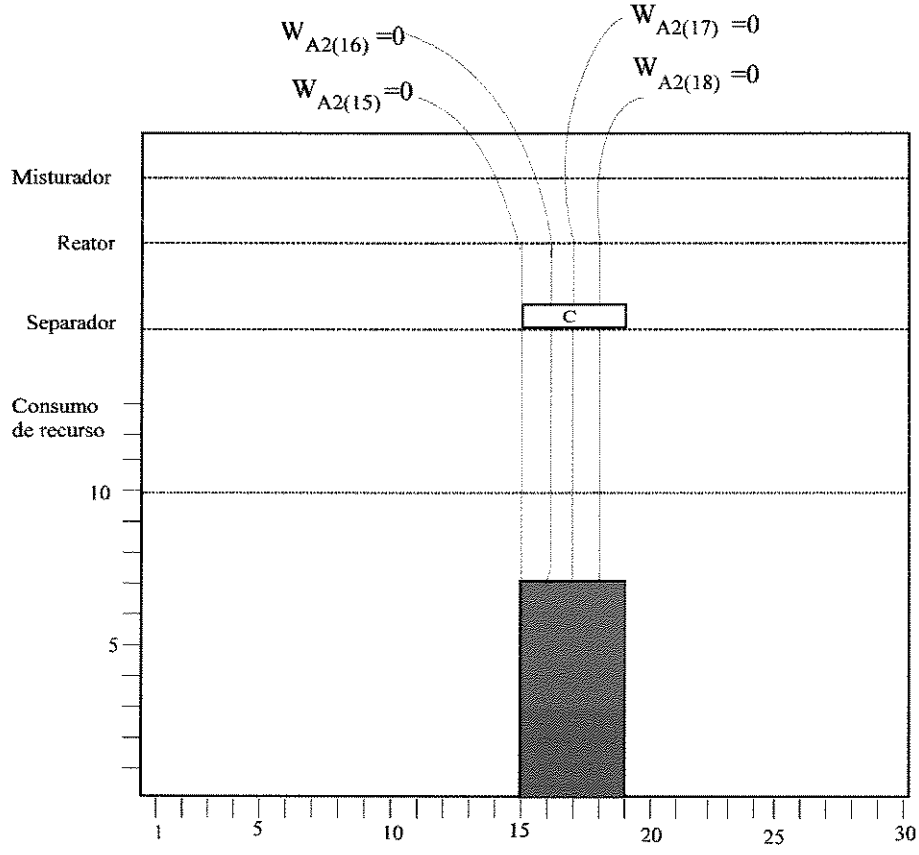


Figura 7.5: Fixação de variáveis binárias em 0 pela análise do conjunto $S1_{A1}$

A figura 7.5 mostra a fixação destas variáveis em 0. A mesma análise é feita para as demais operações que tem seu início de processamento impedido devido à alocação de (C,separado).

b) Conjunto 2 de variáveis de alocação não candidatas à ramificação

De acordo com as características de processamento da planta, onde as operações são desenvolvidas sem interrupção uma vez iniciado o seu processamento, pode-se a partir de um período de alocação de alguma operação determinar um conjunto de períodos onde operações que consomem quantidades de recurso além da oferta não podem ser iniciadas. Este conjunto de períodos infactíveis para uma determinada operação (i',j') é dado abaixo.

$$S2_{i'j'} = \{k - TP_{i'j'} + 1, \ldots, k-1\} \quad \forall\, i' \neq i,\ j' < j \tag{7.2}$$

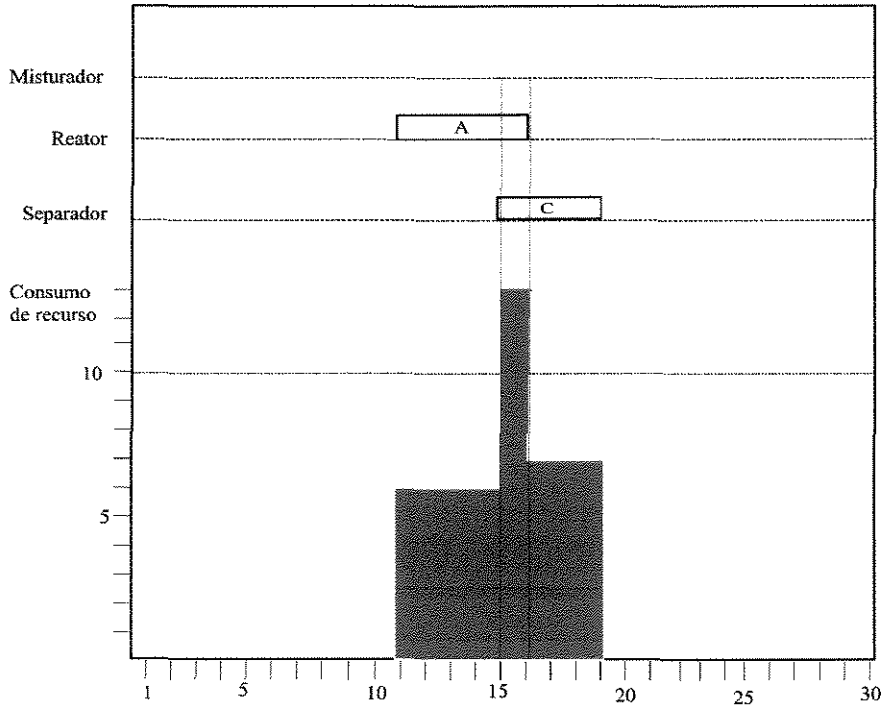


Figura 7.6: Programa de Produção parcial infactível

Onde:

$S2_{i'j'}$: Conjunto 2 de períodos de alocação infactíveis para a operação (i',j')
i: tarefa alocada
j: processador onde i foi alocada
k: período de alocação
i': tarefa ainda não alocada
j': processador onde i' será alocada
$TP_{i'j'}$: tempo de processamento da tarefa i' no processador j'

Os períodos de tempo infactíveis pertencentes ao conjunto $S2_{A2}$ são dados abaixo:

$$S2_{A2} = \{11,12,13,14\}$$

A alocação da operação (A,reator) no período k=11, por exemplo, leva a um programa de produção parcial infactível, mostrado na figura 7.6. Procurando mostrar que existe esta possibilidade de alocação, ou seja, o OSL considera as variáveis $W_{A(2)(11)}$ e $W_{A(2)(14)}$ pertencentes ao conjunto de variáveis candidatas à ramificação na direção 1, a tabela 7.7 apresenta para o nó 5 os valores das variáveis da operação (A,reator) em períodos de tempo que pertencem ao conjunto $S2_{A(2)}$.

Tabela 7.7: Valores das variáveis binárias de alocação da Operação (A,reator)

Períodos	Variáveis binárias de alocação	Valores das variáveis no nó 5
11	$W_{A(2)(11)}$	0.110
12	$W_{A(2)(12)}$	0.035
13	$W_{A(2)(13)}$	0.0
14	$W_{A(2(14)}$	0.140

Através da análise feita sobre o recurso compartilhado r, uma vez alocada a operação (C,separador) no período 15, todas as variáveis apresentadas na tabela 7.7 deveriam possuir valores iguais a zero, evitando com isso que o OSL as considere variáveis candidatas à ramificação. A figura 7.7 mostra a representação gráfica destas variáveis sendo fixadas em 0, que segundo o desenvolvimento da metodologia "branch and bound", elas não assumem outros valores em nós vinculados ao nó 5, ou melhor, que apresentem alguma relação com ele.

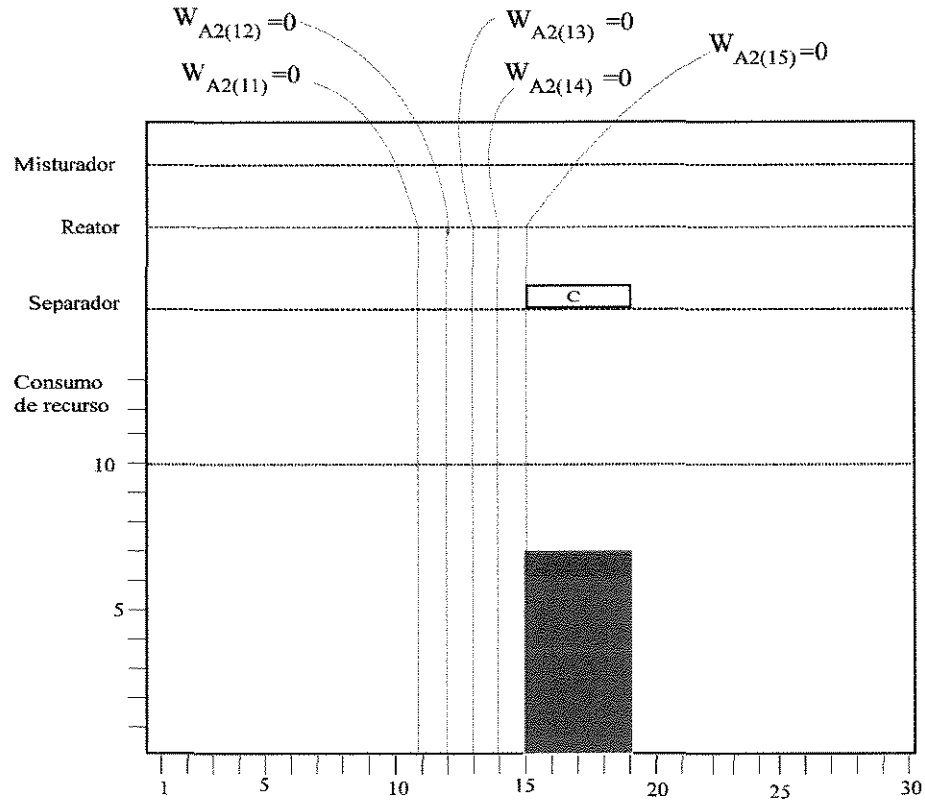


Figura 7.7: Fixação de variáveis binárias pela análise do conjunto $S2_{A2}$

7.2.3. Relações Lógicas Envolvendo Recursos Compartilhados

7.2.3.1. Relações Lógicas entre Operações que não podem ser Desenvolvidas Simultaneamente

Buscando um melhor entendimento no estabelecimento de relações lógicas envolvendo recursos compartilhados, primeiramente são desenvolvidas expressões lógicas, as quais comportam testes relacionais entre operações que não podem ser desenvolvidas simultaneamente durante a programação de produção devido à limitação da oferta de recurso. Seguindo esta linha de raciocínio, a seguinte relação lógica é estabelecida, considerando que a quantidade de recurso exigida por uma determinada operação (i',j') mais a quantidade consumida pela operação alocada supera a oferta do recurso r.

$$W_{ijk} \Rightarrow \neg\left(W_{i',j',k}\right) \qquad \forall\ i' \neq \mathrm{i}, \mathrm{j}' \neq \mathrm{j},\ (i',j') \in CO,k \qquad (7.3)$$

Onde:

$$CO = \left\{(i', j'): CR_{ijr} + CR_{i',j',r} > of_r, \quad \forall i' \in N, j' \in M, i \neq i', j \neq j', r\right\}$$

CO: conjunto de operações (i',j') que não podem ser desenvolvidas simultaneamente com a operação (i,j)
N: conjunto de tarefas
M: conjunto de equipamentos
i: índice de tarefa
j:índice de processador
r: índice de recursos
k: índice de período de alocação
W_{ijk}: variável binária de alocação da operação (i,j) em k
Cr_{ijr}: quantidade de recurso r exigida pela operação (i,j)
of_r: quantidade recurso r disponível

A expressão 7.3 pode ser também colocada utilizando o operador lógico $\vee$ (ou) resultando:

$$\neg W_{ijk} \vee \neg W_{i',j',k} \qquad \forall\ i' \neq \text{i}, \text{j}' \neq \text{j},\ (i', j') \in CO, k \tag{7.4}$$

Utilizando-se da tabela (capítulo 5), que busca a representação de expressões lógicas via elementos matemáticos, a expressão 7.3 pode ser assim transposta para sua forma matemática:

$$W_{ijk} + W_{i',j',k} \leq 1 \qquad \forall\ i' \neq \text{i}, \text{j}' \neq \text{j},\ (i', j') \in CO, k \tag{7.5}$$

A expressão 7.5 garante a não coexistência na planta operações que consumam, juntamente com a operação (i,j) alocada no período k recurso r acima da quantidade ofertada.

Na modelagem utilizada uma vez que uma operação alocada em um determinado período k, os períodos pertencentes ao intervalo que inicia no período k até $k+TP_{ij}$ estão comprometidos com a execução da operação (i,j), pode-se então expandir a expressão para estes períodos.

$$\neg W_{ijk} \vee \neg\left(\sum_{k'} W_{i'j'k'}\right) \quad \forall\ i' \neq i, j' \neq j, (i', j') \in CO, k' \in S1_{i'j'}, S2_{i'j'} \qquad (7.6)$$

Onde:

k: período de alocação da operação (i,j)
Tp_{ij}: tempo de processamento da operação (i,j)

$$S1_{i'j'} = \{k, \ldots, k + TP_{ij} - 1\}$$

$$S2_{i'j'} = \{k - TP_{i'j'} + 1, \ldots, k - 1\}$$

Transpondo a expressão lógica 7.6 para sua forma matemática, tem-se

$$- W_{ijk} - \left(\sum_{k'} W_{i'j'k'}\right) \geq -1 \quad \forall\ i' \neq i, j' \neq j, (i', j') \in CO, k' \in S1, S2 \qquad (7.7)$$

ou

$$W_{ijk} + \left(\sum_{k'} W_{i'j'k'}\right) \leq 1 \quad \forall\ i' \neq i, j' \neq j, (i', j') \in I, k' \in S1, S2 \qquad (7.8)$$

Como exemplo da constituição do conjunto S1 e S2 considere o exemplo onde as operações (A,reator) e (C,separador) não podem ser desenvolvidas simultaneamente, pois o consumo de recurso r supera a quantidade ofertada (10 u.r). Se a tarefa A é alocada no reator na posição k=11 e sabendo-se que o seu tempo de processamento é de 5 u.t (tabela 7.1), o período de término desta operação é de 16. Logo o conjunto $S1_{A2}$ é formado pelos períodos de tempo {11,12,13,14,15} (dados pela expressão 7.1). Nestes períodos a tarefa C no separador não pode ser alocada, por falta de recurso.

O conjunto $S2_{A2}$ para o mesmo exemplo é representado pelos períodos {8,9,10}, ou seja, se a tarefa C for iniciada em algum destes períodos que inviabilizam a alocação da tarefa A no reator.

7.2.3.2 Relações Lógicas Envolvendo Disponibilidade de Recurso r nos Períodos de Tempo.

Seguindo o mesmo raciocínio da seção anterior, relações lógicas agora envolvendo a quantidade total de recurso r consumido até um determinado ponto da busca em árvore "branch and bound" são desenvolvidas. Nesta seção é mostrada a formalização das equações utilizadas no algoritmo A, que comporta testes relacionais envolvendo a quantidade de recurso consumida em cada período.

$$\left(of_r - QC_k\right) < CR_{i',j',r} \Rightarrow \neg W_{i',j',k} \qquad \forall\ (i', j') \in P, k, r \tag{7.9}$$

$$QC_{kr} = \sum_i \sum_j CR_{ijr} W_{ijk} \qquad \forall\ (i,j) \in A, k, r \tag{7.10}$$

Onde:

$$P = \left\{(i', j'): CR_{i',j',r} + QC_k > of_r,\ \forall (i', j') \in NA, k,\ r\right\}$$

P: Conjunto de operações que não podem ser alocadas em k devido à falta de recurso r.
A: Conjunto de operações (i,j) alocadas
NA: Conjunto de operações não alocadas
W_{ijk}: Variável binária de alocação da operação (i,j) no período k
Cr_{ijr}: Quantidade do recurso r exigida pela operação (i,j)
QC_{kr}: Quantidade consumida de recurso r no período k das operações alocadas

Nota-se que as expressões são desenvolvidas levando em consideração o consumo de recurso r disponível em um período para as operações alocadas, ou seja, que estão em processamento no respectivo período de tempo k. Com isso qualquer tentativa de transpor esta expressão para forma matemática, deve contemplar tal consideração extremamente relevante. Como no modelo apresentado no capítulo 3 (equação 3.7), a variável Q_{kr} em um determinado nó da árvore de busca, engloba não somente o consumo das operações que foram anteriormente alocadas (fixação de algum W_{ijk}=1), como também o consumo de operações cujos W_{ijk} apresentam com valores fracionários. Qualquer tentativa de eliminar variáveis de W_{ijk} utilizando a informação fornecida por Q_{kr} pode levar a resultados indesejáveis, uma vez que valores fracionários de W_{ijk} não trazem nenhuma informação se a operação estará ou não alocada no período de tempo igual a k em nós subsequentes àquele analisado. Portanto, a quantidade de

recurso r utilizada para inferir sobre as variáveis de alocação que não podem mais compor o conjunto de variáveis candidatas à ramificação, deve contemplar tão somente o consumo das operações que realmente estão em processamento no respectivo período de tempo k, ou seja, cujo W_{ijk} é fixado em 1. Esta análise leva o problema de inferência lógica sobre a quantidade de recurso compartilhado disponível no período k. A única possibilidade de desenvolvimento deste tipo de estratégia é utilizar pacotes ou códigos que permitam a análise do problema durante a sua solução, pois pacotes fechados tal como o GAMS, por exemplo, não permite acessar a sua árvore de busca "branch and bound".

7.3. Interfaceamento Arquivo "Mps"e Sistema OSL

Tendo em vista a necessidade de conhecer quais as operações que estão sendo desenvolvidas em um determinado período de tempo e consequentemente determinar a quantidade de recurso r consumida em um determinado período de tempo k, considere o seguinte exemplo de sequenciamento em uma planta de multiproduto, cuja a modelagem corresponde a utilização das expressões 3.13 a 3.17. O problema visa a minimização do "makespan" (expressão 3.20). O modelo assim constituído é compilado no GAMS, gerando um arquivo em formato MPS mostrado no apêndice B. As colunas de 1 a 20 são representantes das variáveis binárias do problema, ou seja, são as variáveis de alocação W_{ijk}. Este números (números de referência) como são os utilizados pelos pacotes otimizadores para indicar qual a variável é ramificada a partir de um nó, logo para se determinar qual a variável no formato de variável de alocação, W_{ijk} é necessário algum procedimento de transposição daquele formato "MPS" para o formato de variável W_{ijk}, podendo assim determinar todas as informações parciais sobre o programa de produção que está sendo constituído durante o desenvolvimento da árvore, inclusive o consumo de recurso r ainda disponível em um determinado período k, fundamental o desenvolvimento de sistema de interferência lógica apresentado neste trabalho. No desenvolvimento deste interfaceamento considere os dados mostrados na tabela 7.8.

Tabela 7.8: Tempos de Processamento

Tarefas	Reator(1)	Separador(2)
A(1)	1	1
B(1)	1	1

Considerando o arquivo no formato "MPS" para este problema (Apêndice B), as variáveis binárias W_{ijk} com os seus respectivos números de referência são mostradas na tabela 7.9.

Tabela 7.9: Associação de variáveis W_{ijk} à representação adotada pelo OSL

Variável	Significação	Representação 'MPS' e OSL
$W_{A,Reator,1}$	Alocação da Tarefa A no Reator no período 1	1
$W_{A,Reator,2}$	Alocação da Tarefa A no Reator no período 2	2
$W_{A,Reator,3}$	Alocação da Tarefa A no Reator no período 3	3
$W_{A,Reator,4}$	Alocação da Tarefa A no Reator no período 4	4
$W_{A,Reator,5}$	Alocação da Tarefa A no Reator no período 5	5
$W_{A,Separador,1}$	Alocação da Tarefa A no Separador no período 1	6
$W_{A,Separador,2}$	Alocação da Tarefa A no Separador no período 2	7
$W_{A,Separador,3}$	Alocação da Tarefa A no Separador no período 3	8
$W_{A,Separador,4}$	Alocação da Tarefa A no Separador no período 4	9
$W_{A,Separador,5}$	Alocação da Tarefa A no Separador no período 5	10
$W_{B,Reator,1}$	Alocação da Tarefa B no Reator no período 1	11
$W_{B,Reator,2}$	Alocação da Tarefa B no Reator no período 2	12
$W_{B,Reator,3}$	Alocação da Tarefa B no Reator no período 3	13
$W_{B,Reator,4}$	Alocação da Tarefa B no Reator no período 4	14
$W_{B,Reator,5}$	Alocação da Tarefa B no Reator no período 5	15
$W_{B,Separador,1}$	Alocação da Tarefa B no Reator no período 1	16
$W_{B,Separador,2}$	Alocação da Tarefa B no Reator no período 2	17
$W_{B,Separador,3}$	Alocação da Tarefa B no Reator no período 3	18
$W_{B,Separador,4}$	Alocação da Tarefa B no Reator no período 4	19
$W_{B,Separador,5}$	Alocação da Tarefa B no Reator no período 5	20

Logo para se referir a uma determinada variável, por exemplo a variável que determina a alocação da tarefa A no separador no período 2 (que equivale $W_{ASeparador2}$) que no arquivo "MPS" é denotada pelo número 5, é necessário que exista algum procedimento que transforme a forma de representação no formato "MPS" em variáveis W_{ijk} e de variáveis de alocação W_{ijk} para números de referência, isto para que se possa proceder as devidas fixações de variáveis.

Nesta seção são detalhados os procedimentos desenvolvidos para esta transformação da representação de uma variável no formato "MPS" em variáveis de alocação W_{ijk}. O objetivo portanto é, dado uma variável em formato do "OSL" (que é um número inteiro) encontrar:

- A tarefa i
- O processador j onde a tarefa i está sendo processada
- O período de tempo k de alocação da operação (i,j)

Com estas três informações é possível determinar a variável de alocação W_{ijk} e consequentemente todas as informações do programa de produção, como por exemplo o consumo de recurso r até então consumido até então em um determinado nó. Na verificação da funcionalidade do procedimento desenvolvido para este fim, considere o exemplo abaixo.

A figura 7.8 mostra a representação de uma variável de alocação no formato MPS/OSL e em termos de W_{ijk}.

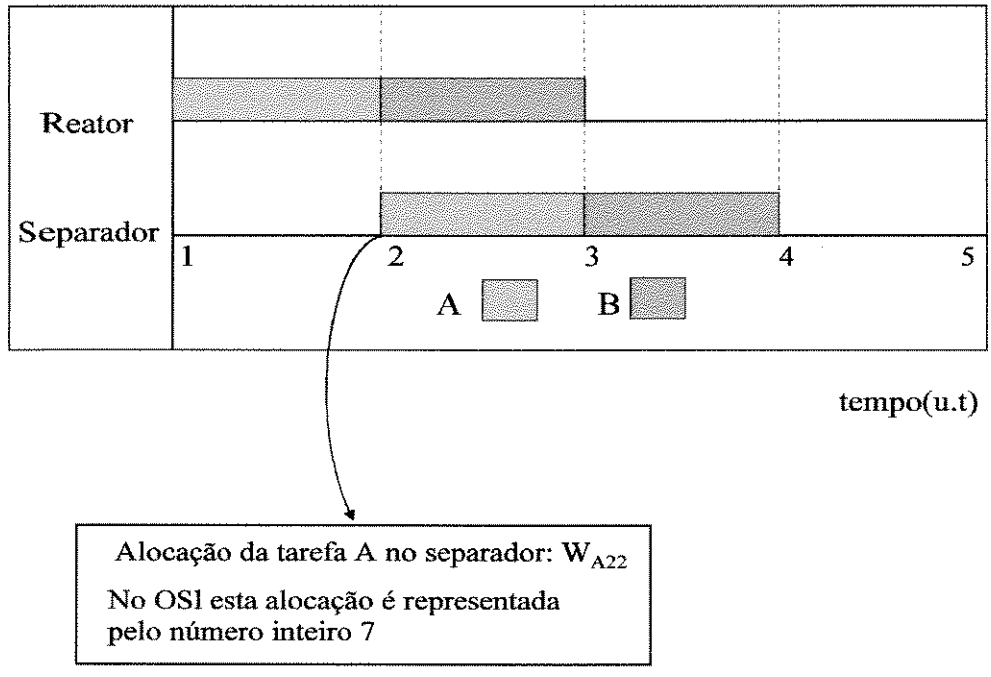


Figura 7.8: Associação de variável binária W_{A22} com o formato "MPS"

Considere o seguinte exemplo:

- Encontre W_{ijk} a partir de NUMVAR=7, onde NUMVAR é representação no formato MPS/OSL da variável escolhida para ramificação.
- Considere a cardinalidade (Card) do conjunto K como a dimensão do conjunto K.
- O número de períodos considerados no problema é 5, logo K={1,2,3,4,5} e Card(K)=5.

A tabela 7.10 mostra as operações para o problema sendo atribuído um número a cada uma delas, denominado "intervalo".

Tabela 7.10: Ordenação das Operações

Intervalo	Tarefa	Processador
1	A(1)	Reator(1)
2	A(1)	Separador(2)
3	B(2)	Reator(1)
4	B(2)	Separador(2)

a) Determinar NUMA1=NUMVAR/CARD(K)

Card(K)= 5

NUMVAR= 7

NUMA1= 1.4

b) Determinar o teto de NUMA1, o numero inteiro imediatamente acima de um número fracionário.

$$INTERVALO = \lceil NUMA1 \rceil \tag{7.11}$$

O intervalo fornecido pela expressão 7.11 para o exemplo é de 2. O operador $\lceil\ \rceil$ determina o número inteiro imediatamente acima de um número fracionário (teto de um número fracionário).

c) Determinação da tarefa e do processador correspondente ao INTERVALO 2

Utilizando a tabela 7.10 o valor do INTERVALO da operação igual a 2, tem-se:

- i= 1 (corresponde à tarefa A)
- j= 2 (corresponde ao processador 2)

d) A determinação do período de alocação é feita pela equação 7.12

$$k = NUMVAR - CARD(K) \left\lfloor \frac{NUMVAR}{CARD(K)} \right\rfloor \tag{7.12}$$

No caso de k apresentar valor 0 pela expressão 7.12, k recebe o valor do horizonte de tempo H.

O operador $\lfloor\ \rfloor$ determina o número inteiro imediatamente abaixo de um número fracionário (piso de um número fracionário).

O valor de k dado pela expressão 7.12 é de 2. Logo a variável escolhida para a ramificação é W_{A22}.

A transformação inversa, a partir de uma variável W_{ijk} encontrar a representação desta no formato MPS/OSL pode ser feito pela equação abaixo:

$$NUMVAR = Card(K) \quad (INTERVALO \quad -1) \quad + k \tag{7.13}$$

Onde k é o período de alocação da operação.

- Determinar número, no formato MPS/OSL, correspondente à variável W_{A22}?

A operação (A,2) encontra-se no intervalo 2 da tabela 7.10 (INTERVALO=2) e sendo k=2 a equação 7.13 fornece o número de referência(NUMVAR =3.(2-1) + 2) igual a 5.

7.4. Incorporação de Relações Lógicas envolvendo recursos compartilhados à Metodologia "Branch And Bound".

7.4.1. Introdução

A seguir as expressões lógicas provenientes de toda a análise feita anteriormente envolvendo o compartilhamento de recursos são incorporadas à metodologia "branch and bound". Como anteriormente apresentado, o OSL é um sistema que permite interferências durante o desenvolvimento da árvore de busca, para tanto são fornecidas as chamadas subrotinas "user exit" que possibilitam que o usuário altere procedimentos padrões do OSL. Entre estas subrotinas estão:

- EKKBRNU: Permite que regras de seleção da variável a ser ramificada sejam implementadas pelo usuário.
- EKKCHNU: Permite que a regra padrão de escolha do nó do sistema OSL seja modificada.
- EKKEVNU: Permite que sejam feitas análises em cada nó da árvore "branch and bound". É nesta subrotina são implementadas as expressões deduzidas anteriormente(expressões 7.9 e 7.10).

7.4.2. Descrição do Sistema com Interferência Lógica Externa(Algoritmo A)

No algoritmo A detalhado a seguir são mantidas a regra de escolha da variável a ser ramificada e a regra de escolha do nó padrões do Sistema OSL. Mesmo assim as 'user exit' EKKBRNU e EKKCHNU foram utilizadas para fins específicos ligados à implementação do sistema desenvolvido.

Procurando eliminar o número de nós infactíveis gerados na árvore de busca, quando restrições de recursos são impostas ao modelo, é desenvolvido nesta seção um sistema de interferência lógica o qual permite que as expressões sejam inseridas na busca padrão "branch and bound" do OSL. Os principais passos do Sistema de Interferência Lógica(Algoritmo A) são:

Passo 0

- A Leitura de dados(a estrutura de dados utilizado no algoritmo é apresentado no Apêndice C.
- Definição do Problema(número de blocos da matriz, número de modelos na aplicação).
- Leitura do modelo via arquivo "MPS".
- Solução do modelo pela subrotina EKKSSLV(esta solução relaxada constitui o nó 0 da árvore desenvolvida pela subrotina EKKMSLV). A variável escolhida para. Vá para o Passo 1.

Passo 1

- Desenvolvimento da pesquisa "branch and bound" pela subrotina EKKMSLV(busca de uma solução que satisfaça a integralidade das variáveis inteiras). Vá para o Passo 2.

Passo 2:

- Chamar a subrotina EKKBRNU para escolha da variável a ser ramificada no nó 0(zero), segundo regras previamente estabelecidas. Vá para o Passo 3.

Passo 3

Existem nós não analisados na lista de nós ativos?

Sim, continue a pesquisa passando para o Passo 4.

Não, vá para o Passo 14(pare a pesquisa).

Passo 4

- Na lista de nós ativos, segundo a regra de escolha previamente adotada, escolher o nó que vai ser sondado(esta etapa é desenvolvida pela "user exit" EKKCHNU). Vá para o Passo 5.

Passo 5

- Depois da chamada da subrotina EKKCHNU, a subrotina EKKEVNU que soluciona os subproblemas lineares é chamada. Antes de solucionar o subproblema linear correspondente a um determinado nó, é desenvolvido o **Procedimento de Inferência Lógica A1**(expressões), o qual é detalhado posteriormente a seção 7.4.3. Vá para o Passo 6.

Passo 6

- Solução do subproblema linear correspondente ao nó da árvore. Vá par o passo 7.

Passo 7

- A solução é factível(satisfação das restrições)?

Sim, vá para o para o Passo 8

Não, vá para o Passo 3

Passo 8

- A solução é inteira(restrições de integralidades satisfeitas)?

Sim, vá para o passo 9

Não, vá para o nó 10.

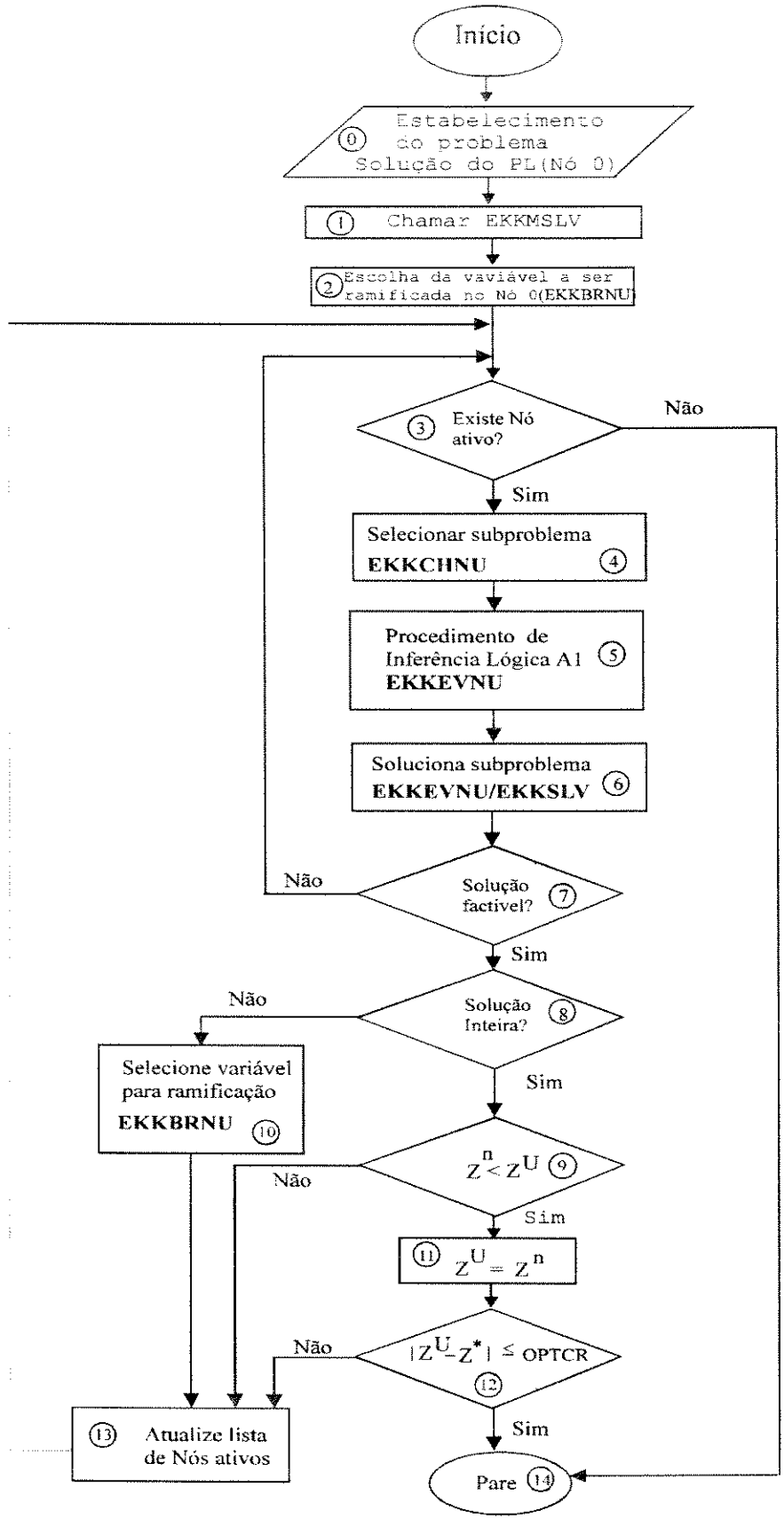


Figura 7.9: Fluxograma do Algoritmo A

Passo 9

- Solução inteira obtida no nó (Z^n) corrente é menor que o upper bound (Z^U)

  Sim, vá para o Passo 11

  Não, vá para o Passo 13

Passo 10

- Selecione a variável a ser ramificada no nó corrente. Vá para o Passo 13.

Passo 11

- limitante superior(melhor solução encontrada anteriormente) recebe solução do subproblema linear do nó corrente (Z^n). Vá para o Passo 12.

Passo 12

- $/Z^U - Z^*/$ é menor que a tolerância?

  Sim, vá para o Passo 14.

  Não, vá para o Passo 13

Passo 13

- Atualização da lista de nós ativos. Vá para o Passo 3.

Passo 14

- Pare a busca e imprima solução.

Todo o procedimento acima descrito é esquematizado pela figura 7.9.

7.4.3. Procedimento de Interferência Lógica A1

Passo 1

- Inicialização de variáveis. Vá para o Passo 2.

Passo 2

- Escolha do nó a ser ramificado
- NUMVAR(Número da variável ramificada no formato "MPS")
- DIREÇÃO(direção de ramificação).

Passo 3

- Se direção de ramificação da variável for 1, vá o Passo 4, se não vá(Passo 8).

Passo 4

- Calcular os valores de i, j e k a partir de NUMVAR(variável binária no formato MPS) como apresentado na seção 7.3.

Passo 5

- Calcular as operações do conjunto A(operações que já foram anteriormente alocadas), ou seja, fixadas em 1. Vá para o Passo 6.

Passo 6

- Avaliar a disponibilidade de recurso r em cada período de tempo
- Determinação dos conjuntos de períodos de tempo infactíveis para cada operação ainda não alocada.
- Determinação dos períodos k pertencentes a S1 e S2.

- A partir das variáveis binárias que representam a alocação das operações do conjunto P(as quais não podem ser alocadas em k devido à indisponibilidade de recurso r) calcula-se os valores de referência no formato MPS destas variáveis. Vá para o Passo 7.

Passo 7

- Uma vez feita a identificação de variáveis que podem ser fixadas em 1 em determinado período de tempo k, ao limitante superior("upper bound") de cada uma destas variáveis é atribuído o valor 0. No OSL isto é possível segundo o comando

mostrado a seguir, o qual é parte do procedimento de inferência lógica da subrotina EKKEVNU(apêndice D).

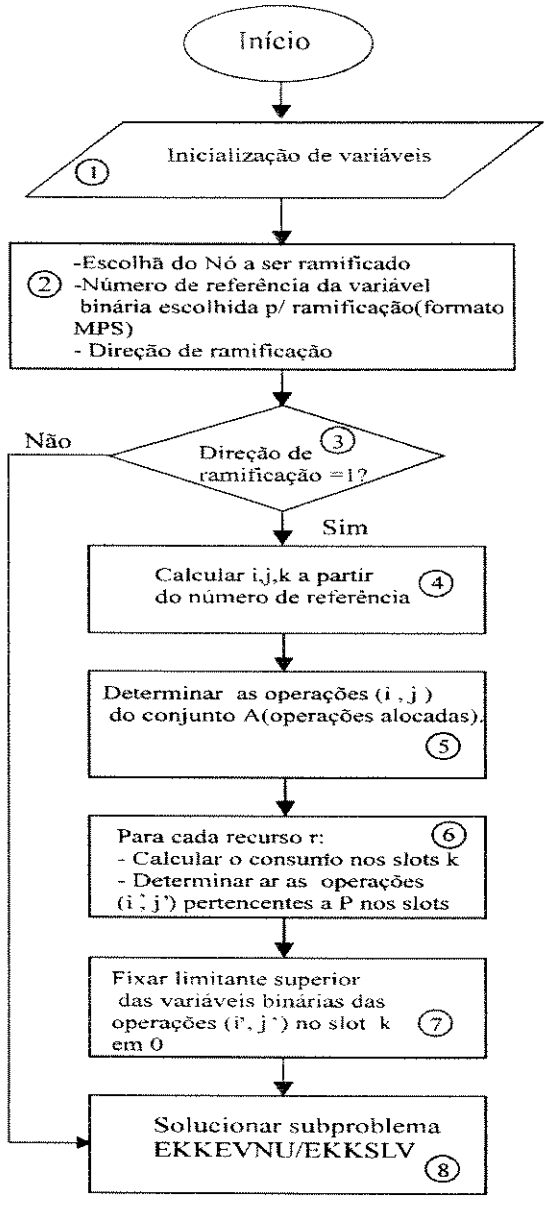


Figura 7.10: Procedimento Lógico A1

A figura 7.10 mostra a o desenvolvimento de procedimento A1

O algoritmo A desenvolvido neste trabalho é constituído pelo programa principal apresentado no apêndice C e pelas subrotinas: EKKEVNU(apêndice D), EKKCHNU(apêndice E) e EKKBRNU(apêndice F).

A estrutura de dados utilizado pelo algoritmo A é apresentado no apêndice G.

7.5. Aplicação do Sistema de Interferência Lógica(Algoritmo A) em Problema de Programação de Produção.

7.5.1. Introdução

Utilizando o sistema de interferência lógica externa são apresentados nesta seção os resultados obtidos. Neste ponto é ressaltado que não se pretende com os problemas aqui analisados limitar e nem explorar todas as potencialidades da modelagem Rede-Estado-Tarefa no tratamento de problemas de programação de produção. Os problemas aqui solucionados visam a verificação da potencialidade da metodologia de interferência lógica externa abordada neste trabalho, as quais procuram explorar as características de processamento de cada problema buscando sempre a diminuição de complexidade de solução.

Os oito exemplos apresentados apresentam diferenças, as quais interferem na obtenção dos resultados, tais como:

- Dimensão do problema número de variáveis binárias e de não-zeros). À medida que os problemas vão sendo apresentados estes vão aumentando de dimensão.
- A quantidade de recurso ofertada. Esta interfere diretamente na potencialidade de inferência sobre as variáveis binárias.

Os problemas são analisados em termos de nós gerados e iterações gastas na solução dos mesmos. O tempo de CPU (em segundos) também é mostrado, mas devido a não uniformidade em termos de plataformas onde os problemas foram solucionados, este parâmetro de análise não é utilizado para a comparação desses problemas. Isto ocorre em alguns exemplos, como será visto durante a apresentação e análise dos resultados.

7.5.2. Exemplos e Análise dos Resultados obtidos com Aplicação do Sistema de Interferência Lógica (Algoritmo A) em problemas de Programação de Produção em Sistemas Flexíveis de Produção.

7.5.2.1. Exemplo 1: O exemplo mostrado a seguir é solucionado utilizando o modelo constituído das expressões 4.1 a 4.3, 4.5 e 4.8. A receita de produção da planta é mostrada na figura 7.10, onde se pode obter as percentagens de materiais necessários

para o desenvolvimento de cada tarefa, assim como os tempos de processamento de uma operação. O problema consiste em maximizar o lucro de produção com a venda dos produtos A e B. Este lucro é proposto pela equação abaixo:

$$Lucro = \sum_{s} PP_{sH+1} S_{s,H+1} - \sum_{s} PI_{sH+1} S_{sH+1} - \sum_{r} \sum_{k} (PU) Q_{kr} \qquad (7.14)$$

Onde:

PP_{sH+1}: Preço de produtos no final do horizonte de produção.
PI_{sH+1}: Custo de estoque de material intermediário no final do horizonte de produção.
PU: Preço de utilidade (energia de refrigeração, vapor)
S_{sH+1}: Estoque de estado s no final do horizonte de produção.
Q_{kr} : Quantidade de recurso r consumida no período de tempo k

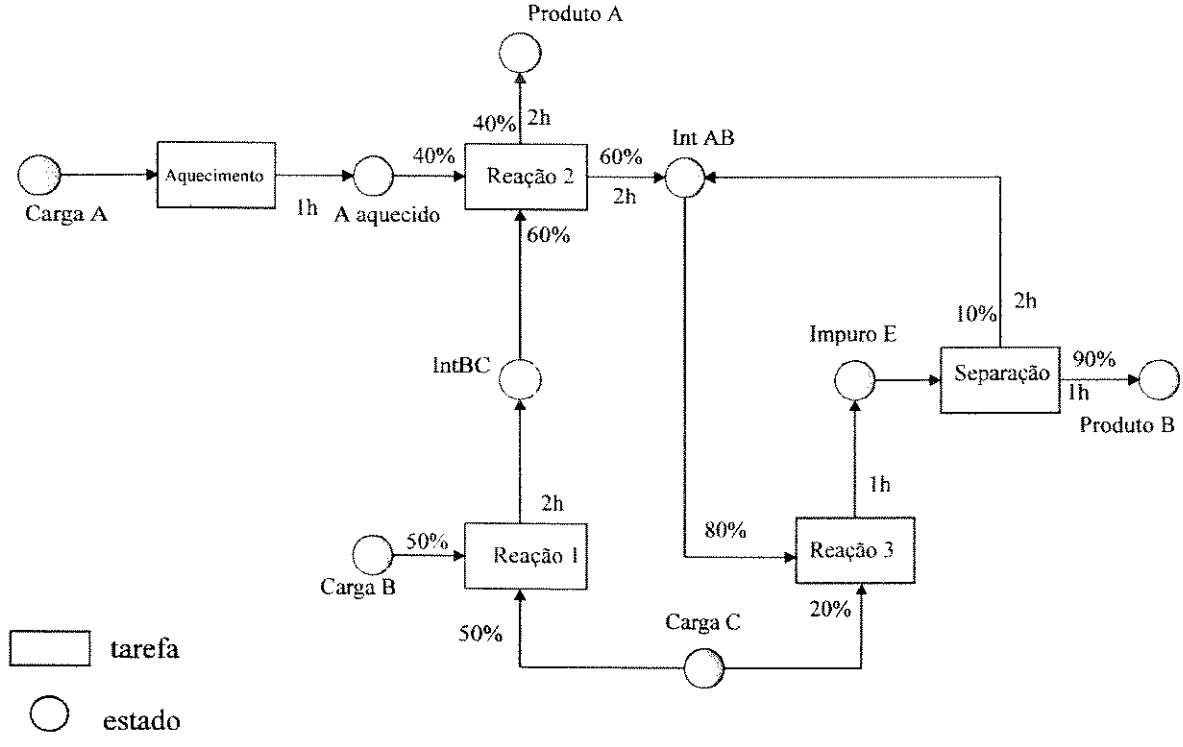


Figura 7.10: Representação da Estrutura de Produção via Rede-Estado-Tarefa

As operações e seus tempos de processamento em cada equipamento são fornecidos pela tabela 7.11.

Tabela 7.11: Tempos de Processamento das Operações

Tarefas	Trocador	Reator1	Reator2	Separador
Aquecimento	1	-	-	-
Reação 1	-	2	2	-
Reação 2	-	2	2	-
Reação 3	-	1	1	-
Separação	-	-	-	2

A tabela 7.12 mostra os equipamentos disponíveis na planta, as tarefas que podem ser desenvolvidas nestes equipamentos e capacidade de cada um.

Tabela 7.12: Dados estruturais da planta

Equipamentos	Tarefas Convenientes	Capacidade de Produção(kg)
Trocador de Calor	aquecimento	100
Reator 1	reação 1,2 e 3	80
Reator 2	reação 1,2 e 3	50
Separador	separação	200

Existe limite na capacidade de armazenagem para alguns estados intermediários, ou seja, a armazenagem não é ilimitada durante o processo produtivo. A tabela 7.13 mostra estes dados.

Tabela 7.13: Limite de capacidade de armazenamento dos estados

Estados	Capacidade de armazenamento (kg)
Cargas A, B e C	armazenagem ilimitada
A aquecido (Hot A)	100
Intermediário AB	200
Intermediário BC	150
Impuro E	100
Produtos A e B	armazenagem ilimitada

Os recursos existem de forma limitada e a demanda de cada de energia de cada operação é dada na tabela 7.14.

Tabela 7.14: Demanda de Recurso energia das Operações (kwh)

Tarefas	Trocador	Reator1	Reator2	Separador
Aquecimento	10	-	-	-
Reação 1	-	15	15	-
Reação 2	-	15	15	-
Reação 3	-	10	10	-
Separação	-	-	-	5

Tabela 7.15: Resultados Computacionais para o **Exemplo 1**

	GAMS/OSL	Sistema OSL sem Interferência Lógica	OSL com Interferência Lógica
Número de:			
Tarefas	5	5	5
Equipamentos	4	4	4
Número de:			
Restrições	285	285	285
Variáveis	271	271	271
Variáveis binárias	80	80	80
Não-zeros	915	915	915
Número de:			
Nós	955	5325	284
Iterações	4101	42983	3902
CPU (s)	108.0 (*)	666.4 (+)	38.1 (+)
Iterações/s	37	65	93
Solução:			
Relaxada	2516.9	2516.9	2516.9
Inteira	1756.0	1756.0	1756.0

(*) PC 486(DX2) 66 MHz (+) RISC/6000

O Preço dos produtos (PP) A e B é de 10 R$/kg e o custo de armazenamento de matérias intermediários (PI) no final do horizonte de produção é de 1R$/kg. O custo de utilidade considerado foi de 10^{-4} R$/kwh.

Aplicando o Sistema de Interferência Lógica neste problema, os seguintes resultados foram obtidos.

A redução no número de nós, comparando o resultado dado pelo Algoritmo A e o fornecido pelo pacote OSL (subrotinas padrões) é de 95% e em relação ao GAMS/OSL é de 70%. O optcr (diferença percentual entre a solução e a melhor anteriormente encontrada) utilizado nas análises acima é de 0.0.

Quando se compara os resultados entre o GAMS/OSL com o pacote OSL mostra que a interface GAMS traz em sua implementação ajustes e heurísticas que melhoram o desempenho do biblioteca OSL.

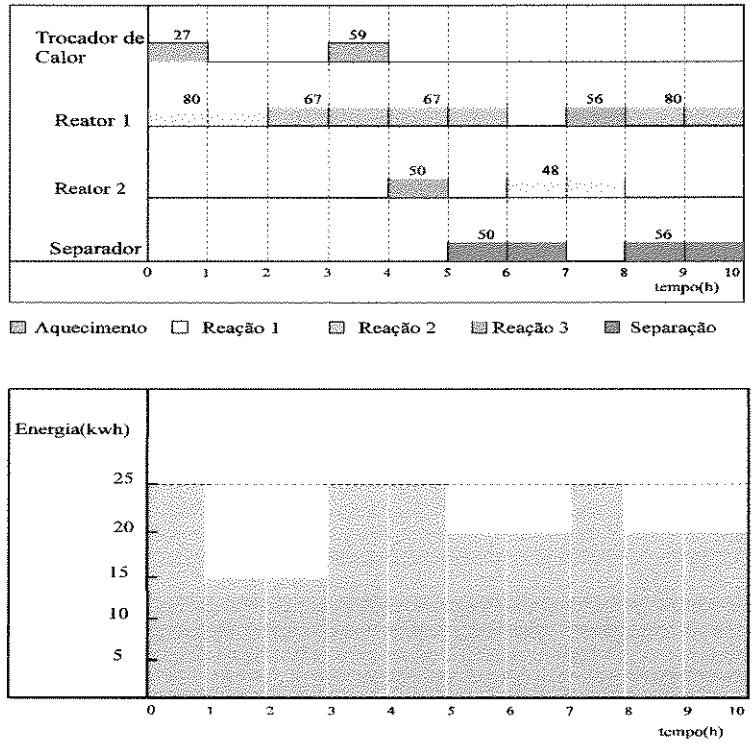


Figura 7.11: Programa de Produção ótimo e Distribuição de recurso

O número que aparece acima de cada operação na carta de Gantt é a quantidade de material em processamento no equipamento (B_{ijk}).

O programa de produção ótimo para este problema é mostrado na figura 7.11, juntamente com a utilização de energia. As quantidades de produtos A e B obtidas são de 85.0 kg e 96.0 kg respectivamente, dando um lucro de produção de 1756.0 R$.

7.5.2.2. Exemplo 2: Sequenciamento de 3 tarefas em planta uma multiproduto com 3 equipamentos visando a minimização do tempo total de produção. A oferta de recurso é de 15 u.r e o horizonte de produção considerado é de 25 h. O modelo para este problema consta das expressões 4.13 a 4.20.

Tabela 7.16: Dados da Planta

Tarefas	Tempos de Processamento			Demanda de recurso		
	P1	P2	P3	P1	P2	P3
A	4	5	2	8	6	5
B	2	3	7	3	4	3
C	3	5	7	3	2	5

Os resultados para o problema é mostrado na tabela 7.17 abaixo

Tabela 7.17: Resultados computacionais para o **Exemplo 2**

	GAMS/OSL	Sistema OSL sem Interferência Lógica	OSL com Interferência Lógica
Número de:			
Tarefas	3	3	3
Equipamentos	3	3	3
Restrições	362	362	362
Variáveis	476	476	476
Variáveis Binárias	225	225	225
Não-zeros	2895	2895	2895
Número de:			
Nós	494	550	62
Iterações	17834	19455	37827
CPU(s)	269.1(+)	149.6(*)	38.1(*)
Iterações/s	66	130	992
Solução Relaxada	17.6	17.6	17.6
Solução Inteira	24.0	24.0	24.0

(*) RISC/6000

(+) PC 486(DX2) 66 MHz

A redução no número de nós, comparando o resultado dado pelo Algoritmo A e o fornecido pelo pacote OSL (subrotinas padrões) é de 79% e em relação ao GAMS/OSL é de 87%. O opter utilizado nas análises acima é de 0.0.

7.5.2.3. Exemplo 3: Sequenciamento de 4 tarefas em planta multiproduto com 3 equipamentos visando a minimização do tempo total de produção. A oferta de recurso é de 15 u.r e o horizonte de produção é de 25 h. O modelo para este problema consta das expressões 4.13 a 4.20.

Tabela 7.18: Dados da Planta

Tarefas	Tempos de Processamento			Demanda de recurso		
	P1	P2	P3	P1	P2	P3
A	4	5	2	8	6	5
B	2	3	7	3	4	3
C	3	5	7	3	2	5
D	5	2	4	7	3	6

Tabela 7.19: Resultados Computacionais para o **Exemplo 3**

	GAMS/OSL	Sistema OSL sem Interferência Lógica	OSL com Interferência Lógica
Número de:			
Tarefas	4	4	4
Equipamentos	3	3	3
Restrições	611	611	611
Variáveis	876	876	876
Variáveis Binárias	420	420	420
Não-zeros	5377	5377	5377
Número de:			
Nós	4007	58887	1776
Iterações	140000	2530794	156572
CPU(s)*	2239.2(+)	29816.3(*)	1929.6(*)
iterações/s	62	849	81
Solução Relaxada	19.4	19.4	19.4
Solução Inteira	32.0	32.0	31.0

O problema foi resolvido, utilizando sistema OSL com e sem inferência lógica, em uma RISC/6000 e, utilizando o GAMS/OSL, em um 486 (DX2) 66 MHz.

A redução no número de nós, comparando o resultado dado pelo Algoritmo A e o fornecido pelo pacote OSL (subrotinas padrões) é de 97% e em relação ao GAMS/OSL é de 56%. O optcr utilizado nas análises acima é de 0.0.

7.5.2.4. Exemplo 4: Sequenciamento de 8 tarefas em planta multiproduto com 5 equipamentos, onde os equipamentos 3 e 4 estão em paralelo. O problema visa a minimização do tempo total de produção. A oferta de recurso é de 9 homens e o horizonte de produção é de 30 h. O modelo para este problema consta das expressões 4.13 a 4.19, 4.21, 4.22.

Tabela 7.20: Tempos de Processamento

Tarefas	P1	P2	P3	P4	P5
A	3	4	2	2	1
B	1	2	-	-	1
C	3	2	1	1	-
D	1	2	2	2	-
E	-	3	2	2	1
F	-	2	1	1	1
G	1	-	2	2	1
H	3	-	4	4	2

Tabela 7.21: Demanda de mão-de-obra

Tarefas	P1	P2	P3	P4	P5
A	2	2	1	1	3
B	2	3	-	-	2
C	1	1	2	2	-
D	1	2	1	1	-
E	-	1	3	3	1
F	-	2	1	1	1
G	2	-	2	2	1
H	1	-	1	1	3

Tabela 7.22: Resultados Computacionais para **Exemplo 4**

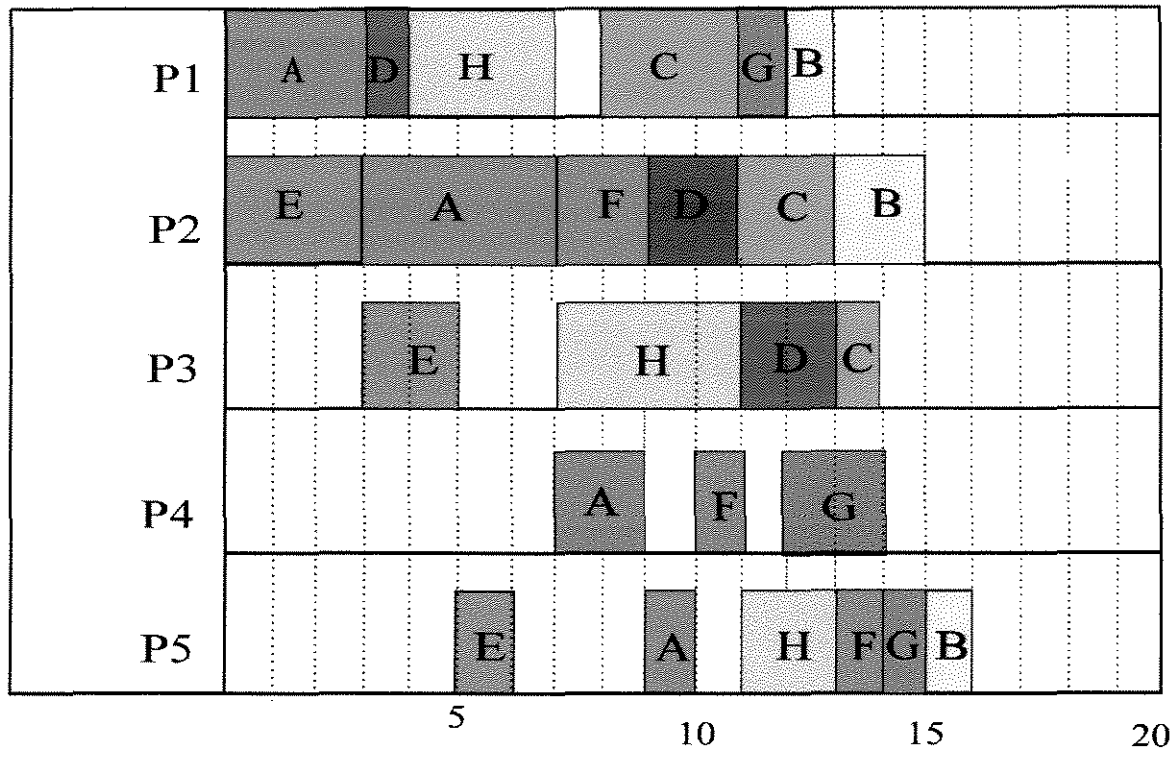
	GAMS/OSL	OSL com Interferência Lógica
Número de:		
Tarefas	8	8
Processadores	5	5
Restrições	1233	1233
Variáveis	1981	1981
Variáveis Binárias	960	960
Não zeros	8735	8735
Número de:		
Nós	2400	411
Iterações	360834	50948
CPU(s)	3342.5(+)	1602.5(*)
Iterações/s	108.0	32.0
Solução Relaxada	12.5	12.5
Solução Inteira	16.0	16.0

(+) SPARC/SOLARIS

(*) RISC/6000

A redução no número de nós, comparando o resultado dado pelo Algoritmo A e o GAMS/OSL é de 83%. O optcr utilizado nas análises acima é de 0.0.

O programa ótimo de produção para o exemplo 4 é dado pela figura 7.12.



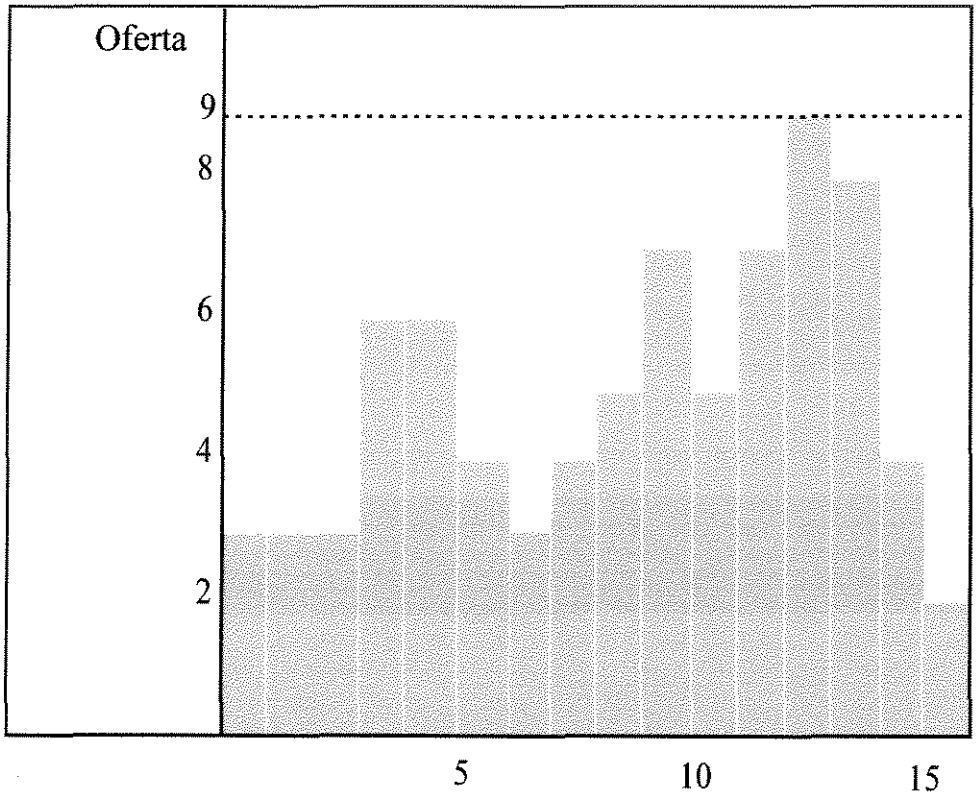


Figura 7.12: Programa de Produção ótimo para o problema 4

7.5.2.5. Exemplo 5: Sequenciamento de 8 tarefas em planta multiproduto com 5 equipamentos, onde os equipamentos 3 e 4 estão em paralelo. O problema visa a minimização do tempo total de produção. A oferta de recurso é de 7 homens e o horizonte de produção é de 40 h. O modelo para este problema consta das expressões 4.13 a 4.19, 4.21, 4.22.

Tabela 7.23: Tempos de Processamento

Tarefas	P1	P2	P3	P4	P5
A	3	4	2	2	1
B	1	2	-	-	1
C	3	2	1	1	-
D	1	2	2	2	-
E	-	3	2	2	1
F	-	2	1	1	1
G	1	-	2	2	1
H	3	-	4	4	2

Tabela 7.24: Demanda de mão-de-obra

Tarefas	P1	P2	P3	P4	P5
A	2	2	1	1	3
B	2	3	-	-	2
C	1	1	2	2	-
D	1	2	1	1	-
E	-	1	3	3	1
F	-	2	1	1	1
G	2	-	2	2	1
H	1	-	1	1	3

Tabela 7.25: Resultados Computacionais para o **Exemplo 5**

	GAMS/OSL	OSL com Interferência Lógica
Número de:		
Tarefas	8	8
Processadores	5	5
Restrições	1233	1233
Variáveis	1981	1981
Variáveis Binárias	960	960
Não-zeros	8735	8735
Número de:		
Nós	19257	10977
Iterações	5185584	1027147
CPU(s)	50000(+)	33203(*)
Iterações/s	104	30
Solução Relaxada	12.5	12.5
Solução Inteira	17.0	17.0

(+) SPARC/SOLARIS

(*) RISC/6000

A redução no número de nós, comparando o resultado dado pelo Algoritmo A e o GAMS/OSL é de 43%. O optcr utilizado nas análises acima é de 0.0.

7.5.2.6. Exemplo 6: Sequenciamento de 8 tarefas em planta multiproduto com 5 equipamentos, onde os equipamentos 3 e 4 estão em paralelo. O problema visa a minimização do tempo total de produção. A oferta de recurso é de 10 homens e o horizonte de produção é de 35 h. O modelo para este problema consta das expressões 4.13 a 4.19, 4.21, 4.22.

Tabela 7.26: Tempos de Processamento

Tarefas	P1	P2	P3	P4	P5
A	3	4	3	3	5
B	5	2	-	-	2
C	1	2	6	6	-
D	3	2	5	5	-
E	-	4	3	3	3
F	-	2	5	5	2
G	2	-	8	8	3
H	3	-	4	4	2

Tabela 7.27: Demanda de mão-de-obra

Tarefas	P1	P2	P3	P4	P5
A	2	2	1	1	3
B	2	3	-	-	2
C	1	1	2	2	-
D	1	2	1	1	-
E	-	1	3	3	1
F	-	2	1	1	1
G	2	-	2	2	1
H	1	-	1	1	3

Tabela 7.28: Resultados Computacionais para o **Exemplo 6**

	GAMS/OSL	OSL com Interferência Lógica
Número de:		
Tarefas	8	8
Processadores	5	5
Restrições	1433	1433
Variáveis	2311	2311
Variáveis Binárias	1120	1120
Não zeros	13711	13711
Número de:		
Nós	-	19227
Iterações	-	3880306
CPU(s)	-	132109(*)
Solução Relaxada	12.5	12.5
Solução Inteira	não resolvido	25.0

(-) não apresentou solução

(*) RISC/6000

7.5.2.7. Exemplo 7: Sequenciamento de 4 tarefas em planta multiproduto com 5 equipamentos, onde os equipamentos 3 e 4 estão em paralelo. O problema visa a minimização do tempo total de produção. A oferta de recurso é de 5 homens e o horizonte de produção é de 40 h.. O modelo para este problema consta das expressões 4.13 a 4.19, 4.21, 4.22.

Tabela 7.29: Tempos de Processamento

Tarefas	P1	P2	P3	P4	P5
A	3	4	3	3	5
B	5	2	-	-	2
C	1	2	6	6	-
D	3	2	5	5	-

Tabela 7.30: Demanda de mão-de-obra

Tarefas	P1	P2	P3	P4	P5
A	3	2	1	1	3
B	2	1	-	-	1
C	2	1	2	2	-
D	2	1	3	3	-

Tabela 7.31: Resultados Computacionais para o **Exemplo 7**

	GAMS/OSL	OSL com Interferência Lógica
Número de:		
Tarefas	4	4
Processadores	5	5
Restrições	977	977
Variáveis	1361	1361
Variáveis Binárias	640	640
Não-zeros	7865	7865
Número de:		
Nós	4728	1652
Iterações	485478	181710
CPU(s)	3838	5026.5(*)
Iterações/s	127	36
Solução Relaxada	18.4	18.4
Solução Inteira	23.0	22.0

(+) SPARC/SOLARIS

(*) RISC/6000

A redução no número de nós, comparando o resultado dado pelo Algoritmo A e o GAMS/OSL é de 68%. O optcr utilizado nas análises acima é de 0.0.

7.5.2.8. Exemplo 8: Sequenciamento de 5 tarefas em planta multiproduto com 4 equipamentos visando a minimização do tempo total de produção. A oferta de energia é de 20 kwh e de mão-de-obra de 5 homens. O horizonte de produção é de 50 h. O modelo para este problema consta das expressões 4.13 a 4.20.

Tabela 7.32: Tempos de Processamento

Tarefas	P1	P2	P3	P4
A	2	8	5	4
B	6	3	3	6
C	9	4	3	5

Tabela 7.33: Demanda de Energia das operações

Tarefas	P1	P2	P3	P4
A	6	10	8	7
B	3	9	15	6
C	6	8	7	12

Tabela 7.34: Demanda de mão-de-obra

Tarefas	P1	P2	P3	P4
A	1	2	1	1
B	2	1	2	3
C	3	2	3	1

A tabela 7.35 mostra a evolução em termos de número de nós e iterações na solução do problema 5 utilizando o GAMS/OSL (sparc/solaris). A tabela 7.36 mostra a evolução para mesmo problema utilizando o Sistema de Inferência Lógica (algoritmo A).

Tabela 7.35: Evolução do Número de nós e Iterações no GAMS/OSL

Iterações	Nós	Solução Inteira
17664	157	52
76301	622	48
277632	2536	45
330188	2970	40
100000	8412	40

Tabela 7.36: Evolução do número de nós e Iterações (Algoritmo A)

Iterações	Nós	Solução Inteira
2649	9	53
2965	12	49
6650	58	48
8450	77	45
128129	1561	40
249723	2577	40

A tabela 7.37 complementa a apresentação de resultados para o problema 5. O optcr utilizado na solução do problema foi 0.1.

Tabela 7.37: Resultados Computacionais para o **Exemplo 8**

	GAMS/OSL	OSL com Interferência Lógica
Número de:		
Tarefas	3	3
Equipamentos	4	4
Restrições	1015	1015
Variáveis	1301	1301
Variáveis Binárias	600	600
Não-zeros	11440	11440
Número de:		
Nós	8412	2577
Iterações	100000	249723
CPU(s)	10163.8(+)	4266.7(*)
Iterações/s	10	58
Solução Relaxada	26.6	26.6
Solução Inteira	40.0	40.0

(+) SPARC/SOLARIS

(*) RISC/6000

A redução no número de nós, comparando o resultado dado pelo Algoritmo A e o GAMS/OSL é de 69%. O optcr utilizado nas análises acima é de 0.0.

7.6. Considerações e Análise de Expressões Lógicas Insertas no modelo

Como discutido no capítulo 5, o percurso existente entre a solução relaxada Raman e Grossmann (1991) incorporam inequações provenientes de expressões lógicas, gerando restrições lineares e adicionando ao modelo MILP (na tentativa de diminuir a diferença entre a solução relaxada e a solução inteira do problema ou gap). Outra estratégia sugerida pelo autor, é considerar a análise de expressões lógicas em cada nó da árvore "branch and bound" (inferência lógica). Nesta seção, são apresentados os resultados obtidos utilizando a estratégia de inserir equações lógicas

no modelo e em seguir aplicar o Algoritmo A que desenvolve a inferência lógica externa.

Nesta etapa é analisada a alternativa de inserção de expressões lineares, obtidas na seção 7.2.3 , no modelo. Esta abordagem tem como objetivo conseguir uma melhor solução relaxada e consequentemente uma diminuição no "gap" (diferença entre solução relaxada e solução inteira).

A aplicação do Sistema de Interferência Lógica (Algoritmo A) ao exemplo 1 com expressões lógicas envolvendo recursos compartilhados inseridas no modelo. No seguinte exemplo as expressões são inseridas no modelo.

$$W_{(\mathrm{Re}acao1)(\mathrm{Re}ator1)k} + W_{(\mathrm{Re}acao1)(\mathrm{Re}ator2)k} \leq 1 \qquad \forall \mathrm{k} \qquad (7.15)$$

$$W_{(\mathrm{Re}acao2)(\mathrm{Re}ator1)k} + W_{(\mathrm{Re}acao1)(\mathrm{Re}ator2)k} \leq 1 \qquad \forall \mathrm{k} \qquad (7.16)$$

Estas inequações são derivadas da expressão lógica apresentada na seção 7.2.3 (expressão 7.3), que formaliza a não coexistência de pares de operações que consomem recurso além da oferta estipulada no problema. Mesmo sabendo que quando inseridas no modelo as expressões causam um aumento no número de restrições, a alternativa tem como único intuito a diminuição do "gap". A tabela 7.38 mostra os resultados obtidos

Tabela 7.38: Resultados Computacionais

	GAMS/OSL	GAMS/OSL Com equações lógicas	Algoritmo A Com equações lógicas
Número de:			
Tarefas	5	5	5
Equipamentos	4	4	4
Número de:			
Restrições	285	305	305
Variáveis	271	271	271
Variáveis binárias	80	80	80
Número de:			
Nós	955	820	708
Iterações	4101	5687	7522
CPU(s)	108.0(*)	202.0	80.0(+)
Solução:			
Relaxada	2516.9	2496.0	2496.0
Inteira	1756.0	1756.0	1756.0

(+) GAMS/OSL - PC 486(DX2) 66 Mhz

(*) Algoritmo A - RISC/6000

Comparando este resultado com o da tabela 7.15, nota-se que mesmo o "gap" do modelo tenha sido reduzido quando se adicionou as restrições 7.15 e 7.16, o aumento do número de restrições foi determinante para o aumento do número de nós, iterações e tempo de CPU de solução do problema.

CAPÍTULO 8

ANÁLISE DOS RESULTADOS

E

CONCLUSÃO

A proposta deste trabalho tem como principal objetivo mostrar que entre a solução devidamente apresentada, dentro de um critério de satisfação admissível por parte do usuário, e os procedimentos utilizados pelos pacotes matemáticos existem pontos que podem e devem ser explorados. Este trabalho utiliza a modelagem via Rede-Estado-Tarefa (Kondili et al., 1993) como forma de representação de características de processo em sistemas flexíveis.

O estudo detalhado do processo de solução relativo ao processo de busca inserta nos pacotes destinados à solução desses modelos, como é o caso do pacote utilizado neste trabalho (o OSL), mostra deficiências oriundas da junção de elementos matemáticos (representados pela função objetivo e pelas restrições que garantem a fenomenologia do problema) e elemento de inteligência artificial (representado pela metodológica "branch and bound" que busca satisfazer as restrições de integralidade das variáveis binárias, que devem apresentar valores 0 ou 1 na solução). Sendo assim, se tais deficiências não forem devidamente eliminadas do processo de busca leva a um aumento excessivo na dificuldade computacional e muitas vezes impossibilita a solução do problema, como foi mostrado no exemplo 6, onde o GAMS/OSL não apresentou nenhuma solução para o problema.

Em todos os exemplos apresentados, no que se refere ao número de nós gerados na árvore de busca entre os pacotes utilizados e o OSL com inferência lógica, houve diminuição na quantidade de nós. Isto devido principalmente à diminuição de nós infactíveis que poderiam ser gerados se variáveis representantes de operações com demanda de recurso superior à quantidade disponível não tivessem sido eliminadas do processo de busca.

No que se refere às iterações necessárias para a solução do problema o que se nota quando se compara os resultados obtidos utilizando o OSL padrão e o OSL com o procedimento de inferência lógica é que com este último, o número de iterações pode aumentar consideravelmente. Isto pode ser visto nos exemplos 2, que é justificado pela forma que OSL desenvolve o procedimento SIMPLEX (canalizado). As variáveis inteiras são postas como restrições do tipo $0 \leq y_i \leq 1$, onde y é uma variável binária. Sendo assim ao fixar variáveis binarias em 0 (zero) e um determinado nó, em um nó

subsequente, maior será o número de necessário para a satisfação das novas restrições, que são $1 \leq y_i \leq 1$. Pois como é sabido que restrições de igualdade são rígidas e aumenta complexidade de solução de dos problemas.

Os resultados apresentados neste trabalho mostram um caminho promissor para a diminuição de complexidade de solução dos problemas que são fortemente de natureza combinatorial. No entanto, os resultados também mostram que somente com abordagem de inferência lógica não é possível a solução de problemas mais realistas no que diz respeito ao número de operações presentes no problema. Contudo é indubitável, pelos resultados conseguidos, que a metodologia de inferência lógica deve servir, pela própria conduta da metodologia como um procedimento que elimina ineficiências da programação matemática, diminuindo assim a complexidade de solução dos problemas

Os resultados também mostram que o sistema OSL é uma ferramenta que abre caminhos para a solução de problemas de programação matemática que são inexequíveis em outros pacotes, possibilitando desta forma o surgimento de novas pesquisas que utilizam essa flexibilidade do OSL.

SUGESTÕES

Uma possibilidade de imediata ampliação da abordagem aqui adotada é interferir externamente na escolha da variável a ser ramificada, buscando aquela que fixe um maior número de variáveis, de acordo com a quantidade de recurso ainda disponível, em nós subsequentes. Com isto diminuiria o número de variáveis candidatas à ramificação em nós ainda a serem analisados.

Como neste trabalho é mostrado que as restrições de recursos compartilhados mesmo sendo de fácil formulação, elas são elementos que complicam a solução do modelo. Concatenando este fato ao procedimento de avaliação do limitante(o qual é calculado por relaxação das variáveis inteiras), nota-se que esta forma de avaliação não contempla nenhum comportamento futuro envolvendo as operações ainda não processadas. Assim, de forma apriorística pode-se desenvolver uma função que indique um futuro conflito entre operações por um determinado recurso, a qual pode auxiliar a escolha da variável de um determinado nó, buscando, por exemplo, diminuir situações conflitantes devido aos recursos.

Apêndice A: Análise Lógica de Proposições

De acordo com a proposta do trabalho, faz-se necessário uma introdução ao estudo de satisfatoriedade de expressões lógicas e sua relação.

Uma proposição é uma sentença(afirmação) de algum evento, a qual pode assumir dois valores falso ou verdadeiro. Simultaneamente, uma proposição não pode assumir estes dois valores. Para a verificação das operações envolvidas entre duas mais proposições, considere dois eventos "p" e "q".

- Operador "negação"

Neste trabalho, o operador de negação de sentenças é representado pelo símbolo "¬". A tabela A1 mostra a ação do operador negação sobre uma sentença p.

Tabela A1: Análise lógica do operador negação

p	¬p
V	F
F	V

Nas diversas sentenças lógicas analisadas, uma proposição é verdadeira quando a ela for atribuída o valor "V" e falsa quando recebe o valor "F".

- Operadores Conectores

Considerando as mesmas sentenças "p" e "q", a proposição formada por (p ∧ q) (p e q) denota que a sentença é verdadeira se e somente se p e q forem verdadeiras, como é mostrado na tabela A2.

Tabela A2: Análise lógica do operador "e"

p	q	p ∧ q
V	V	V
V	F	F
F	V	F
F	F	F

Para o operador "ou" a análise é oposta à anterior, pois esta proposição só assume o valor verdadeiro quando uma ou as duas sentenças forem verdadeiras. A tabela A3 mostra este resultado.

Tabela A3: Análise lógica do operador "ou"

p	q	p ∨ q
V	V	V
V	F	V
F	V	V
F	F	F

Um conectivo bastante comum na teoria de lógica matemática é o "ou exclusivo", simbolizado por ⊕. Duas sentenças conectadas por este operador indica que uma das duas deve ser verdadeira. A tabela A4 mostra o comportamento deste conectivo.

Tabela A4: Análise de verdade do operador "⊕"

p	q	p ⊕ q
V	V	F
V	F	V
F	V	V
F	F	F

Uma visão bem prática do uso do "ou exclusivo" numa relação lógica por exemplo, é a proibição de utilização de dois equipamentos simultaneamente para executar uma mesma tarefa (produto). Neste caso deve-se escolher somente um dos equipamentos para atender à produção do produto.

O conectivo implicação, simbolizado por "⇒ " é outro conector que é referenciado neste trabalho. A implicação p⇒q é uma proposição que é falsa se somente se q for falsa. A sentença p é chamada de hipótese(premissa) e q de conclusão (conseqüência). A tabela A5 mostra a análise lógica da ação desse conector sobre as sentenças p e q.

Tabela A5: Análise lógica do operador "$\rightarrow$"

p	q	$p \rightarrow q$
V	V	V
V	F	F
F	V	V
F	F	V

O último conector lógico tratado neste trabalho é o bicondicional. Duas sentenças ligadas por este operador é verdadeira se somente se as duas sentenças possuem os mesmos valores, como é mostrado na tabela A6.

Tabela A6: Análise lógica do operador "$\Leftrightarrow$"

p	q	$p \leftrightarrow q$
V	V	V
V	F	F
F	V	F
F	F	V

Note que esse operador leva a proposição a um valor verdadeiro quando ambas as implicações $p \Rightarrow q$ e $q \Rightarrow p$ são verdadeiras.

- Equivalência de Proposições

Um importante passo usado nos problemas de lógica matemática é a substituição de uma afirmação por outra com o mesmo valor, ou seja, através da manipulação das proposições e operadores presentes na proposição composta(combinação de várias proposições). Para o estudo dessas equivalências são necessárias, a priori, algumas definições.

Definição 1: Uma proposição composta que é sempre verdadeira, não importando os valores dos termos presentes nela é chamada de "tautologia", enquanto que aquela que é sempre falsa é chamada de "contradição".

Para exemplificação de tautologias e contradições, considere uma proposição dada por p. A sentença composta "$p \vee \neg q$" e "$p \wedge \neg p$" é sempre verdadeira e falsa respectivamente, como mostrado na tabela A7.

Tabela A7: Análise lógica de proposições tautológicas e contradição

p	$\neg p$	$p \vee \neg p$	$p \wedge \neg p$
V	F	V	F
F	V	V	F

Quando as proposições compostas que têm o mesmo valor verdadeiro diz-se que elas são logicamente equivalente.

Definição 2: As proposições p e q são logicamente equivalente se $p \leftrightarrow q$ é uma tautologia. A notação $p \Leftrightarrow q$ denota que p e q são equivalentes.

Como exemplo de proposições equivalentes, considere as sentenças $\neg(p \vee q)$ e $(\neg p \wedge \neg q)$. A tabela A8 mostra a análise lógica entre as duas sentenças citadas.

Tabela A8: Análise de equivalência entre duas sentenças

p	q	$p \vee q$	$\neg(p \vee q)$	$\neg p$	$\neg q$	$\neg p \vee \neg q$
V	V	V	F	F	F	F
V	F	V	F	F	V	F
F	V	V	F	V	F	F
F	F	F	V	V	V	V

Note que na terceira e sexta colunas os valores das sentenças são os mesmos, indicando que as proposições citadas como exemplo são equivalentes.

A tabela mostra algumas propriedades de proposições e suas designações.

Tabela A9: Relações Equivalentes

Equivalência	Propriedade
$p \wedge V \Leftrightarrow p$ $p \vee F \Leftrightarrow F$	identidade
$p \vee V \Leftrightarrow V$ $p \wedge F \Leftrightarrow F$	dominação
$\neg(\neg p) \Leftrightarrow p$	negação
$p \vee q \Leftrightarrow q \vee p$ $p \wedge q \Leftrightarrow q \wedge p$	comutativa
$(p \vee q) \vee r \Leftrightarrow p \vee (q \vee r)$ $(p \wedge q) \wedge r \Leftrightarrow p \wedge (q \wedge r)$	associativa
$p \vee (q \wedge r) \Leftrightarrow (p \vee q) \wedge (p \vee r)$ $p \wedge (q \vee r) \Leftrightarrow (p \wedge q) \vee (p \wedge r)$	distributiva
$\neg(p \wedge q) \Leftrightarrow \neg p \vee \neg q$ $\neg(p \vee q) \Leftrightarrow \neg p \wedge \neg q$	Leis de De Morgan

Algumas equivalências importantes da teoria de lógica matemática são apresentadas na tabela A10.

Tabela A10: Proposições Equivalentes

$(p \vee \neg p) \Leftrightarrow V$
$(p \wedge \neg p) \Leftrightarrow F$
$(p \rightarrow q) \Leftrightarrow (\neg p \vee q)$

Apêndice B: Exemplo de um Arquivo no Formato MPS

```
NAME          GAMS/OSL
ROWS
 N  OBJECTRW
 E  R0000001
 E  R0000002
 E  R0000003
 E  R0000004
 L  R0000005
 L  R0000006
 L  R0000007
 L  R0000008
 L  R0000009
 L  R0000010
 L  R0000011
 L  R0000012
 L  R0000013
 L  R0000014
 E  R0000015
 E  R0000016
 E  R0000017
 E  R0000018
 E  R0000019
 E  R0000020
 E  R0000021
 E  R0000022
 E  R0000023
 E  R0000024
 E  R0000025
 E  R0000026
 E  R0000027
```

E R0000028
E R0000029
E R0000030
E R0000031
E R0000032
E R0000033
E R0000034
G R0000035
G R0000036

COLUMNS

C0000001	R0000001	1.000000	R0000005	1.000000
C0000001	R0000015	1.000000	R0000026	-1.000000
C0000002	R0000001	1.000000	R0000006	1.000000
C0000002	R0000017	1.000000	R0000027	-1.000000
C0000003	R0000001	1.000000	R0000007	1.000000
C0000003	R0000018	1.000000	R0000028	-1.000000
C0000004	R0000001	1.000000	R0000008	1.000000
C0000004	R0000019	1.000000	R0000029	-1.000000
C0000005	R0000001	1.000000	R0000009	1.000000
C0000005	R0000020	1.000000		
C0000006	R0000002	1.000000	R0000010	1.000000
C0000006	R0000025	1.000000	R0000035	-1.000000
C0000007	R0000002	1.000000	R0000011	1.000000
C0000007	R0000026	1.000000	R0000035	-2.000000
C0000008	R0000002	1.000000	R0000012	1.000000
C0000008	R0000027	1.000000	R0000035	-3.000000
C0000009	R0000002	1.000000	R0000013	1.000000
C0000009	R0000028	1.000000	R0000035	-4.000000
C0000010	R0000002	1.000000	R0000014	1.000000
C0000010	R0000029	1.000000	R0000035	-5.000000
C0000011	R0000003	1.000000	R0000005	1.000000
C0000011	R0000016	1.000000	R0000031	-1.000000

C0000012	R0000003	1.000000	R0000006	1.000000
C0000012	R0000021	1.000000	R0000032	-1.000000
C0000013	R0000003	1.000000	R0000007	1.000000
C0000013	R0000022	1.000000	R0000033	-1.000000
C0000014	R0000003	1.000000	R0000008	1.000000
C0000014	R0000023	1.000000	R0000034	-1.000000
C0000015	R0000003	1.000000	R0000009	1.000000
C0000015	R0000024	1.000000		
C0000016	R0000004	1.000000	R0000010	1.000000
C0000016	R0000030	1.000000	R0000036	-1.000000
C0000017	R0000004	1.000000	R0000011	1.000000
C0000017	R0000031	1.000000	R0000036	-2.000000
C0000018	R0000004	1.000000	R0000012	1.000000
C0000018	R0000032	1.000000	R0000036	-3.000000
C0000019	R0000004	1.000000	R0000013	1.000000
C0000019	R0000033	1.000000	R0000036	-4.000000
C0000020	R0000004	1.000000	R0000014	1.000000
C0000020	R0000034	1.000000	R0000036	-5.000000
C0000021	R0000015	1.000000	R0000017	-1.000000
C0000022	R0000017	1.000000	R0000018	-1.000000
C0000023	R0000018	1.000000	R0000019	-1.000000
C0000024	R0000019	1.000000	R0000020	-1.000000
C0000025	R0000020	1.000000		
C0000026	R0000025	1.000000	R0000026	-1.000000
C0000027	R0000026	1.000000	R0000027	-1.000000
C0000028	R0000027	1.000000	R0000028	-1.000000
C0000029	R0000028	1.000000	R0000029	-1.000000
C0000030	R0000029	1.000000		
C0000031	R0000016	1.000000	R0000021	-1.000000
C0000032	R0000021	1.000000	R0000022	-1.000000
C0000033	R0000022	1.000000	R0000023	-1.000000
C0000034	R0000023	1.000000	R0000024	-1.000000

```
    C0000035  R0000024    1.000000
    C0000036  R0000030    1.000000  R0000031   -1.000000
    C0000037  R0000031    1.000000  R0000032   -1.000000
    C0000038  R0000032    1.000000  R0000033   -1.000000
    C0000039  R0000033    1.000000  R0000034   -1.000000
    C0000040  R0000034    1.000000
    C0000041  OBJECTRW    1.000000  R0000035    1.000000
    C0000041  R0000036    1.000000
RHS
    RHS1      R0000001    1.000000  R0000002    1.000000
    RHS1      R0000003    1.000000  R0000004    1.000000
    RHS1      R0000005    1.000000  R0000006    1.000000
    RHS1      R0000007    1.000000  R0000008    1.000000
    RHS1      R0000009    1.000000  R0000010    1.000000
    RHS1      R0000011    1.000000  R0000012    1.000000
    RHS1      R0000013    1.000000  R0000014    1.000000
    RHS1      R0000015    1.000000  R0000016    1.000000
BOUNDS
 BV BOUND1    C0000001    1.000000
 BV BOUND1    C0000002    1.000000
 BV BOUND1    C0000003    1.000000
 BV BOUND1    C0000004    1.000000
 BV BOUND1    C0000005    1.000000
 BV BOUND1    C0000006    1.000000
 BV BOUND1    C0000007    1.000000
 BV BOUND1    C0000008    1.000000
 BV BOUND1    C0000009    1.000000
 BV BOUND1    C0000010    1.000000
 BV BOUND1    C0000011    1.000000
 BV BOUND1    C0000012    1.000000
 BV BOUND1    C0000013    1.000000
 BV BOUND1    C0000014    1.000000
```

```
BV BOUND1   C0000015    1.000000
BV BOUND1   C0000016    1.000000
BV BOUND1   C0000017    1.000000
BV BOUND1   C0000018    1.000000
BV BOUND1   C0000019    1.000000
BV BOUND1   C0000020    1.000000
FR BOUND1   C0000041    0.000000
ENDATA
```

Apêndice C: Programa Principal - Exmslv2.f

```
C*************************************************************************************
C              FACULDADE DE ENGENHARIA QUIMICA
C
C       DEPARTAMENTO DE ENGENHARIA DE SISTEMAS QUIMICOS
C
C                  PROGRAMA  EXMSLV2
C
C  AUTOR:  EDILSON DE JESUS SANTOS
C  ORIENTADORA: Dra. MARIA TERESA M. RODRIGUES
C
C  ESTE    PROGRAMA    SOLUCIONA  UM    PROBLEMA  DE  MINIMIZACAO
C  EM PLANTAS  FLEXIVEIS  UTILIZANDO  A  MODELAGEM  KONDILI  COM
C  ABORDAGEM  DE INFERENCIA LOGICA SOBRE RECURSOS COMPARTILHADOS.
C  TESTES  LOGICOS SAO  DESENVOLVIDOS  UTILIZANDO INFERENCIA DIRETA
C  DE  PARES  DE  OPERACOES  QUE  NAO  PODEM  SER  DENSENVOLVIDAS
C  SIMULTANEAMENTE NA  PLANTA  E TESTES ENVOLVENDO DISPONIBILIDADE
C  DE  RECURSO EXISTENTE  EM UM DETERMINADO  NIVEL DA PRODUCAO.
C
C  A ENTRADA DO MODELO E FEITA UTILIZANDO O FORMATO MPS GERADO
C  PELO GAMS.
C
C  DATA INICIO: 22/04/96
C  DATA DE TERMINO:
C*************************************************************************************
PROGRAM MAIN
C
C   IMPLICIT NONE
C
C  Inclui arquivos de controle de variaveis.
    INCLUDE (OSLI)
    INCLUDE (OSLR)
    INCLUDE (OSLN)
C
C  Definicao de variaveis e alocacao do dspace.
    INTEGER*2  N,M,H,NOP,NR,R,NE,IR,I,I2,I3,I4,J,INTEIRAS,MINTIJ
    INTEGER*2 COLINICIO,COLFIM,NPISONUMA1,INTERV,ESTADO
    INTEGER*4  MAXSPC,RTCOD,INDICA
```

```
      PARAMETER  (MAXSPC=15000000)
      REAL*4 TP,CR,OFR,NUMA1,CONSLOT,DIF1,NUMVAR
      REAL*8  DSPACE(MAXSPC)
      DIMENSION CONSLOT(10,100)
C
C  Definicao inteira(MSPACE) do Dspace.
      INTEGER*4 MSPACE(MAXSPC*2)
      EQUIVALENCE(DSPACE,MSPACE)
      COMMON/BIG/DSPACE
C
C  Comunicacao com a subrotina EKKEVNU
      COMMON/C1SUB1/TP(20,20),CR(20,20,20)
      COMMON/C2SUB1/N,M,H,NOP,NR,NE,INTEIRAS
      COMMON/C3SUB1/MINTIJ(50,4)
      COMMON/C4SUB1/ESTADO(50)
      COMMON/C5SUB1/OFR(20)

C  Comunicacao com ekkevnu e ekkbrnu
C  INDICA: indicador de primeira solucao inteira
      COMMON/C2S1/INDICA
C
      DATA RTCOD /0/
C*************************************************************************************
C  DOCUMENTACAO
C       N: Numero de tarefas
C       M: Numero de equipamentos na planta
C       H: Horizonte de Producao
C       NOP: Numero de operacoes
C       NR: Numero de recursos
C       NE: Numero de Estados
C       TP: Tempos de Processamento das operacoes
C       CR: Consumo de recursos das operacoes
C       CONSLOT: Consumo de recurso total nos slots
C       OF: Oferta de recursos
C       MINTIJ: Conjunto de operacoes envolvidas na programacao
C       INTERV: Intervalo correspondente a uma determinada operacao em MINTIJ
C       INTEIRAS: Numero de variaveis inteiras
C       NUMVAR: Numero de referencia no OSL de uma variavel binaria/
```

```
C*************************************************************************************************
C  Leitura dos tempos de processamento
      WRITE(*,*) 'LENDO DADOS DO PROBLEMA'
      OPEN(UNIT=22,FILE='tempos.dat',STATUS='OLD')
      READ(22,*) N,M,NOP,H,NR,NE,INTEIRAS
      DO I=1,N
           READ(22,*) (TP(I,J),J=1,M)
      END DO
      CLOSE(UNIT=22)
C
C  Leitura dos Consumos das Operacoes
      OPEN (UNIT=23,FILE='consumo.dat',STATUS='OLD')
      DO IR=1,NR
       READ(23,*) R
       DO I=1,N
           READ(23,*) (CR(R,I,J),J=1,M)
       ENDDO
      ENDDO
      CLOSE(UNIT=23)
C
C  Leitura da Matriz Relacional entre intervalos e operacoes
      OPEN(UNIT=24,FILE='mintij.dat',STATUS='OLD')
      DO I=1,NOP
           READ(24,*) (MINTIJ(I,J),J=1,4)
      ENDDO
      CLOSE(UNIT=24)
C
C  Leitura das ofertas
      OPEN(UNIT=25,FILE='ofr.dat',STATUS='OLD')
           READ(25,*) (ofr(J),J=1,NR)
      CLOSE(UNIT=25)
C
C  Leitura do Estados
      OPEN(UNIT=26,FILE='estado.dat',STATUS='OLD')
           READ(26,*) (estado(J),J=1,NE)
      CLOSE(UNIT=26)
C
      WRITE(*,*) 'INDICA=',INDICA
```

```
C
C   Descricao da Aplicacao(1 modelo).
      CALL EKKDSCA(RTCOD,DSPACE,MAXSPC,1)
       IF (RTCOD.GT.0) CALL CHKRT('EKKDSCA',RTCOD)
C
C   Determina que o modelo so tem 1 bloco.
      CALL EKKDSCM(RTCOD,DSPACE,1,1)
       IF (RTCOD.GT.0) CALL CHKRT('EKKDSCM',RTCOD)
C
C   Estabelece 5000 linhas para a variable Imaxrows.
      CALL EKKIGET(RTCOD,DSPACE,OSLI,OSLILN)
       IF (RTCOD.GT.0) CALL CHKRT('EKKIGET',RTCOD)
       IMAXROWS = -10000
      CALL EKKISET(RTCOD,DSPACE,OSLI,OSLILN)
       IF (RTCOD.GT.0) CALL CHKRT('EKKISET',RTCOD)
C
C    Estabelece se e um problema de minimizacao(rmaxmin=1)
C    ou se e um problema de maximizacao(rmaxmin= -1).
      CALL EKKRGET(RTCOD,DSPACE,OSLR,OSLRLN)
       IF (RTCOD.GT.0) CALL CHKRT('EKKRGET',RTCOD)
       RMAXMIN=1.0D0
      CALL EKKRSET(RTCOD,DSPACE,OSLR,OSLRLN)
       IF (RTCOD.GT.0) CALL CHKRT('EKKRSET',RTCOD)
C
C   Le o arquivo dados do arquivo MPS na unidade 98.
      CALL EKKMPS(RTCOD,DSPACE,98,1,9)
       IF (RTCOD.GT.0) CALL CHKRT('EKKMPS ',RTCOD)
C
C   Diminui os a dimensao dos valores envolvidos no problema
      CALL EKKSCAL(RTCOD,DSPACE)
C
C   Preprocessa o problema(eliminacao de linhas redundantes)
C   O quarto parametro especifica o tipo de reducao(default do GAMS 3)
C     CALL EKKPRSL(RTCOD,DSPACE,30,3)
C
C   Fornece uma base inicial
      CALL EKKCRSH(RTCOD,DSPACE,1)
C*************************************************************************************
```

```
C       CALL EKKRGET(RTCOD,DSPACE,OSLR,OSLRLN)
C        RPRINTCPU= 0.00001
C       CALL EKKRSET(RTCOD,DSPACE,OSLR,OSLRLN)
C*************************************************************************************
C  Preprocessamento da arvore  Branch-and-Bound Tree.  O
C  ultimo parametro determina o tipo de preprocessamento a
C  ser desenvolvido.
C    CALL EKKMPRE(RTCOD,DSPACE,1)
C     IF (RTCOD.GT.0) CALL CHKRT('EKKMPRE',RTCOD)
C*************************************************************************************
C  Esta secao e utilizada para problemas que estabelecam data de inicio
C  das operacoes.
C  Fixando variaveis que representam alocacao antes do tempo
C  de inicio previamente estabelecido para as operacoes.
C
        CALL EKKNGET(RTCOD,DSPACE,OSLN,OSLNLN)
        WRITE(*,*) 'fixando variaveis antes do EBT'
         DO I2=1,NOP
          IF (MINTIJ(I2,3).NE.0) THEN
           K2=MINTIJ(I2,3) -1
           COLFIM=H*(I2-1) + K2
           COLINICIO=H*(I2-1) + 1
           DO I3=COLINICIO,COLFIM
            IF (DSPACE(NCOLLOWER-I3+1).NE.1) THEN
              WRITE(*,*) 'fixando variavel', I3
              DSPACE(NCOLUPPER+I3-1)=0.0D0
            ENDIF
           ENDDO
          ENDIF
        ENDDO
C*************************************************************************************
C  Soluciona o Problema Linear  para o Node 0. O tipo  de algoritmo
C  utilizado  para  solucionar  o  PL , primal-simplex  (terceiro parametro).
C  O ultimo parametro permite que seja  usada  uma  base existente.
    CALL EKKSSLV(RTCOD,DSPACE,1,0)
C*************************************************************************************
C  Desativando Menssagens
C    CALL EKKMSET(IRTCOD,DSPACE,50,0,-1,0,0,50,1)
```

```
C     CALL EKKMSET(IRTCOD,DSPACE,87.0,-1.0,0.87,1)
C     CALL EKKMSET(IRTCOD,DSPACE,100.0,-1.0,0.100,1)
C     CALL EKKMSET(IRTCOD,DSPACE,57,0,-1,0.0.57.1)
C     CALL EKKMSET(IRTCOD,DSPACE,101,0,-1.0,0,101.1)
C     CALL EKKMSET(IRTCOD,DSPACE,102,0,-1,0,0,102,1)
C     CALL EKKMSET(IRTCOD,DSPACE,6,0,-1,0,0,6,1)
C
C       CALL EKKRGET(RTCOD,DSPACE,OSLR,OSLRLN)
C       RDEGSCALE= 0.0D0
C       CALL EKKRSET(RTCOD,DSPACE,OSLR,OSLRLN)
C
C      WRITE(*,*) 'RTARGET = ', RTARGET
C*************************************************************************************
C  Soluciona o problema MILP
        CALL EKKMSLV(RTCOD,DSPACE,1,35,50)
        IF (RTCOD.GT.0) CALL CHKRT('EKKMSLV',RTCOD)
C*************************************************************************************
        CALL EKKRGET(RTCOD,DSPACE,OSLR,OSLRLN)
        RPRINTCPU= 0.00001
        CALL EKKRSET(RTCOD,DSPACE,OSLR,OSLRLN)
        CALL EKKIGET(RTCOD,DSPACE,OSLI,OSLILN)
        CALL EKKRGET(RTCOD,DSPACE,OSLR,OSLRLN)
        CALL EKKNGET(RTCOD,DSPACE,OSLN,OSLNLN)
C*************************************************************************************
C  Impressao Direta do Dspace
        WRITE (6,75) ROBJVALUE
 75   FORMAT(/' VALOR DA FUNCAO OBJETIVO =  ',F15.4)
C
C     WRITE (6,100) (DSPACE(NROWACTS-1+I),
C    1   DSPACE(NROWNAMES-1+I),I = 1,INUMROWS)
C 100  FORMAT(/   ' VALORES DAS RESTRICOES  -  RESTRICOES',
C    1   / (T5,F8.2,T30,A8))
C
C     WRITE (6,200) (DSPACE(NCOLSOL-1+I),
C    1   DSPACE(NCOLNAMES-1+I),I = 1,INUMCOLS)
C 200   FORMAT(/   ' VALORES DAS VARIAVEIS  -  VARIAVEIS',
C    1     / (T5,F8.2,T30,A8))
C*************************************************************************************
```

```
C  A partir do Dspace, para as variaveis binarias com valores iguais a 1 e feita a
C  transformacao para i,j,k,  obtendo-se assim  as  operacoes alocadas. Impressao das
C  variaveis alocadas
          WRITE(*,*) ' '
          DO i2=1,NOP
C         WRITE(*,*) Mintij(i2,1)
C         WRITE(*,*) Mintij(i2,2)
          enddo
          DO I2=1,R
           DO I3=1,H
            CONSLOT(I2,I3)=0
           ENDDO
          ENDDO
          DO I2=1,INTEIRAS
           IF (DSPACE(NCOLSOL-1+I2).NE.0) THEN
            write(*,*) 'i2=',i2
            NUMVAR=I2
            NUMA1=NUMVAR/H
            NPISONUMA1=INT(NUMA1)
            DIF1= NUMA1 - NPISONUMA1
            IF (DIF1.GE.0.5.OR.DIF1.EQ.0) THEN
             INTERV=NINT(NUMA1)
             ELSE
              NUMA1= NUMA1 + 0.5
              INTERV=NINT(NUMA1)
            ENDIF
C
C   Determinacao de i e j
            DO I3=1,NOP
             IF (I3.EQ.INTERV) THEN
              I=MINTIJ(I3,1)
              J=MINTIJ(I3,2)
             ENDIF
            ENDDO
C
C   Determinacao do slot de alocacao
            k=NUMVAR - INT(NUMVAR/H)*H
            IF (K.EQ.0) K=H
```

```
            WRITE(*,*) 'Variaveis alocadas: Wijk=',i,j,k
C
C   Calculo do consumo total nos slots k
            DO I3=1,NR
             R=I3
             DO I4=K,(K+TP(I,J) - 1)
              CONSLOT(R,I4)=CONSLOT(R,I4) + CR(R,I,J)
             ENDDO
            ENDDO
           ENDIF
          ENDDO
C*****************************************************************************************
C           WRITE(*,*)
C*****************************************************************************************
C   Impressao das Bateladas Produzidas
C         DO I2=81,INTEIRAS + 80
C          IF (DSPACE(NCOLSOL-1+I2).NE.0) THEN
C        write(*,*) 'i2=',i2
C           NUMVAR=I2 - 80
C           NUMA1=NUMVAR/H
C           NPISONUMA1=INT(NUMA1)
C           DIF1= NUMA1 - NPISONUMA1
C           IF (DIF1.GE.0.5.OR.DIF1.EQ.0) THEN
C            INTERV=NINT(NUMA1)
C            ELSE
C             NUMA1= NUMA1 + 0.5
C             INTERV=NINT(NUMA1)
C           ENDIF
C
C   Determinacao de i e j
C           DO I3=1,NOP
C            IF (I3.EQ.INTERV) THEN
C             I=MINTIJ(I3,1)
C             J=MINTIJ(I3,2)
C            ENDIF
C           ENDDO
C
C   Determinacao do slot de alocacao
```

```
C          k=NUMVAR - INT(NUMVAR/H)*H
C          IF (K.EQ.0) K=H
C           WRITE(*,*) 'Bateladas: Bijk=',i,j,k,DSPACE(NCOLSOL-1+I2)
C         ENDIF
C        ENDDO
C*****************************************************************************************
        WRITE(*,*)
C*****************************************************************************************
C    Imprimindo consumo
          DO I3=1,NR
          WRITE(*,*)
          R=I3
          WRITE(*,*) 'RECURSO=',R
          WRITE(6,300) (CONSLOT(R,I4),I4=1,H)
          WRITE(*,*)
          ENDDO
 300  FORMAT(F8.2)
C
    STOP
    END
C
C*****************************************************************************************
C  ESTA SUBROTINA INDICA  SE HOUVE  ALGUM  ERRO DURANTE  A  EXECUCAO
C  DE ALGUMA DAS  SUBROTINAS  DO OSL(SE RTCOD FOR MAIOR QUE 0). A
C  VARIAVEL RTCOD E UM CODIGO  DE  RETORNO  DE  TODAS AS  SUBROTINAS
C  DO OSL QUE RECEBE UM VALOR DIFERENTE DE ZERO QUANDO OCORRE
C  ALGUM ERRO EM ALGUMA  DAS SUBROTINAS DO SISTEMA OSL.
C*****************************************************************************************
    SUBROUTINE CHKRT(RTNAME,RTCOD)
    CHARACTER*7 RTNAME
    INTEGER*4   RTCOD
C
    WRITE(6,9000) RTNAME,RTCOD
    IF (RTCOD.GE.200) STOP 16
    RETURN
9000  FORMAT (1X,'********** ',A7,' return code of ',I4,' **********')
    END
```

Apêndice D: Subrotina Ekkevnu

```
C*************************************************************************************
C            FACULDADE DE ENGENHARIA QUIMICA
C
C     DEPARTAMENTO DE ENGENHARIA DE SISTEMAS QUIMICOS
C
C     Autor: EDILSON DE JESUS SANTOS
C     Orientadora: Dra. MARIA TERESA M. RODRIGUES
C
C     Esta subrotina avalia o lower bound  do Node gerado utilizando
C     algoritmo primal-simplex.
C     A escolha do algoritmo simplex  dual  ou  primal pode ser feita
C     de acordo  as  caracteristicas  do modelo(terceiro parametro de
C     EKKSSLV).
C*************************************************************************************
      SUBROUTINE EKKEVNU(DSPACE,MSPACE,JRTCOD)
C
C     Inclui Arquivos de definicoes de variaveis
      INCLUDE (OSLI)
      INCLUDE (OSLR)
      INCLUDE (OSLN)
C
C     Definicoes de variaveis
      REAL*8   DSPACE(*)
      REAL*4   TP,CR,CRTOTAL,OFR,DIF1,NUMA1,NUMVAR,NUMVARAUX
      REAL*4   DISPR,OBJET
      INTEGER*2 MINTIJ,ALOC,CONSLOT,NUMIJ,COLINICIO,COLINICIO1
      INTEGER*2 COLFIM,I,J,NUMOSL,IND1,IND2,NPISONUMA1,BATELADA
      INTEGER*2 N,M,H,K,NR,R,S,NOP,INTERV,I2,I3,I4,I5,J4,K2,K4
      INTEGER*2 NE,ESTADO,INTEIRAS,IAUX,JAUX,KAUX,COLFIM1
      INTEGER*4 MSPACE(*),INDICA
      INTEGER*4 JRTCOD,IRTCOD,NUMNODE,DIRECAO
      DIMENSION NUMIJ(20,20),ALOC(50,100),CONSLOT(100)
C
C     Comunicacao com o  programa principal(exmslv2i.f)
      COMMON/C1SUB1/TP(20,20),CR(20,20,20)
```

```
      COMMON/C2SUB1/N,M,H,NOP,NR,NE,INTEIRAS
      COMMON/C3SUB1/MINTIJ(50,4)
      COMMON/C4SUB1/ESTADO(50)
      COMMON/C5SUB1/OFR(20)
C
C     Comunicacao com a subrotina EKKCHNU(SUB3)
      COMMON/C2SUB1SUB3/ NUMVAR,NUMNODE,DIRECAO
C
C     Comunicacao com EKKBRNU(SUB4)
      COMMON/C2SUB1SUB4/OBJET
C
C     Comunicacao com ekkbrnu
C     INDICA se uma solucao inteira foi encontrada
      COMMON/C2S1/INDICA
C*************************************************************************************
C     Inicializacao de variaveis
      DATA IRTCOD /0/
C*************************************************************************************
C     Documentacao: Variaveis
C       NUMNODE: Numero do Node escolhido
C       NUMVAR:  Numero da coluna da variavel
C       DIRECAO: Direcao de ramificacao
C       N: Numero de tarefas
C       M: Numero de equipamentos
C       NOP: Numero de Opercoes
C       NE: Numero de estados
C       NR: Numero de recursos
C       TP: Tempos de processamento
C       CR: Consumo de recurso  das operacoes
C       OFR: Oferta do recurso
C       CRTOTAL: Consumo de recurso de duas operacoes
C       INTEIRAS: Numero de variaveis binarias no modelo
C       ALOC: Operacoes anteriormente alocadas
C       CONSLOT: Consumo total de R1 no slot k
C       DISPR: Quantidade de recurso disponivel no slot k
C*************************************************************************************
C         WRITE(*,*) 'Entrando em EKKEVNU'
C         WRITE(*,*) 'INDICA=',INDICA
```

```
C       WRITE(*,*) 'Imprimindo o Node e variavel escolhidos'
C       WRITE(*,*) 'NUMNODE=', NUMNODE
C       WRITE(*,*) 'NUMVAR=' , NUMVAR
C       WRITE(*,*) 'DIRECAO=', DIRECAO
C
        IF (NUMNODE.EQ.0) THEN
        WRITE(*,*) 'Numero de tarefas=',N
        WRITE(*,*) 'Numero de equipamentos=',M
        WRITE(*,*) 'Horizonte de Producao=',H
        WRITE(*,*) 'Numero de estados=',NE
        WRITE(*,*) 'Numero de operacoes=',NOP
        WRITE(*,*) 'Numero de recursos=',NR
        WRITE(*,*) 'Numero de var. inteiras=',INTEIRAS
C*****************************************************************************
        WRITE(*,*)
C*****************************************************************************
C   Imprimindo os tempos de processamento
        WRITE(*,*) 'IMPRIMINDO TEMPOS DE PROCESSAMENTO'
        DO I2=1,N
         WRITE(*,*) (TP(I2,J2),J2=1,M)
        ENDDO
C*****************************************************************************
        WRITE(*,*)
C*****************************************************************************
C   Imprimindo os Consumos
       WRITE(*,*) 'IMPRIMINDO CONSUMOS'
       DO R=1,NR
        DO I2=1,N
         WRITE(*,*) (CR(R,I2,J2),J2=1,M)
        ENDDO
       ENDDO
C*****************************************************************************
        WRITE(*,*)
C*****************************************************************************
C   Imprimindo ofertas de recurso
        WRITE(*,*) 'OFERTA DE RECURSO'
        WRITE(*,*) (OFR(I2),I2=1,NR)
C*****************************************************************************
```

```
        WRITE(*,*)
C*************************************************************************************
C    Imprimindo Operacoes
        WRITE(*,*) 'IMPRIMINDO OPERACOES  I E J'
        DO I2=1,NOP
        WRITE(*,*) (MINTIJ(I2,J2),J2=1,4)
        ENDDO
C*************************************************************************************
        WRITE(*,*)
C*************************************************************************************
C    Imprimindo estados
        WRITE(*,*) 'IMPRIMINDO ESTADOS'
        WRITE(*,*) (ESTADO(I2),I2=1,NE)
    ENDIF
C*************************************************************************************
C     WRITE(*,*)
C*************************************************************************************
C    Eliminando as messagens 50 a 100)
        CALL EKKMSET(IRTCOD,DSPACE,50,0,-1,0,0,102,1)
C*************************************************************************************
C       WRITE(*,*) 'IMPRESSAO ANTES DE RESOLVER O PL'
C*************************************************************************************
C    Esta impressao permite verificar a reconstituicao da trajetoria
C    do Node escolhido atraves das variaveis ja fixadas.
C        0 - equivalente a estabelecer iprintfomask=63
C        1 - fornece estatistica
C        2 - fornece iteracao, funcao objetivo e status do problema
C        4 - nomes de linhas e colunas do problema
C        8 - informacao de cada variavel
C       16 - valores ativos
C       32 - valores reduzidos e duais
C       64 - limitante inferior
C      128 - limitante superior
C      256 - custos de entrada
C      512 - matriz de elementos do problema
C*************************************************************************************
C         CALL EKKIGET(IRTCOD,DSPACE,OSLI,OSLILN)
C         IPRINTFOMASK= 1+ 2 + 4 + 8 + 16 + 32 + 64 + 128 + 256
```

```
C         CALL EKKISET(IRTCOD,DSPACE,OSLI,OSLILN)
C         CALL EKKPRTS(IRTCOD,DSPACE)
C*****************************************************************************
C    Subrotinas de definicoes de variaveis de controle
      CALL EKKNGET(IRTCOD,DSPACE,OSLN,OSLNLN)
      CALL EKKIGET(IRTCOD,DSPACE,OSLI,OSLILN)
      CALL EKKRGET(IRTCOD,DSPACE,OSLR,OSLRLN)
C*****************************************************************************
C  Imprime valores das variaveis a partir do DSPACE
C       WRITE(6,50) (DSPACE(NCOLSOL-1+I2),
C    1   DSPACE(NCOLNAMES-1+I2),I2 = 1, INTEIRAS)
C  50   FORMAT(/ 'VALORES DAS VARIAVEIS - VARIAVEIS',
C    1     /(T5,F8.2,T30,A8))
C*****************************************************************************
C    Imprime status das Variaveis direto do MSPACE
C       CALL EKKNGET(IRTCOD,DSPACE,OSLN,OSLNLN)
C       WRITE(*,*) 'IMPRESSAO DO STATUS'
C       DO I2=1,INTEIRAS
C         WRITE(*,*) I2,MSPACE(NCOLSTAT-1+I2)
C       ENDDO
C*****************************************************************************
C   DIMINUICAO DO HORIZONTE DE TEMPO
C           IF (INDICA.EQ.1) THEN
C            WRITE(*,*) 'DIMINUINDO HORIZONTE DE TEMPO'
C            DO I3=1,NOP
C            WRITE(*,*) 'I3=',I3
C
C            NUMOSL= H*(I3-1) + (OBJET + 1)
C            COLINICIO=NUMOSL
C            COLFIM=H*(I3-1) + H
C
C            WRITE(*,*) 'COLINICIO E COLFIM',COLINICIO,COLFIM
C
C    Subrotinas de definicoes de variaveis de controle
C            CALL EKKNGET(IRTCOD,DSPACE,OSLN,OSLNLN)
C
C            DO I5=COLINICIO,COLFIM
C             IF (DSPACE(NCOLLOWER+I5-1).NE.1) THEN
```

```
C             IF (DSPACE(NCOLUPPER+I5-1).NE.0) THEN
C              WRITE(*,*) 'Variavel fixada=',I5
C              DSPACE(NCOLUPPER+I5-1)=0.0D0
C             ENDIF
C            ENDIF
C           ENDDO
C          ENDDO
C   Retorna o valor incial
C           INDICA=0
C          ENDIF
C*****************************************************************************************
C    PROCEDIMENTO PARA O CALCULO DE Wijk A PARTIR DO SEU
C    NUMERO DE REFERENCIA
C
       IF (DIRECAO.EQ.1) THEN
C    Calculo do teto de um numero fracionario
          NUMA1=NUMVAR/H
          NPISONUMA1=INT(NUMA1)
          DIF1= NUMA1 - NPISONUMA1
          IF (DIF1.GE.0.5.OR.DIF1.EQ.0) THEN
           INTERV=NINT(NUMA1)
           ELSE
            NUMA1= NUMA1 + 0.5
            INTERV=NINT(NUMA1)
          ENDIF
C          WRITE(*,*) 'NUMERO DO INTERVALO=',INTERV
C
C    Determinacao de i e j
          DO I2=1,NOP
           IF (I2.EQ.INTERV) THEN
            I=MINTIJ(I2,1)
            J=MINTIJ(I2,2)
           ENDIF
          ENDDO
C
C    Determinacao do slot de alocacao
          k=NUMVAR - INT(NUMVAR/H)*H
          IF (K.EQ.0) K=H
```

```
C
C          WRITE(*,*) 'variavel ramificada Wijk=',i,j,k
           IAUX=I
           JAUX= J
           KAUX=K
C*******************************************************************************
C          WRITE(*,*)
C*******************************************************************************
C  Para cada recurso desenvolver inferencia logica
         DO R=1,NR
C           DO I2=1,N
C            DO J2=1,M
C             NUMIJ(I2,J2)=0
C            ENDDO
C           ENDDO
C
           I=IAUX
           J=JAUX
           K=KAUX
C
C    Calculo das operacoes que nao podem ser executadas com Wijk
C           DO I3=1,NOP
C            I4=MINTIJ(I3,1)
C            J4=MINTIJ(I3,2)
C            IF (J4.NE.J) THEN
C             CRTOTAL= CR(R,I,J) + CR(R,I4,J4)
C             WRITE(*,*)  'I4=',I4,'J4=',J4,'CRTOTAL=',CRTOTAL
C             IF (CRTOTAL.GT.OFR(R)) THEN
C              NUMIJ(I4,J4)=H*(I3-1) + K
C              WRITE(*,*) NUMIJ(i4,j4)
C             ENDIF
C            ENDIF
C          ENDDO
C*******************************************************************************
C    PROCEDIMENTO DE INFERENCIA LOGICA  A1
C
C         WRITE(*,*) 'PROCEDIMENTO A1 DE INFERENCIA LOGICA '
C          DO I2=1,N
```

```
C          DO J2=1,M
C           IF (NUMIJ(I2,J2).NE.0) THEN
C            COLINICIO= NUMIJ(I2,J2)
C            COLFIM= COLINICIO + TP(I,J) - 1
C
C     Calculo do numero de referencia de W(i2,j2,H)
C             DO I5=1,NOP
C              IF (MINTIJ(I5,1).EQ.I2.AND.MINTIJ(I5,2).EQ.J2) THEN
C               COLFIM1=H*(I5-1) + H
C              ENDIF
C             ENDDO
C            WRITE(*,*) 'Variavel W(i2,j2,H)=', COLFIM1
C
C             IF  (COLFIM.GT.COLFIM1) THEN
C              COLFIM=COLFIM1
C             ENDIF
C
C     Inferencia logica
C            DO I3=COLINICIO,COLFIM
C             IF (DSPACE(NCOLLOWER + I3 -1).NE.1) THEN
C              IF (DSPACE(NCOLUPPER + I3 - 1).NE.0) THEN
C               WRITE(*,*) 'Variavel fixada=',I3
C               DSPACE(NCOLUPPER+I3-1)=0.0D0
C              ENDIF
C             ENDIF
C            ENDDO
C           ENDIF
C           ENDDO
C          ENDDO
C*****************************************************************************************
C         WRITE(*,*)
C*****************************************************************************************
C         WRITE(*,*) 'PROCEDIMENTO A2 DE INFERENCIA LOGICA'
C          DO I2=1,N
C          DO J2=1,M
C           IF (NUMIJ(I2,J2).NE.0) THEN
C            COLFIM= NUMIJ(I2,J2)
C            COLINICIO=COLFIM - TP(I2,J2) +1
```

```
C
C  Verificacao do numero de referencia da variavel w(i2,j2,1)
C            DO I5=1,NOP
C             IF (MINTIJ(I5,1).EQ.I2.AND.MINTIJ(I5,2).EQ.J2) THEN
C              COLINICIO1=H*(I5-1) + 1
C             ENDIF
C            ENDDO
C
C            IF (COLINICIO.LT.COLINICIO1) THEN
C             COLINICIO=COLINICIO1
C            ENDIF
C
C            WRITE(*,*) 'VARIAVEL DO SLOT 1=',COLINICIO1
C            DO I3=COLINICIO,COLFIM
C             IF (MSPACE(NCOLSTAT + I3 -1).NE.0) THEN
C              WRITE(*,*) 'Variavel fixada=',I3
C              DSPACE(NCOLUPPER+I3-1)=0.0D0
C             ENDIF
C            ENDDO
C           ENDIF
C          ENDDO
C         ENDDO
C*****************************************************************************************
C PROCEDIMENTO PARA INFERENCIA LOGICA  CONSIDERANDO A DISPONIBILIDADE
C  DE RECURSO
C
C          WRITE(*,*) 'PROCEDIMENTO POR FALTA DE RECURSO NO SLOT'
C
C  Armazenar variavel ramificada
       NUMVARAUX=NUMVAR
C
C  Inicializacao de variaveis
      DO I2=1,NOP
       DO J2=1,H
        ALOC(I2,J2)=0
       ENDDO
      ENDDO
C
```

```
      DO K2=1,100
       CONSLOT(K2)=0
      ENDDO
C
C      WRITE(*,*) 'PROCEDIMENTO PARA A VARIAVEL RAMIFICADA'
      ALOC(INTERV,1)=I
      ALOC(INTERV,2)=J
C
C       WRITE(*,*) 'PROCEDIMENTO PARA VARIAVEIS ALOCADAS'
C
     DO I3=1,INTEIRAS
      IF (DSPACE(NCOLLOWER + I3 - 1).EQ.1) THEN
        BATELADA=0
C
        NUMVAR=I3
        NUMA1=NUMVAR/H
        NPISONUMA1=INT(NUMA1)
        DIF1= NUMA1 - NPISONUMA1
        IF (DIF1.GE.0.5.OR.DIF1.EQ.0) THEN
         INTERV=NINT(NUMA1)
         ELSE
         NUMA1= NUMA1 + 0.5
         INTERV=NINT(NUMA1)
        ENDIF
C         WRITE(*,*) 'NUMERO DO INTERVALO=',INTERV
C
C   Determinacao de i e j
        DO I2=1,NOP
         IF (I2.EQ.INTERV) THEN
          I=MINTIJ(I2,1)
          J=MINTIJ(I2,2)
         ENDIF
        ENDDO
C
C   Determinacao do slot de alocacao
        k=NUMVAR - INT(NUMVAR/H)*H
        IF (K.EQ.0) K=H
C
```

```
C          WRITE(*,*) 'Wijk ja alocada=',i,j,k
C
C    Determinar se ja existe alguma batelada de ij alocada
           IND1=1
           DO I2=3,H+2
            IF (ALOC(INTERV,I2).EQ.0) THEN
             BATELADA=I2
             GOTO 55
            ENDIF
           ENDDO

 55         ALOC(INTERV,1)=I
           ALOC(INTERV,2)=J
           ALOC(INTERV,BATELADA)=NUMVAR
C
          DO I2=K,(K+TP(I,J)-1)
           CONSLOT(I2)=CONSLOT(I2)+CR(R,I,J)
          ENDDO
         ENDIF
        ENDDO
C*****************************************************************************
C         WRITE(*,*)
C*****************************************************************************
C         WRITE(*,*) 'IMPRESSAO DAS OPERACOES ALOCADAS'
C         DO I2=1,NOP
C          WRITE(*,*) (ALOC(I2,J2),J2=1,H+2)
C         ENDDO
C*****************************************************************************
C         WRITE(*,*)
C*****************************************************************************
C          WRITE(*,*) 'Impressao do consumo total nos slots'
C          WRITE(*,*) (CONSLOT(K),K=1,H+1)
C*****************************************************************************
C    FIXACAO DE VARIAVEIS PELA INDISPONIBILIDADE DE RECURSO
C
       DO K=1,H
        DISPR= OFR(R) - CONSLOT(K)
        DO I2=1, NOP
```

```
          I=MINTIJ(I2,1)
          J=MINTIJ(I2,2)
         IF (DISPR.LT.CR(R,I,J)) THEN
C             WRITE(*,*) 'SLOT K=',K,'CONSUMO DISPONIVEL=',DISPR
           NUMOSL= H*(I2-1) + K
C
          COLFIM= NUMOSL
          COLINICIO=NUMOSL - TP(I,J) + 1
C
          COLINICIO1=H*(I2-1) + 1
C
          IF (COLINICIO.LT.COLINICIO1) THEN
           COLINICIO=COLINICIO1
          ENDIF
C*****************************************************************************************
C     Verifica se  uma operacao ja alocada
           IND2=1
           DO I4=3,H+2
            IF (ALOC(I2,I4).EQ.NUMOSL) THEN
             IND2=0
            ENDIF
           ENDDO
C
C             WRITE(*,*) 'Wijk e NUMOSL',i,j,k,NUMOSL
           IF (IND2.EQ.1) THEN
            DO I5=COLINICIO,COLFIM
             IF (DSPACE(NCOLLOWER+I5-1).NE.1) THEN
              IF (DSPACE(NCOLUPPER+I5-1).NE.0) THEN
C                WRITE(*,*) 'Variavel fixada=',I5
               DSPACE(NCOLUPPER+I5-1)=0.0D0
              ENDIF
             ENDIF
            ENDDO
           ENDIF
          ENDIF
         ENDDO
        ENDDO
C para cada recurso
```

```
      ENDDO
C se direcaO for 1
      ENDIF
C*************************************************************************************
C    Resolve o subproblema linear
      CALL EKKSSLV(IRTCOD,DSPACE,1,0)
C*************************************************************************************
C    Armazenando funcao objetivo
      IF (INDICA.EQ.0) THEN
       CALL EKKRGET(IRTCOD,DSPACE,OSLR,OSLRLN)
       OBJET=ROBJVALUE
      ENDIF
C       WRITE(*,*) 'funcao objetivo=',OBJET
C*************************************************************************************
C       WRITE(*,*) 'IMPRESSAO DEPOIS DE RESOLVER O PL'
C
C    Impressao das bateladas maiores que 0(81:Primeira variavel Bijk)
C       DO I3=81,160
C        IF (DSPACE(NCOLSOL-1+I3).NE.0) THEN
C         IF (DSPACE(NCOLSOL-1+I3).GE.0) THEN
C          NUMVAR=I3 - 80
C          NUMA1=NUMVAR/H
C          NPISONUMA1=INT(NUMA1)
C          DIF1= NUMA1 - NPISONUMA1
C          IF (DIF1.GE.0.5.OR.DIF1.EQ.0) THEN
C           INTERV=NINT(NUMA1)
C           ELSE
C            NUMA1= NUMA1 + 0.5
C            INTERV=NINT(NUMA1)
C          ENDIF
C         WRITE(*,*) 'NUMERO DO INTERVALO=',INTERV
C
C    Determinacao de i e j
C          DO I2=1,NOP
C           IF (I2.EQ.INTERV) THEN
C            I=MINTIJ(I2,1)
C            J=MINTIJ(I2,2)
C           ENDIF
```

```
C          ENDDO
C
C    Determinacao do slot de alocacao
C          k=NUMVAR - INT(NUMVAR/H)*H
C          IF (K.EQ.0) K=H
C
C          WRITE(*,*) 'I3=',I3
C          WRITE(*,*) 'Bijk=',i,j,k,DSPACE(NCOLSOL-1+I3)
C         ENDIF
C        ENDIF
C       ENDDO
C       WRITE(*,*)
C*****************************************************************************************
C     Impressao das variaveis com valores maiores que 0
      DO R= 1,NR
       WRITE(*,*) 'RECURSO NUMERO=',R
       DO I3=1,INTEIRAS
        IF (DSPACE(NCOLSOL-1+I3).NE.0) THEN
         IF (DSPACE(NCOLSOL-1+I3).GE.0) THEN
          NUMVAR=I3
          NUMA1=NUMVAR/H
          NPISONUMA1=INT(NUMA1)
          DIF1= NUMA1 - NPISONUMA1
          IF (DIF1.GE.0.5.OR.DIF1.EQ.0) THEN
           INTERV=NINT(NUMA1)
           ELSE
            NUMA1= NUMA1 + 0.5
            INTERV=NINT(NUMA1)
          ENDIF
         WRITE(*,*) 'NUMERO DO INTERVALO=',INTERV
C
C    Determinacao de i e j
          DO I2=1,NOP
           IF (I2.EQ.INTERV) THEN
            I=MINTIJ(I2,1)
            J=MINTIJ(I2,2)
           ENDIF
          ENDDO
```

```
C
C    Determinacao do slot de alocacao
       k=NUMVAR - INT(NUMVAR/H)*H
       IF (K.EQ.0) K=H
        WRITE(*,*) 'W=',i,j,k,DSPACE(NCOLSOL-1+I3),'Rec=',CR(R,I,J)
       ENDIF
      ENDIF
     ENDDO
    ENDDO
C*****************************************************************************
      WRITE(*,*)
C*****************************************************************************
      WRITE(*,*) 'CONSUMO TOTAL SLOTS'
      WRITE(*,*) (CONSLOT(K),K=1,H+1)
C*****************************************************************************
C     CRITICALIDADE DE RECURSOS - MUDANCA NA ESCOLHA DA VARIAVEL
C     RAMIFICADA
C*****************************************************************************
C    Impressao dos estados(161:Primeira variavel Ssk)
C      DO I3=161,259
C       IF (DSPACE(NCOLSOL-1+I3).NE.0) THEN
C        IF (DSPACE(NCOLSOL-1+I3).GE.0) THEN
C         NUMVAR= I3 - 160
C         NUMA1=NUMVAR/(H+1)
C         NPISONUMA1=INT(NUMA1)
C         DIF1= NUMA1 - NPISONUMA1
C         IF (DIF1.GE.0.5.OR.DIF1.EQ.0) THEN
C          INTERV=NINT(NUMA1)
C          ELSE
C           NUMA1= NUMA1 + 0.5
C           INTERV=NINT(NUMA1)
C         ENDIF
C
C       WRITE(*,*) 'NUMERO DO INTERVALO=',INTERV
C
C    Determinacao do estado
C         DO I2=1,NE
C          IF (I2.EQ.INTERV) THEN
```

```
C           S=ESTADO(I2)
C           ENDIF
C          ENDDO
C
C    Determinacao do slot de alocacao
C           k=NUMVAR - INT(NUMVAR/(H+1))*(H+1)
C           IF (K.EQ.0) K=H + 1
C           WRITE(*,*) 'Ssk=',S,k,DSPACE(NCOLSOL-1+I3)
C         ENDIF
C        ENDIF
C       ENDDO
C*****************************************************************************
C       WRITE(*,*)
C*****************************************************************************
C    Esta impressao permite verificar a reconstituicao da trajetoria
C    do Node escolhido atraves das variaveis ja fixadas.
C    Altera informacao fornecida por EKKPRTS
C         CALL EKKIGET(IRTCOD,DSPACE,OSLI,OSLILN)
C         IPRINTFOMASK= 1 + 2 + 4 + 8 + 16 + 32 + 64 + 128 + 256
C         CALL EKKISET(IRTCOD,DSPACE,OSLI,OSLILN)
C         CALL EKKPRTS(IRTCOD,DSPACE)
C*****************************************************************************
C    Subrotinas de definicoes de variaveis de controle
      CALL EKKNGET(IRTCOD,DSPACE,OSLN,OSLNLN)
      CALL EKKIGET(IRTCOD,DSPACE,OSLI,OSLILN)
      CALL EKKRGET(IRTCOD,DSPACE,OSLR,OSLRLN)
C*****************************************************************************
C    Imprime valores de variaveis de controle
C       WRITE (*,*) 'Valor de Rtarget=  ' , Rtarget
C  60    FORMAT(/'VALOR DE Rtarget = ',F15.4)
C
C       WRITE (6,65) Rdegscale
C  65    FORMAT(/'VALOR DE Rdegscale= ',F15.4)
C
C       WRITE (*,*) Rbestsol
C  70    FORMAT(/'VALOR DE Rbestsol = ',E10.2)
C
C       WRITE (6,75) Riweight
```

```
C  75   FORMAT(/'VALOR DE Riweight = ',F15.4)
C
C      WRITE (6,80) Rbestest
C  80   FORMAT(/'VALOR DE Rbestest = ',F15.4)
C
C      WRITE (*,*) Rbestposs
C  85   FORMAT(/'VALOR DE Rbestposs = ',F15.4)
C
C      WRITE (*,*) 'FUNCAO OBJETIVO=',Robjvalue
C  90   FORMAT(/'VALOR DE Robjvalue = ',F15.4)
C
C      WRITE(*,*)
C
C      WRITE(*,*) 'VALOR de Rbbcutoff= ', Rbbcutoff
C*************************************************************************************
C   Imprime valores das variaveis a partir do DSPACE
C
C      WRITE(6,100) (DSPACE(NCOLSOL-1+I),
C   1  DSPACE(NCOLNAMES-1+I),I = 81, 160)
C 100   FORMAT(/ 'VALORES DAS VARIAVEIS - VARIAVEIS',
C   1     /(T5,F8.2,T30,A8))
C*************************************************************************************
C   Imprime o status das Variaveis direto do MSPACE
C
C      WRITE(*,*)'IMPRESSAO DO STATUS'
C      DO I2=1,80
C       WRITE(*,*) I2,DSPACE(NCOLSOL-1+I2)
C      ENDDO
C*************************************************************************************
C   Estabelece o retorno(JRTCOD=1) se a solucao nao for encontrada
      IF (IPROBSTAT.NE.0) THEN
       JRTCOD =1
C        WRITE(*,*) 'Node infactivel'
C        WRITE(*,*) 'IPROBSTAT=',IPROBSTAT
      ENDIF
C
C   Estabelecimento de Condicao de Parada
C   Valores possiveis de JRTCOD
```

```
C    JRTCOD = 0, Problema solucionado com sucesso
C            1, Problema e infactivel
C            2, Problema ilimitado
C            3, Maximo numero de iteracoes atingido
C            4, Nenhuma solucao encontrada
C            5, Maximo numero de solucao atingido
C*************************************************************************************
C    Calculo da condicao de parada
C    Se a diferenca entre duas solucoes inteiras e atingida pare
      ROPTCR=ABS((RBESTSOL-RBESTPOSS)/RBESTPOSS)
C
      IF (INDICA.EQ.1) THEN
       WRITE(*,*) 'ROPTCR = ', ROPTCR
       INDICA=0
      ENDIF
C
      IF (ROPTCR.LE.0.20) THEN
       JRTCOD=3
      ENDIF
C*************************************************************************************
      RETURN
      END
```

Apêndice E: Subrotina Ekkchnu

```
C*************************************************************************************
C            FACULDADE DE ENGENHARIA QUIMICA
C
C       DEPARTAMENTO DE ENGENHARIA DE SISTEMAS QUIMICOS
C
C  Autor: EDILSON DE JESUS SANTOS
C  Orientadora: Dra. MARIA TERESA M. RODRIGUES
C
C                  PROGRAMA EKKCHNU2
C
C  ESTA SUBROTINA EKKCHNU.F TEM O OBJETIVO DE SELECIONAR O PROXIMO
C  NODE A SER ANALISADO DURANTE A BUSCA BRANCH AND BOUND.
C
C  A ESCOLHA E FEITA UTILIZANDO A BUSCA INCORPORADA NO OSL.
C
C  obs: Este programa deve ser renomeado para EKKCHNU.f para que seja
C     usado pelo sistema OSL.
C*************************************************************************************
C  IREASN=1 : Subrotina é chamada antes de analisar a lista de nodes ativos.
C
C       =2 : Subrotina é chamada para cada node ativo
C
C       =3 : Subrotina é chamada depois de processar a lista  de nodes ativos
C
C  MARRAY(1): se IREASN=1, MARRAY(1) numero do node ativos;
C           o resto de MARRAY e DARRAY nao tem significados.
C
C           se IREASN=2, MARRAY(1) numero do node corrente.
C
C           se IREASN=3, MARRAY(1) numero do node escolhido;
C           o resto of MARRAY and DARRAY nao tem significados.
C
C  MARRAY(2): Numero de variaveis infactiveis.
C
C  DARRAY(2): valor da funcao objetivo do Node pai.
C*************************************************************************************
```

```
      SUBROUTINE EKKCHNU(DSPACE,MSPACE,IREASN,MARRAY,DARRAY,JRTCOD)
C
C     Inclui arquivos de definicao de variaveis tipo real.
      IMPLICIT NONE
      INCLUDE (OSLR)
C
C     Definicao de variaveis
      REAL*8    DSPACE(*),DARRAY(4),RXTARGET,DBEST,DVALOBJ,BESTOBJ
      REAL*4    NUMVAR,LISTVAR
      INTEGER*4 MSPACE(*),IREASN,MARRAY(6),JRTCOD,RTCOD,IBEST,CONT1,I
      INTEGER*4 NUMNODE,DIRECAO,NUMNODES,LISTNODES,LISTDIR
      DIMENSION LISTNODES(50000),LISTDIR(50000),LISTVAR(50000)
      SAVE      BESTOBJ,NUMNODES,CONT1,LISTNODES,LISTDIR,LISTVAR
C
C     Comunicacao com a subrotina EKKEVNU
      COMMON/C2SUB1SUB3/ NUMVAR,NUMNODE,DIRECAO
C
C       WRITE (*,*) 'Entrando em ekkchnu'
C
      GOTO (1000,2000,3000),IREASN
C
C     Incializacao das variaveis BESTOBJ e NUMNODE
C 1000 BESTOBJ=99999
 1000 NUMNODE=99999
      NUMVAR=9999
      DIRECAO=9999
      CONT1=0
C
C     Escreve o numero de nodes ativos
C       WRITE(*,*) 'Numero de nos ativo = ', marray(1)
      NUMNODES= MARRAY(1)
C
C     Inicializacao dos vetores LISTNODES,LISTDIR,LISTVAR
      DO I=1,NUMNODES
       LISTNODES(CONT1)=0
       LISTVAR(CONT1)=0
       LISTDIR(CONT1)=0
      END DO
```

```
C
C   Estabelece que esta subrotina seja chamada em cada node ativo.
    JRTCOD=0
    GOTO 4000
C
C   Fornece informacoes de cada Node ativo
C
C 2000  WRITE(*,*) 'analisando: Node numero  =', marray(1)
C     WRITE(*,*) 'numero de infactibilidades=',marray(2)
C     WRITE(*,*) 'depth do no = ', marray(3)
C     WRITE(*,*) 'tipo de ramificacao = ',marray(4)
C      WRITE(*,*) 'numero da coluna da variavel= ', marray(5)
C      WRITE(*,*) 'direcao da ramificacao= ', marray(6)
C     WRITE(*,*) ' degradacao estimada = ',darray(1)
C     WRITE(*,*) 'valor da solucao da no corrente= ',darray(2)
C     WRITE(*,*) 'valor da variavel continua no NODE pai= ', marray(3)
C     WRITE(*,*) 'somatorio de infactibilidades =', darray(4)
C
 2000  CONT1=CONT1+1
     LISTNODES(CONT1)=MARRAY(1)
     LISTVAR(CONT1)=MARRAY(5)
     LISTDIR(CONT1)=MARRAY(6)
     GOTO 4000
C
C   Estabelece o node que deve ser ramificado
C 3000  WRITE(*,*) 'Numero do No escolhido= ', marray(1)
 3000   CONTINUE
C   Pesquisa na LISTNODES para buscar a variavel e direcao de
C   ramificacao no Node escolhido(MARRAY(1)).
     DO I=1,NUMNODES
      IF (MARRAY(1).EQ.LISTNODES(I)) THEN
       NUMNODE=LISTNODES(I)
       NUMVAR= LISTVAR(I)
       DIRECAO=LISTDIR(I)
      ENDIF
     END DO
C
 4000 CONTINUE
```

```
C
      RETURN
      END
```

Apêndice F: Subrotina Ekkbrnu

```
C**********************************************************************************
C              FACULDADE DE ENGENHARIA QUIMICA
C
C         DEPARTAMENTO DE ENGENHARIA DE SISTEMAS QUIMICOS
C
C                  PROGRAMA EXBRNU
C
C  AUTOR: EDILSON DE JESUS SANTOS
C  ORIENTADORA: MARIA TERESA M. MOREIRA
C
C  Esta subrotina ‹ utilizada para modificar a escolha default de ramificacao do sistema OSL.
C  OBS: Utilizada somente quando for necesario para obter informacoes de variaveis.
C
C  IREASN situacao em que EKKBRNU ‹ chamada.
C
C  IREASN
C     =1: subrotina ‹ chamada antes que algum conjunto seja processado;
C         MARRAY e DARRAY nao sao usados.
C     =2: subrotina ‹ chamada antes do processamento de cada conjunto;
C         somente numero de conjunto, tipo do conjunto, prioridade, e
C         numero de variaveis nos conjuntos MARRAY e DARRAY.
C     =3: subrotina ‹ chamada depois do processamento de cada variavel
C         no conjunto; todas as informacoes de MARRAY e DARRAY sao validos.
C     =4: subrotina ‹ chamada depois do processamento de cada
C         conjunto; todas as informacoes de MARRAY e DARRAY sao
C         validos. Pode-se mudar o numero da variavel e prioridade
C         (Afeta a proxima ramificacao).
C
C     =5: subrotina ‹ chamada depois do processamento de todos os
C         conjuntos; todas as informacoes de MARRAY e DARRAY sao
C         v□lidas e variavel a ser ramificada pode ser mudada.
C
C  MARRAY(1): conjunto que esta sendo processado.
C  MARRAY(2): tipo de conjunto que esta sendo processado.
C  MARRAY(3): prioridade (1 mais alta).
```

```
C  MARRAY(4): numero de variaveis do conjunto
C  MARRAY(5): numero da coluna da variavel ramificada.
C  MARRAY(6): direcao de ramificacao.
C  DARRAY(1): valor corrente da variavel na solucao continua.
C  DARRAY(2): down pseudocost.
C  DARRAY(3): up pseudocost para variaveis simples.For SOS ‹ linha de referencia.
C  DARRAY(4): degradacao estimada para a ramificacao down.
C  DARRAY(5): degradacao estimada para a ramificacao up.
C*************************************************************************************
      SUBROUTINE EKKBRNU(DSPACE,MSPACE,IREASN,MARRAY,DARRAY,JRTCOD)
C
C     Inclui arquivos de definicoes de variaveis de controle.
C      IMPLICIT NONE
      INCLUDE (OSLI)
C
C     Definicoes de variaveis
      REAL*8   DSPACE(*),DARRAY(5)
      REAL*4   OBJET
      INTEGER*4 MSPACE(*),MARRAY(6),M2,IREASN,JRTCOD,RTCOD,ISEQ,
     +         ISET,IF9,INDICA
C     Comunicacao com a subrotina EKKEVNU(SUB1)
      COMMON/C2SUB1SUB4/OBJET
C
C     Comunicacao com exmslv2 e ekkevnu
      COMMON/C2S1/INDICA
C*************************************************************************************
C       Write(*,*) 'Entrando em EKKBRNU'
C
C      M2 = MARRAY(2)
C      IF((M2.EQ.1).OR.(M2.EQ.2).OR.(M2.EQ.3)) THEN
C       PRINT *,'THIS EKKBRNU NOT FOR SOS'
C       PRINT *,'STOPPING YOUR APPLICATION NOW'
C       STOP
C      ENDIF
C
      GOTO (100,200,300,400,500),IREASN
C
C     Inicializacao
```

```
 100   GOTO 600
C      CALL EKKIGET(RTCOD,DSPACE,OSLI,OSLILN)
C      WRITE (6,101) 'Existem ',INUMSETS,' conjuntos.'
C
C    Antes de cada conjunto
 200   CONTINUE
C      WRITE(*,*) 'numero do conjunto = ',MARRAY(1)
C      WRITE(*,*) 'Tipo do conjunto = ',MARRAY(2)
C      WRITE(*,*) 'Prioridade do conjunto = ',MARRAY(3)
C      WRITE(*,*) 'Num. de variaveis = ', MARRAY(4)
C
C    Fixa JRTCOD em 0, forcando a chamada EKKBRNU para cada variavel.
      JRTCOD=0
      GOTO 600
C
C   Em cada conjunto
C   Imprime informacoes de cada variavel
 300   GOTO 600
C      WRITE(*,*) 'Numero da coluna da variavel = ',marray(5)
C      WRITE(*,*) 'Direcao da ramificacao = ',marray(6)
C      WRITE(*,*) 'valor da variavel = ',darray(1)
C      WRITE(*,*) 'down pseudocost =  ',darray(2)
C      WRITE(*,*) 'up pseudocost = ', darray(3)
C      WRITE(*,*) 'degradacao estimada(dn) = ', darray(4)
C      WRITE(*,*) 'degradacao estimada(up) = ', darray(5)
C
C    Escolhe a variavel mais proxima de 1.0 ou de 0.5 if none above .9.
C      IF(DARRAY(1).GE.0.9D0) THEN
C      IF(DARRAY(1).GT.1.000005D0)  THEN
C        PRINT *,'Esta subrotina nao vale para variaveis SOS'
C        PRINT *,'Parando o programa'
C        STOP
C      ENDIF
C      IF(DARRAY(1).GT.DBEST2.AND.DARRAY(1).LE.0.99999D0) THEN
C        ISEQ=MARRAY(5)
C        DBEST2=DARRAY(1)
C        DVAL=DARRAY(1)
C             Branch up.
```

```
C        MARRAY(6)=1
C        ISET=MARRAY(1)
C       IF9=1
C       ENDIF
C     ELSEIF(ABS(DARRAY(1)-0.5D0).LT.DBEST.AND.IF9.EQ.0) THEN
C       ISEQ=MARRAY(5)
C       DBEST=ABS(DARRAY(1)-0.5D0)
C       DVAL=DARRAY(1)
C           Branch up.
C       MARRAY(6)=1
C       ISET=MARRAY(1)
C     ENDIF
      GOTO 600
C
C     No final do processamento de cada conjunto.
 400  CONTINUE
C       WRITE (*,*) 'O OSL escolhe ',MARRAY(5),MARRAY(6),DARRAY(1)
C
      IF (INDICA.EQ.0) THEN
       IF (MARRAY(5).EQ.0) THEN
C     Indica solucao Inteira
        INDICA=1
C         WRITE(*,*) 'EKKBRNU:FUNCAO OBJETIVO=',OBJET
       ENDIF
      ENDIF
C
C     WRITE (6,104) 'Eu escolhe: ',ISEQ,1,DVAL
C     IF(ISET.EQ.MARRAY(1)) THEN
C       MARRAY(5)=-ISEQ
C     ENDIF
      GOTO 600
C
C     No final de processamento dos conjuntos
 500  GOTO 600
C     MARRAY(1)=-ISET
 600  CONTINUE
      RETURN
 101  FORMAT(1X,A10,I5,A5)
```

```
103 FORMAT(1X,A11,I5,I5,F9.2)
104 FORMAT(1X,A8,I5,I5,F9.2)
  END
```

Apêndice G: Estrutura de Dados do Algoritmo A

- **arquivo 1: tempos.dat**

N = 2 M = 3 NOP=6 H = 25 NR =1 INTEIRAS=150

Tempos de processamento das operações

4.0 5.0 2.0

2.0 3.0 7.0

Onde:

N = Número de tarefas

M = Número de processadores

NOP = Número de operações

H = Horizonte de produção

NR = Número de recursos

- **arquivo 2: consumo.dat**

Recurso 1

Demanda do recurso 1

8.0 6.0 5.0

3.0 4.0 3.0

Demanda do recurso 2

7.0 4.0 1.0

2.0 5.0 2.0

- **arquivo 3: mintij.dat**

Tarefa	Processador	LBT	DD
A	1	-	-
A	2	-	-
A	3	-	-
B	1	-	-
B	2	-	-
B	3	-	-

onde:

LBT: Data início previamente estabelecido para a operação

DD: Data de entrega previamente estabelecida para a operação

- **arquivo 4: ofr.dat**

Quantidade oferecida para cada recurso

Recurso 1: 10

Recurso 2: 4

REFERÊNCIAS BIBLIOGRÁFICAS

- Aries, R. S. and Newton, R. D, Cost Estimation, McGraw-Hill, (1955).
- Applegate, D. e Cook, W., A Computational Study of the Job-Shop Scheduling Problem, ORSA Journal on Computing, 3 (2), pp. 149-156, 1991.
- Balas, E, Disjunctive Programming: Properties of the Convex Hull of Feasible Points. MSRR No. 348, Carnegie Mellon University, 1974.
- Baker, K. R., A Comparative Study of Flow-Shop Algorithms, 1973.
- Baker, K. R., Introduction to Sequencing and Scheduling, John Wiley, 1974.
- Beale, E., and Tomlin, J. A., Special Facilities in a Mathematical programming System for Nonconvex problems using Ordered Set of variables, in Proceedings of the Fifth International Conference on Operational Research, J. Lawrence, ed, Tavistock Publications, pp 447-454.
- Benders J. F., Partitioning procedures for solving mixed-variables programming problems. Numer. Math. 4, 238-252 (1962).
- Birewar, D. B. e Grossmann, I. E., Simultaneous Synthesis, Sizing, and Scheduling of Multiproduct Batch Plants, Ind. Eng. Chem. Res. pp 2242-2251, 1990.
- Blair C. E. et. al., Some results and Experiments in Programming techniques for Programming Logic. Comput. & Ops. Res. Vol. 13, No. 5, pp. 633-645, 1986.
- Brooke, A., Kendrick, D. e Meeraus, A., GAMS - A User's Guide, Release 2.25, The Scientific Press Series, Boyd & Fraser Publishing Company, Danvers, Massachusett, 1988.
- Coulson, J. M. e Richardson, J. F., Uma Introdução ao Projeto em Tecnologia Química, Tecnologia Química, Fundação Calouste Gubenkian, 1983.
- Dantzig, G.B., Linear Programming e Its Extensions, Princeton University Press, Princeton, N.J, 1963.
- Escudero, L. F., S3 Sets, An Extension of the Beale-Tomlin Special Ordered Sets, Mathematical Programming 42, pp 113-123, 1988.

- Egli, U. M. e Rippin, D. W. T., Short-Term Scheduling for Multiproduct Batch Chemical PlantsComputers & Chemical Engineering, Vol. 10, No. 4, pp. 303-325, 1986.
- Graves,S.C., A Review of Production Scheduling, Oper. Res., 29 (4), pp 646,1981.
- Grossmann, I. E. e Sargent, R. W. H., Optimum Design of Multipurpose Chemical Plants, Ind. Eng. Chem. Process Des. Dev., Vol. 18, No. 2, 1979.
- Grossmann, I.E., Quesada, I., Raman, R. e Voudouris, V.T., Mixed Integer Optimization Techniques for the Design e Scheduling of Batch Process, NATO ASI - Batch Process Systems Engineering, Antalya, Turkey, 1992.
- Hooker J. N., Logic-Based Methods for Optimization: A Tutorial.. ORSA Computer Science Techical Section Conference, Williamsburg, VA,USA, January 1994.
- Hooker, J. N., Logic-Based Methods for Optimization: A Tutorial, Presented at ORSA Computer Science Technical Section Meeting, Williamsburg, VA,USA., January, 1994.
- Hooker J. N. et. al., Logic Cuts for Processing Networks with Fixed Charges. Computers Ops. Res,Vol. 21, No. 3, pp. 265-279, 1994
- Kondili, E., Pantelides, C.C. e Sargent, W.H., A General Algorithm for Short-Term Scheduling of Batch Operations - I: MILP Formulation, Computers Chem. Eng., 17, pp. 211-227, 1993.
- Kondili, E., Shah, N. e Pantelides, C.C., Production Planning for the Rational Use of Energy in Multiproduct Continuous Plants, European Symposium on Computer Aided Process Enginnering, 1992.
- Ku, H., Rajagopalan, D. e Karimi, I.A., Scheduling in Batch Process, Chem. Eng. Prog., 83 (8), pp. 35-45, 1987.

- Ku, H. e Karimi, I.A., Scheduling in Serial Multiproduct Batch Processes with Finite Interstage Storage: A Mixed Integer Linear Program Formulation, Ind. Eng. Chem. Res., 27, pp. 1840-1848, 1988.
- Lomnicki, Z. A., A "Branch-and-Bound"Algorithm for the Exact Solution of the Three-machine Scheduling Problem, Operational Research Quartely, Vol. 16 No. 1, 1970.
- Mah, R. S. H., Chemical Process Structures and Information Flows, Butterworth Publishers, 1990.
- Ming, S. H. et al., Optimization with IBM OSL, 1994
- Ming, L. L. e Sahinids, N. V., Computational Trends and effects of Approximations in an MILP Model for process Planning, Ind. Eng. Chem. Res., 34, 1662-1673.
- Morton, T. e Pentico, D. W., Heuristic Scheduling Systems: with applications to production systems and project management, Wiley series in engineering & technology, 1993.
- Nemhauser, G.L. e Wolsey, L., Integer e Combinatorial Optimization, Wiley, N.W, 1988.
- Nilson, N. J., Problem-Solving Methods in Artificial Intelligence, McGraw-Hill, BackCompany 1971.
- OSL, A Guide and Reference (release 2), IBM-OSL, Kingston (NY), 816p, 1992.
- Pinto J. M. e Grossmann, I. E., Optimal Scheduling of Multstage Continuous Multiproduct Plants., I. Symposium: Integration in Planning and Operations-Houston, Texas-February 1993
- Pinto, J.M. e Grossmann, I.E., Resource Constrained Model for Short-Term Scheduling of Bacth Plants, Presented at CIMPRO (Conference on Computer Integrated Manufacturing in the Process Industries), Rutgers University, April 1994.

- Pinto, J.M. e Grossmann, I.E., A Logic Based Approach to Scheduling Problems with Resource Constraints, A Tutorial, April, 1995.

- Raman, R. e Grossmann, I. E., Relation Between MILP Modelling and Logical Inference for Chemical Process Syntheis., Computers chem. Engng., Vol. 15, No. 2, pp. 73-84, 1991.

- Raman, R., e Grossmann, I. E., Integration of Logic and Heuristic knowledge in MILP Optimization for Process Synthesis, Computers chem. Egng, Vol 16, No. 3, pp 155-171,1992.

- Raman R. e Grossmann I. E., Symbolic Integration of Logic in Mixed-Integer Linear Programming Techiniques for Process Synthesis. Computers chem. Engng, Vol. 17, No. 9, pp 909-927, 1993.

- Raman R. e Grossmann, I. E., Modelling e Computational Techniques for Logic based Integer programming. Computers chem. Engng, Vol. 18, No. 7, pp. 563-578, 1994.

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

- Reklaitis, G.V., Overview of Scheduling and Planning of Batch Process Operations, NATO Advanced Study Institute - Batch Process Systems Engineeering, Antalya, Turkey, 1992.

- Rich, S.H. e Prokopakis, G.J., Scheduling and Sequencing of Batch Operations in a Multipurpose Plant. Ind. Eng. Chem. Process. Des. Dev., 25, pp. 979-987, 1986.

- Rippin, D.W.T., Simulation of Single e Multiproduct Batch Chemical Plants for Design and Operations,Computers Chem. Eng., 17 (3), pp. 137-156, 1983.

- Rodrigues, M.T.M., Sequenciamente e Alocação de Operações em Flow-Shop na Indústria Química com Restrições sobre os Recursos Compartilhados: Uma Abordagem de Busca Orientada por Restrições, Tese de Doutorado, UNICAMP, Agosto,1992.

- Santos (1992), Análise de Tempos de Estabelecimento e Ordem de Manufatura no Sequenciamento de Tarefas em Processos Batelada, Dissertação de Mestrado, UNICAMP, Abril, 1994.
- Shah, N., Pantelides, C.C. e Sargent, W.H., A General Algorithm for Short- Term Scheduling of Batch Operations - II: Computational Issues, Computers Chem. Eng., 17, pp. 229-244, 1993.
- Sparrow, R. E., Forder, G. J., Rippin, D. W. T., The Choice of Equipment Sizes for Multiproduct Batch Plants: Heuristic vs Branch and Bound. Ind. Eng. Chem. Process des. Dev. 1975, 14, 197-203.
- Uskup, E., & Smith B. S., "A Branch and Bound Algorithm for Two-Stage Production- Sequencing Problems", Operations Research, 23 , No 1.1975.
- Woiler, S. e Washington, F. M., Projetos: Planejamento, Elaboração e Análise, Atlas, 1996.