UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA QUÍMICA

ÁREA DE CONCENTRAÇÃO:
Sistemas de Processos Químicos e Informática

"Estudo da Aplicação da Estratégia de *Simulated Annealing* aos Problemas de Programação da Produção em Unidades Batelada"

Autor: Fábio Ribeiro Conselheiro
Orientadora: Profª Drª Maria Teresa Moreira Rodrigues

Dissertação de Mestrado apresentada a Faculdade de Engenharia Química como parte dos requisitos exigidos para a obtenção do título de Mestre em Engenharia Química.

Campinas, 24 de Setembro de 1999

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

Conselheiro, Fábio Ribeiro
C765e Estudo da aplicação da estratégia de *simulated annealing* aos problemas de programação da produção em unidades batelada / Fábio Ribeiro Conselheiro.-- Campinas, SP: [s.n.], 1999.

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

1. Heurística. 2. Planejamento da produção. 3. Controle de produção. 4. Alocação de recursos. I. Rodrigues, Maria Teresa Moreira. II. Universidade Estadual de Campinas. Faculdade de Engenharia Química. III. Título.

Dissertação de Mestrado defendida e aprovada em 24 de Setembro de 1999 pela Banca Examinadora constituída pelos Professores Doutores:

Orientadora: Profª Drª Maria Teresa Moreira Rodrigues

Prof. Dr. Luis Gimeno Latre

Prof. Dr. Roger Josef Zemp

Este exemplar corresponde à redação final da Dissertação de Mestrado em Engenharia Química defendida por Fábio Ribeiro Conselheiro e aprovada pela Comissão Julgadora em 24/09/1999.

Rodrigues.

Profª Drª Maria Teresa Moreira Rodrigues

Agradecimentos

À Profª Drª Maria Teresa Moreira Rodrigues pela oportunidade de realizar este trabalho e por todo apoio oferecido;

Ao Prof Dr Luiz Gimeno Latre (FEE) por complementar o conhecimento necessário para a modelagem dos problemas;

Aos meus pais, Nelson Conselheiro e Maria do Carmo Ribeiro Conselheiro por terem começado isso tudo, investindo e apoiando. Obrigado pela confiança e por aceitarem a distância de casa;

Aos amigos de trabalho e de farra, Marcos Moreira, Wesley Alvarenga, Eliete Canal (um dia eu devolvo seu livro), Marco Fraga (obrigado pelos conselhos), aos dissidentes de Maringá por sempre arranjarem alguma "bebemoração", Antônio César Teixeira (obrigado pelas críticas) e outros tantos que passaram por aqui;

Aos meus irmãos Andréa e Rogério pelo incentivo e hospedagem na última fase deste trabalho e Marcos e Selma pelo apoio;

Ao CNPq e ao Serviço de Apoio ao Estudante por terem garantido financeira e juridicamente minha residência em Campinas;

E à Sandra, obrigado pela compreensão e apoio.

Resumo

O crescimento recente de indústrias de especialidades como farmacêutica, de alimentos e materiais para indústria eletrônica tem despertado o interesse na programação de produção de plantas de produção em batelada. O problema de Programação de Produção em tais plantas aumenta devido à necessidade de compartilhamento de recursos e tempo disponível para processamento de potencialmente um grande número de produtos, obedecendo à um conjunto de restrições, tais como prazos de entrega e oferta de materiais.

São propostos diversos métodos para a solução desses problemas, tais como Programação Matemática, *Branch and Bound* e métodos heurísticos. Devido à complexidade dos problemas de Programação, os métodos exatos sofrem restrição de uso devido ao tempo computacional que exigem, enquanto métodos heurísticos como o *Simulated Annealing* podem oferecer respostas sub-ótimas em tempo reduzido.

Neste trabalho foi estudada a utilização do método heurístico *Simulated Annealing* para a solução dos problemas de Programação de Produção em plantas Multiproduto e Multipropósito em batelada. São também apresentadas diferentes versões do algoritmo, baseados nos algoritmos de Metropolis e Glauber, além de métodos de recozimento e construção de novas soluções.

Abstract

Recent growth of specialty industries such as pharmaceutical, food products and material for electronic industries has increased the interest in the scheduling of batch production plants. The scheduling problem arises due the shared resources and available time processing of a potentially large number of products within certain constraints, like as due dates and available materials.

A several methods are proposed to solve this problem, such as Mathematics Programming, Branch and Bound and heuristics methods. Due the complexity of scheduling problems, the use of exacts methods are restricted due the large computational time demanded, while heuristics methods such as Simulated Annealing can offer sub optimal solutions in a short time.

In this work the application of the Simulated Annealing method for scheduling problems for Flowshops and Jobshops batch plants was studied. A several versions of the Simulated Annealing algorithm based on Metropolis's and Glauber's algorithm are presented, together with annealing schedules and types of random moves for new solutions.

ÍNDICE GERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

1. Introdução

Durante muitos anos as indústrias de processamento químico se preocuparam em desenvolver processos de produção contínuos, afim de atender a crescente demanda mundial pela química de base. Esse modo de produção exige muito estudo e conhecimento do processo para que o mesmo possa ser controlado, como conseqüência, novos materiais foram desenvolvidos.

Os novos produtos também possuíam processamento mais complexo, necessitando grande atenção do controle de qualidade, o que é facilitado pela produção em pequenas quantidades. Esse grupo de especialidades químicas, do qual fazem parte os fármacos, cosméticos, lubrificantes, bioquímica, entre outros; foi então classificado como Química Fina, que é caracterizada por produtos de alto valor agregado, demanda variável e similaridade de processamento de produtos diferentes.

Com isso, o modo de processamento em batelada passou a receber uma nova atenção, sobretudo devido à sua flexibilidade, podendo ser utilizada para a produção de diversos materiais similares através do compartilhamento de recursos, além de acomodar as variações de demanda com pequenos lotes de produção ou aumento do número de bateladas.

Estudos recentes indicam ainda que 50 % das indústrias são totalmente dependentes dos processos em batelada e destes, o interesse pela substituição por processos contínuos é de apenas 6 %, enquanto para outros 9 % a substituição não é tecnicamente viável (Parakrama (1985)).

Entretanto, essa flexibilidade obtida pelo compartilhamento de recursos como; matérias-primas, processadores, estocagem intermediária, energia, mão-de-obra e sobretudo tempo disponível para processamento; gera uma nova categoria de problemas, a definição do instante de tempo do início de processamento de cada batelada de cada produto diferente, afim de atender à algumas exigências, tais como:

- atender a demanda cumprindo prazos de entrega;
- satisfazer restrições sobre recursos compartilhados, incluindo equipamentos e utilidades;
- satisfazer restrições sobre balanço de massa para matérias-primas e intermediários;
- satisfazer restrições tecnológicas, como rotas de processamento.

Para que se acomode essas alterações do perfil de consumo, obtendo o melhor aproveitamento da capacidade instalada, é necessário um plano de Gerenciamento da Produção, que devido à complexidade, é geralmente dividido em dois níveis, o superior de Planejamento e o inferior de Programação de Produção.

O primeiro é responsável pela tomada de decisões a médio prazo, como determinação de quantidades a serem produzidas, compra de materiais, rotas de processamento e sobretudo, prazos de entrega. Já a Programação de Produção tem abrangência a curto prazo, sendo responsável pela alocação temporal de cada operação, cumprindo as restrições citadas anteriormente.

A determinação da seqüência de produção não é tarefa fácil, sendo que a maior parte dos problemas de relevância industrial são de grandes dimensões, dificultando ou mesmo frustrando a busca por soluções ótimas. O ótimo deve ser citado com cautela nessa área, sendo que adversidades como quebra de equipamentos, cancelamento ou aumento de pedidos não podem ser previstos, no máximo pode-se obter uma melhor acomodação de tais alterações.

Esses problemas são classificados como Não Polinomiais Completos (NP-Completos) cuja característica é o aumento exponencial do tempo de resolução com o aumento da dimensão do problema. Sendo também classificados como computacionalmente difíceis ou *hard problems*. Desta forma, a resolução de problemas de porte industrial só pode ser atendida por estratégias sub-ótimas, como heurísticas. Tamanha é a complexidade que é freqüente abolir o uso de estratégias de programação, efetuando o processamento de acordo com a chegada de pedidos, utilizando técnicas como Kanban que atende a demanda de acordo com o nível de estoque ou ainda de acordo com a intuição do responsável pela Programação de Produção. Abre-se mão de solução melhor frente à complexidade de resolução.

Neste trabalho é proposta a utilização da estratégia heurística *Simulated Annealing* para a resolução de problemas de Programação de Produção, abordando situações encontradas na indústria, como sequenciamento de tarefas que seguem a mesma rota de produção (Plantas Multiproduto) e a alocação de tarefas que compartilham os mesmos recursos com rotas diferentes (Plantas Multipropósito). Na literatura, a utilização deste método se restringe à minimização de trajeto para transportadoras e circuitos eletrônicos e para a indústria, o sequenciamento de operações em Plantas Multiproduto com poucas restrições.

O *Simulated Annealing*, é um método heurístico de busca em vizinhança, desenvolvido em 1953 por Metrópolis *et al.* para estudar o comportamento de substâncias cujas propriedades dependem da iteração de suas moléculas.

É caracterizado por fazer uma busca em vizinhança partindo de uma configuração inicial como referência, e à partir desta, promover modificações e avaliar a propriedade de interesse. As novas configurações vão sendo aceitas probabilisticamente como geradoras de novas configurações, através de uma função que considera a variação do valor da propriedade analisada e a temperatura a que a substância se encontra. Desta forma, o método pode aceitar como geradora de novas configurações uma outra que desvia o sentido da busca em relação ao extremo que se deseja, o resultado disso é a saída de mínimos locais através da transposição de pontos extremos para a possível obtenção do ótimo global.

O significado, próximo de Simulação de Recozimento, é devido à consideração dessa substância em um banho térmico com a avaliação de suas propriedades à medida que a temperatura é alterada. Como analogia, podemos considerar uma função de custo, e.g. tempo total de processamento, como sendo esta propriedade dependente da configuração das suas moléculas ou neste caso, a seqüência das operações. A temperatura do banho representa apenas um parâmetro de controle, utilizado na ponderação para aceitação de uma nova seqüência como geradora de outras.

Em geral, os métodos exatos são inviabilizados pelo tempo que demandam para concluir a busca, ao contrário de alguns métodos heurísticos que abrindo mão da otimalidade oferecem resultados em tempo hábil. Entretanto o seu uso é restrito à situações em que seu resultado só pode ser considerado bom frente ao fato de não se ter nenhuma solução, quando há limitação de tempo ou quando os dados para a resolução do problema são pouco confiáveis; pode ser suficiente a apresentação de uma solução melhor do que a atual.

Uma grande vantagem apresentada pelas heurísticas sobre as técnicas que buscam soluções exatas é que em geral permitem uma maior flexibilidade para o manejo das características do problema e geralmente oferecem mais de uma solução, permitindo ampliar as possibilidades de quem decide, sobretudo quando existem fatores não quantificáveis que não puderam ser adicionados ao modelo, mas que também devem ser considerados.

O *Simulated Annealing* foi proposto inicialmente para tratar de problemas de otimização sem restrições. Neste trabalho é proposta a sua utilização para solução de problemas de programação de produção na presença de diversas restrições tais como; precedência tecnológica, restrições de balanço de massa e ainda, prazos de entrega.

Para tratar de problemas com essas características, é proposta neste trabalho, uma estratégia de solução via SA usando uma abordagem de duas fases:

1) uma pré-qualificação de seqüências (soluções) candidatas com respeito à factibilidade de balanço de massa;
2) alocação temporal das operações, calculando a função de custo e construindo a Carta de Gantt.

Espera-se com isso, viabilizar a utilização *Simulated Annealing* não como uma estratégia isolada para resolução de problemas deste tipo, mas promover sua utilização em conjunto com outras estratégias de otimização que em geral partem sem referências do espaço de busca e demandam muito tempo, inviabilizando seu uso.

2. Revisão da Literatura

2.1 Introdução

O presente capítulo busca de maneira sucinta introduzir o problema de programação de produção de plantas com regime de produção descontínuo, mostrando suas principais características e complexidade de solução.

Na seqüência, são apresentadas características e aspectos positivos e negativos das técnicas atuais para resolução do problema, justificando a escolha do método heurístico *Simulated Annealing* para a resolução dos problemas deste tipo.

Os detalhes e considerações sobre este método heurístico serão tratados nos capítulos seguintes.

2.2 Características do Problema de Programação de Produção

Desde o seu surgimento, o regime de produção industrial vem passando por diversas alterações. Até os anos 30 os processos não eram bem conhecidos, e a interferência humana era facilitada no modo de produção em batelada, que passou a dar lugar à produção contínua, de maior eficiência e produtividade, com o aumento da demanda mundial.

Entretanto, o processamento descontínuo é economicamente interessante e vem diferenciando alguns setores, caracterizados pela produção de uma grande quantidade de itens com a mesma rota de produção, ou itens de alto valor, porém sujeitos à problemas operacionais, como limpeza constante de equipamentos e ainda à demandas sazonais e influenciadas pela situação do mercado. Em geral esses produtos são os de química fina, como fármacos, lubrificantes, corantes, polímeros, cosméticos, bioquímicos, herbicidas; além de aço, gêneros alimentícios entre outros.

Em termos de taxas de produção, plantas de processamento contínuo são adotadas para capacidades acima de 5.000 ton./ano, enquanto as descontínuas são desejáveis para capacidades inferiores a 500 ton./ano (Douglas (1988)). Em virtude dessa baixa escala produtiva, torna-se economicamente inviável o estabelecimento de plantas dedicadas exclusivamente a cada produto desejado, recorrendo-se então, às plantas com alto grau de flexibilidade, que possam absorver um grande número de produtos similares, estabelecendo alternativas e variantes de produção.

Estudos indicam ainda, que 50 % das industrias são totalmente dependentes dos processos em batelada, e destes, o interesse pela substituição por processos contínuos é para apenas 6 %, enquanto para outros 9 % a substituição não é tecnicamente viável (Parakrama (1985)).

A Programação de Produção em industrias de processamento é caracterizada pela realização de um número de projetos ou tarefas, que são compostos por partes elementares chamadas de atividades ou operações. Cada operação requer certas quantidades de um recurso específico por um certo período de tempo chamado de tempo de processamento. Recursos também são constituídos por partes elementares, como transporte, mão-de-obra, processadores, matéria-prima, energia entre outros.

Problemas de Programação de Produção são freqüentemente complicados devido ao grande número de restrições relacionando cada operação às demais, recursos às operações e cada recurso ou operação à eventos externos ao sistema, como alterações de demanda.

Para acomodar as variações do perfil de produção, é necessário um Plano de Gerenciamento flexível, que inclui o Planejamento e a Programação de Produção, que não possui solução simples, pois considera simultaneamente diversos fatores.

Quando a demanda é conhecida ou previsível por um longo período de tempo, o gerenciamento da produção é bastante facilitado, dispondo de certa liberdade na alocação das operações e a planta pode ser operada no regime de campanha. Durante este período, os recursos são dedicados à produção de um subconjunto de produtos, estabelecendo um perfil de produção chamado "cíclico". Entretanto, se as previsões de demanda não são confiáveis, a produção da planta será definida por pedidos a curto prazo, resultando um horizonte de planejamento restrito, cuja alteração de programação para atender a cada novo pedido origina o problema de programação de curto prazo. O mesmo é provocado por quebra de equipamentos ou atraso na chegada de matéria-prima, exigindo decisões rápidas, muitas vezes abrindo mão da maior capacidade produtiva.

A literatura cita dois tipos de abordagens para a resolução do problema de gerenciamento de produção: a abordagem nível único e a abordagem hierárquica.

A abordagem em nível único considera simultaneamente os aspectos envolvidos no planejamento e na programação da produção. O objetivo é a análise e determinação da demanda de produtos, os recursos requeridos, os equipamentos destinados à execução de cada operação, a seqüência de produção em cada período de tempo e a política de estocagem intermediária utilizada, a fim de otimizar um critério de desempenho.

Na abordagem em nível hierárquico, o problema global é subdivido em pelo menos dois subníveis, o Planejamento (a longo prazo) e a Programação de Produção (a curto prazo).

No nível superior de Planejamento, são tomadas as decisões táticas como a expansão de utilidades, expansão de capacidade instalada, decisão das rotas de produção, nível de atividades, consumo de recursos, tipo de armazenagem intermediária e perfis temporais de oferta de recursos. São ainda definidos os intervalos de tempo para execução das operações e número de bateladas de cada operação para atender as demandas, contudo, não são tomadas decisões de sequenciamento de operações, nem são necessárias informações temporais detalhadas do plano de produção.

A Programação de Produção é definida como a alocação temporal de recursos, sujeita a restrições e com o objetivo de otimizar um critério de desempenho durante a execução de uma série de tarefas. O problema de programação de produção é também chamado de agendamento de produção ou *Scheduling*, e tem natureza combinatorial, pois deseja-se alocar M processadores disponíveis para a realização de N conjuntos de operações que constituem uma tarefa. Dependendo da natureza do problema, podem existir $(N!)^M$ seqüências possíveis.

Devido a esta grande complexidade e dimensão, são propostas diversas ferramentas computacionais para o planejamento e programação da produção, porém mesmo para os computadores atuais, não é tarefa fácil encontrar uma solução ótima ou mesmo factível em tempo hábil, visto que o tempo de solução cresce exponencialmente com o crescimento do tamanho do problema.

Por motivo destas dificuldades, os algoritmos propostos são orientados para resolver problemas com configurações específicas. A maior dificuldade que se encontra para o desenvolvimento de um algoritmo geral é a dimensão final do problema, que impede sua solução em um tempo "razoável", neste sentido, são comuns abordagens que, orientadas pela estrutura do problema, busquem soluções "boas" em lugar de soluções ótimas.

2.3 Aspectos Relevantes para o Problema de Programação a Curto Prazo

A seguir são brevemente apresentados alguns aspectos usados para a caracterização e tratamento de problemas de programação de produção.

2.3.1 Estruturas de Processamento

As plantas descontínuas podem ser classificadas quanto à sua estrutura de processamento como Multiproduto e Multipropósito.

2.3.1.1 Plantas Multiproduto (*Flowshop*)

Uma planta multiproduto é caracterizada pelo processamento de N tarefas em M processadores, sendo que as operações que compõem cada tarefa seguem através dos mesmos processadores e na mesma ordem. Sua concepção básica é o fluxo unidirecional, sendo indicado para o processamento de produtos similares.

Um caso particular e muito importante na indústria é o *flowshop* com seqüências de permutação, para o qual a seqüência de operações é a mesma para todos os processadores, resumindo o problema a N! seqüências possíveis.

No caso mais geral não existe a imposição de que a seqüência de operações seja a mesma em cada processador, isso é possível quando alguma tarefa não precisa passar por todos os processadores ou seja, o seu tempo de processamento é nulo nesse processador. Dessa forma podem existir N! seqüências diferentes em cada processador, elevando o problema à $N!^{M}$ seqüências possíveis e o problema é chamado de *flowshop* generalizado.

No *flowshop* generalizado, um processador "p" pode interagir com outro que não seja o anterior "p-1" ou o próximo "p+1". De acordo com a seqüência de tarefas em cada processador, é possível se estabelecer duas formas de interações, o desvio (*bypass*) e o cruzamento (*crossing*) de tarefas. Isso ocorre principalmente quando os tempos de limpeza e preparação entre as operações não são os mesmos ou quando as tarefas possuem o mesmo fluxo na planta mas, não possuem a mesma rota de produção, ou seja, não utilizam os mesmos processadores, podendo existir mais processadores do que etapas para a realização de um produto final.

A solução dos problemas de Programação de Produção é geralmente representada graficamente através de um diagrama chamado Carta de Gantt. Este quadro é constituído por uma escala horizontal cujas subdivisões representam o menor período de tempo considerado (*slots*), os processadores são representados por linhas horizontais que são subdivididas por essa escala. Quando uma operação requisita e ocupa um processador, as subdivisões que correspondem a esse intervalo de tempo são codificadas com cores ou números que representem esta operação. Outras operações só poderão ocupar o mesmo processador em intervalos de tempo disponíveis, desta forma, temos a visualização gráfica da solução, indicando habilitação, início e fim de operação nos processadores.

Na figura 2.1 é mostrado um exemplo de estrutura de processamento tipo *flowshop* cuja principal característica é o fluxo unidirecional, os processadores são utilizados na mesma ordem para produzir produtos diferentes. Na figura 2.2 são mostrados exemplos de soluções temporais (cartas de Gantt), para situações em que podem ocorrer *crossing* ou *by pass*.

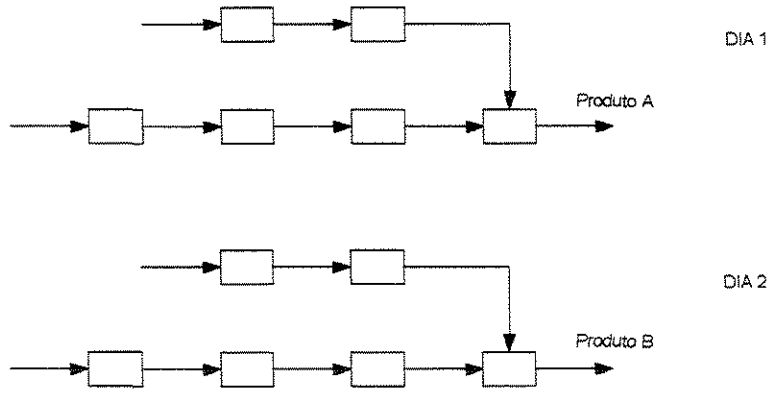


FIGURA 2.1: PLANTA MULTIPRODUTO (FLOWSHOP)

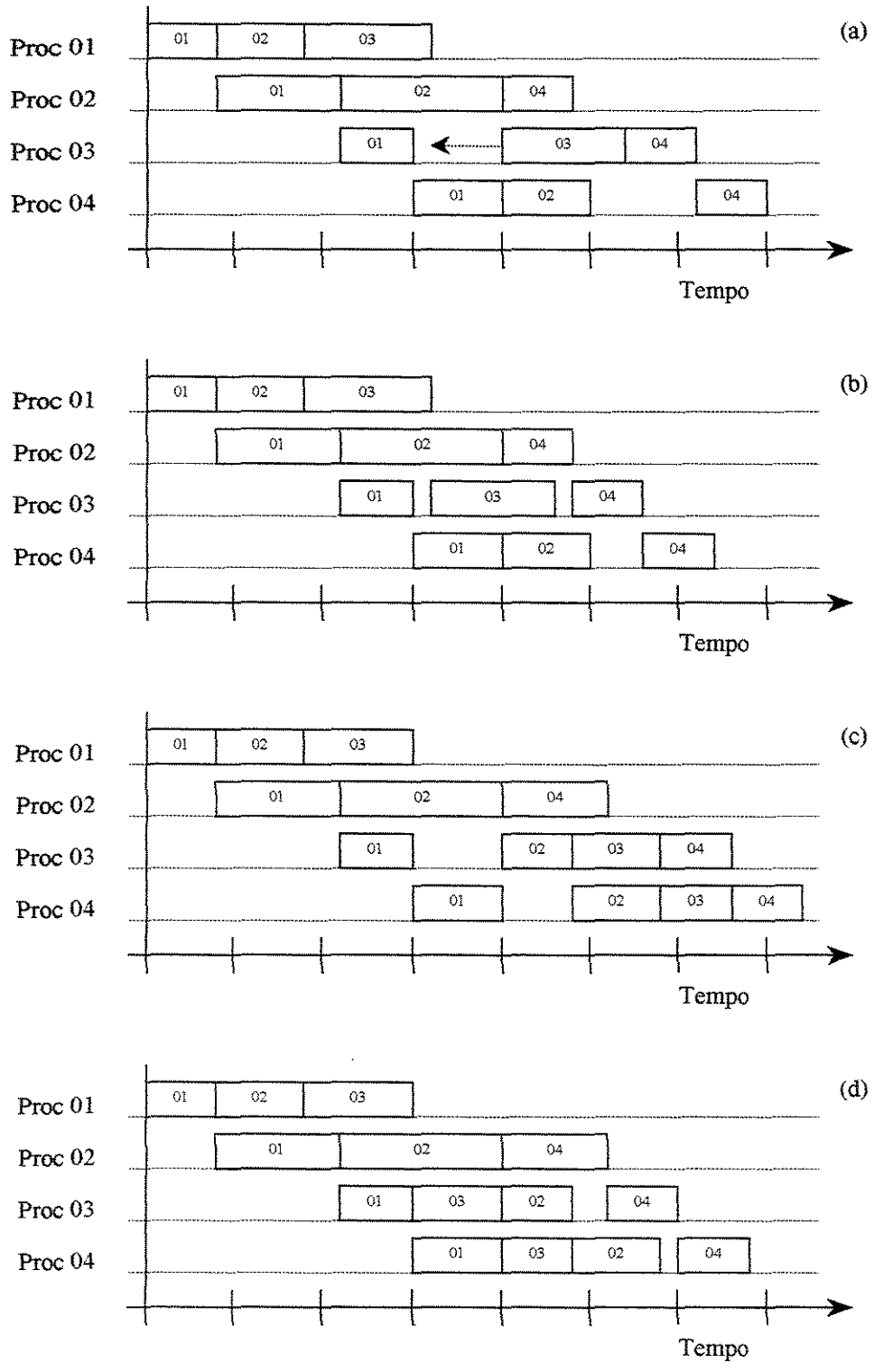


FIGURA 2.2: INTERAÇÕES ENTRE OPERAÇÕES EM UMA PLANTA MULTIPRODUTO.

(a) Seqüência de permutação sem desvio (*bypass*).
(b) Seqüência de permutação com desvio (*bypass*).
(c) Seqüência de permutação, *flowshop* sem cruzamento (*crossing*).
(d) *Flowshop* generalizado, com cruzamento (*crossing*).

2.3.1.2 Plantas Multipropósito (*Jobshop*)

Em uma planta multipropósito as N tarefas seguem rotas diferentes umas das outras através dos M processadores, não há o fluxo unidirecional como no *flowshop*. Para o estabelecimento das diferentes rotas de processamento com o transporte de material, as plantas multipropósito exigem complexas interconexões físicas entre os processadores, o que torna o seu uso pouco freqüente na prática.

A figura 2.3 ilustra uma planta multipropósito habilitada a produzir três produtos diferentes, A, B e C. Estes produtos, embora utilizem os mesmos processadores, o fazem seguindo rotas diferentes.

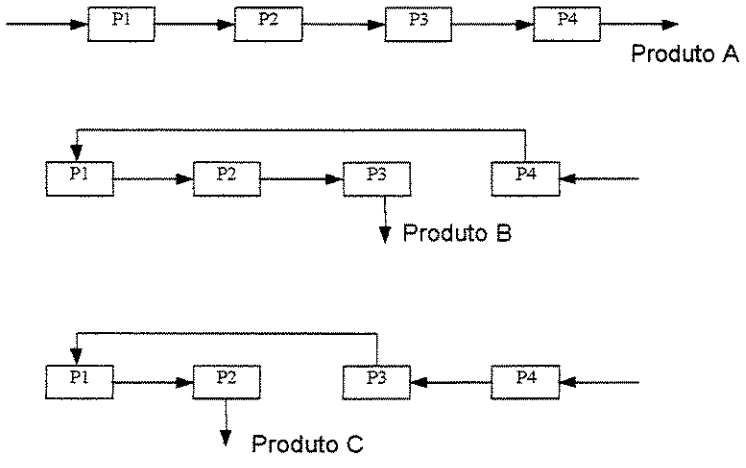


FIGURA 2.3 PLANTA MULTIPROPÓSITO (*JOBSHOP*)

A figura 2.4 representa a Carta de Gantt para a planta multipropósito acima, envolvendo quatro processadores e três produtos A, B e C. A ordem de processamento é indicada pelo número à direita da letra.

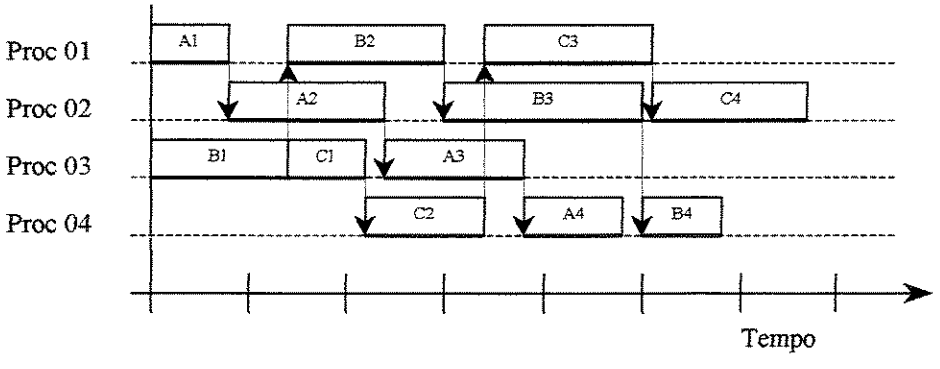


FIGURA 2.4 CARTA DE GANTT DE PLANTA MULTIPROPÓSITO

2.3.2 Armazenagem Intermediária

Em uma indústria química, o produto resultante de uma operação pode ser um intermediário ou precursor para o processamento de outras operações. Dessa forma, o processamento de uma operação esta relacionado á disponibilidade de intermediários e precursores em quantidade suficiente para a execução da mesma, o que nos leva à consideração de políticas de armazenagem intermediária.

O agendamento de produção de uma planta batelada é significativamente afetado pelo tipo de estocagem intermediária adotado. A estocagem intermediária ilimitada UIS (*Unlimited Intermediate Storage*) é provavelmente a mais simples para se trabalhar e também a mais estudada. Para essa política é considerada a transferência de um produto intermediário para tanques de estocagem tão logo o processamento seja completado, não havendo limite para o número de produtos ou quantidade de material transferido, existindo armazenagem suficiente para atender a qualquer demanda.

Armazenagem intermediária finita FIS (*Finite Intermediate Storage*), nesta política, os tanques estão disponíveis em quantidade limitada podendo ser compartilhados ou não entre os produtos intermediários. Caso o tanque de estocagem não esteja disponível, o material não pode ser transferido, devendo aguardar dentro do processador ou a produção deverá ser sincronizada.

A armazenagem dedicada emprega tanques de estocagem destinados a apenas um intermediário, não são portanto compartilhados entre produtos diferentes.

Caso não haja tanques de estocagem e o intermediário seja estável, isto é, não altera suas propriedades com o tempo, o mesmo pode ser estocado no próprio processador até que o próximo processador esteja livre. Esta política de não estocagem intermediária NIS (*Non Intermediate Storage*) é bastante empregada, porém acarreta a indisponibilidade do processador durante o uso como estoque, o que deve ser economicamente considerado.

Outra situação ocorre quando o intermediário é instável, bastante comum em indústrias siderúrgicas e processos biológicos. O material só pode sair de um processador quando o próximo puder recebê-lo. O intermediário, portanto não pode esperar entre dois processamentos, definindo a política ZW (*Zero Wait*).

Muitas vezes essas políticas são aplicadas juntamente em diversas partes da planta, definindo uma política de estocagem intermediária mista MIS (*Mixed Intermediate Storage*). A figura 2.5 representa o efeito da armazenagem intermediária sobre o tempo de conclusão das tarefas em uma planta multiproduto. A área hachurada representa o material sendo estocado no próprio processador.

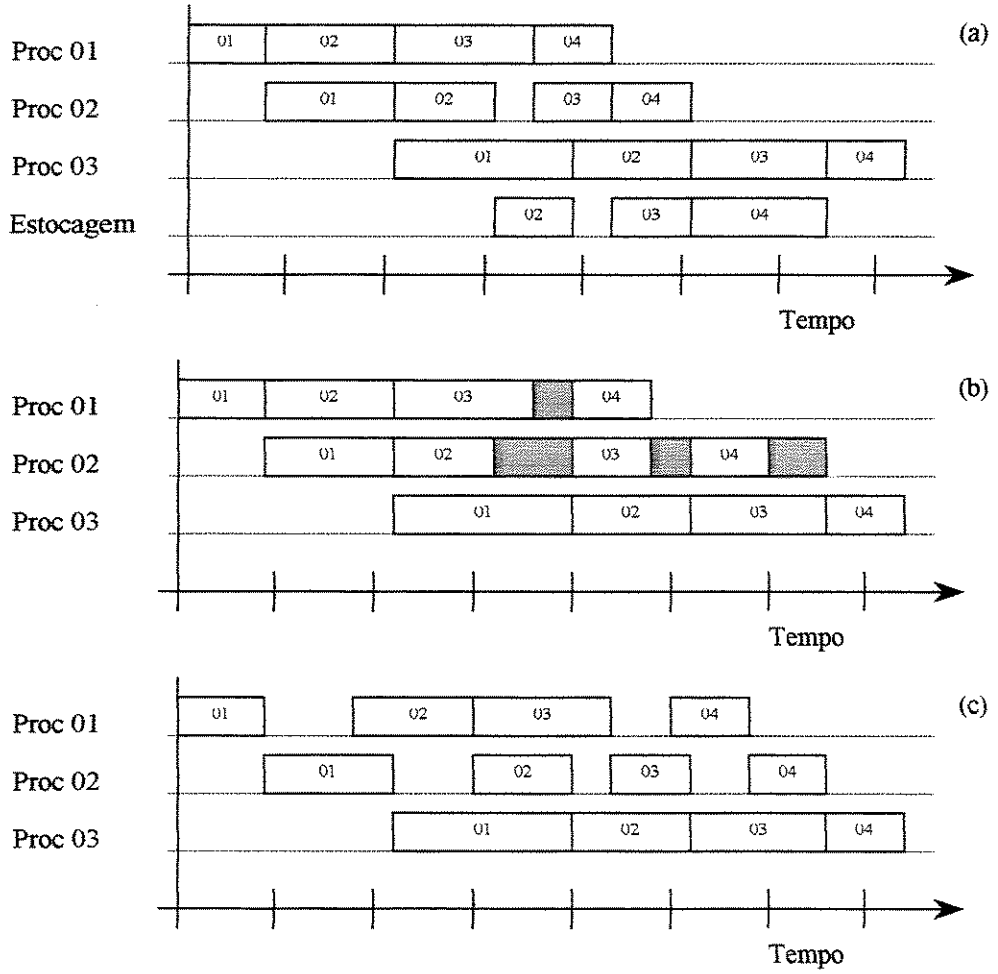


FIGURA 2.5: EFEITO DA ARMAZENAGEM INTERMEDIÁRIA SOBRE O TEMPO DE CONCLUSÃO DAS TAREFAS.

(a) Estocagem intermediária infinita (UIS).
(b) Não estocagem intermediária (NIS).
(c) Processamento sem espera (ZW).

2.3.3 Recursos Compartilhados e Outras Restrições

Programação de produção significa distribuir recursos disponíveis de acordo com restrições de oferta a fim de se alcançar um critério de desempenho. Como já foi visto, é muito grande o número de soluções possíveis para esse tipo de problema, sendo de grande importância, para limitar o espaço de busca de soluções possíveis, a identificação, representação e uso correto das restrições impostas a um problema.

As restrições podem ser classificadas como restrições físicas e restrições impostas pelo modo de operação da planta.

Restrições físicas dizem respeito aos recursos em comum, como processadores, tempo, capacidade de armazenagem intermediária, relações de precedência entre operações, tempos de limpeza e preparação de processadores, matéria prima, além dos já citados mão-de-obra, energia elétrica e vapor. As restrições impostas pelo modo de operação da planta estão relacionadas as datas de entrega dos pedidos e restrições sobre estoques intermediários.

Algumas restrições devem ser satisfeitas sempre e são chamadas de restrições rígidas (*hard constraints*) ou restrições de factibilidade. As restrições rígidas que reduzem o espaço de busca são chamadas de restrições estáticas e as restrições que geram conflitos no instante em que deve ser tomada uma decisão são chamadas de restrições dinâmicas. Uma restrição estática bastante presente nesses problemas são relações de precedência entre as operações, dadas pela rota de processamento.

Nos problemas de programação de produção, as restrições mais comuns são as dinâmicas, que dizem respeito à utilidades e mão-de-obra. Os processadores também são recursos compartilhados, cuja ocupação está vinculada à disponibilidade dos outros recursos consumidos durante o processamento (Campos (1993)).

Restrições que podem ser relaxadas (ou violadas), são chamadas de restrições flexíveis (*soft constraints*) e podem ser relaxadas mediante uma penalidade. Estão relacionadas a multas por atraso na entrega de pedidos, contratação de mão-de-obra extra e custo de estocagem extra.

2.3.4 Critérios de Otimização

Os critérios de otimização são geralmente de dois tipos: critério baseado no custo e critério baseado no desempenho. O critério baseado em custo é geralmente empregado em modelos de planejamento em nível único, nos quais quantidades bem como programação são resolvidas simultaneamente. Muitos trabalhos que tratam de estruturas mais complexas utilizam o critério baseado no desempenho. Os três critérios de desempenho mais usados são:

I) Minimização do tempo total de processamento de todos os produtos, também denominado *makespan*.

II) Minimização do tempo médio de residência de todos os produtos no sistema. O tempo médio de residência é também calculado usando coeficientes ponderados para cada produto.

III) Minimização da média máxima do atraso dos pedidos em relação ao tempo total de processamento e a data de entrega.

Se o problema de agendamento é tratado como um subproblema do planejamento de produção, os custos devem ser levados em consideração no nível superior de planejamento e desta forma, o critério baseado no desempenho é apropriado. Para a maioria dos problemas o critério de tempo total de processamento é satisfatório. Entretanto, se o sistema de produção é sobrecarregado, então os critérios baseados na média do tempo de residência e média do atraso máximo irão ajudar a garantir que as operações sejam alocadas de maneira a distribuir os atrasos de entrega.

2.4 Representação Rede Estado Tarefa (Kondili *et al.* (1993))

A representação da estrutura de processamento é um elemento essencial do modelamento. Como se viu nos diferentes exemplos anteriores (*flowshop, jobshop*), modelar o problema de programação da produção para cada estrutura de processamento é inviável, além disso, modificações na estrutura exigiram alterações no algoritmo de solução. Deve-se também ter em conta que as estruturas citadas representam apenas alguns exemplos de tipos de processamento.

Nos exemplos citados, é importante notar que está implícito que cada operação de uma determinada tarefa/produto tem apenas uma operação sucessora e uma operação precursora. Além disto, considera-se que cada operação produz apenas uma batelada, de modo que o

balanço de massa ao longo da estrutura de processamento já é previamente resolvido. Nos casos mais gerais, as estruturas são mais complexas com junções e separações, bem como o tamanho de batelada pode ser variável, exigindo que o número de bateladas de cada operação a ser realizada seja variável ao longo da estrutura de processamento.

De forma a permitir o modelamento de uma variedade bastante mais ampla de estruturas de processamento, será usada a representação Rede Estado-Tarefa (STN) introduzida por Kondili. A vantagem desta ferramenta é que permite representar adequadamente os problemas de balanço de massa em plantas compartilhadas.

Os processos em batelada são usualmente representados em termos da rota de produção também chamada de receita tecnológica. A rota de produção de cada produto é semelhante aos diagramas de plantas de processamento contínuo, mas com o objetivo de representar o processo em si e não uma planta específica.

Embora a representação por diagramas seja adequada para estruturas de processamento em série, o seu uso em estruturas mais complexas pode provocar ambigüidades na interpretação.

A figura 2.6 representa uma estrutura de processamento com cinco operações e o fluxo de material entre as operações é indicado por arcos. Essa representação não deixa claro se a operação 1 produz dois produtos diferentes que constituem a alimentação da operação 2 e 3 ou se há apenas um tipo de produto que é dividido entre as operações 2 e 3. Da mesma forma, não é claro se a operação 4 requer dois tipos diferentes de alimentação vindas de 2 e 5 ou se as correntes de alimentação são do mesmo tipo vindas de 2 e de um reciclo de 5.

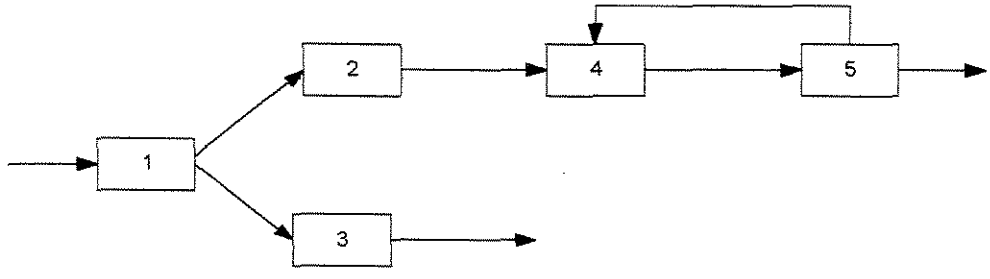


FIGURA 2.6: REPRESENTAÇÃO DA ROTA DE PRODUÇÃO DE UM PROCESSO QUÍMICO.

A representação rede estado tarefa (STN, *State Task Network*) foi proposta para eliminar essas ambigüidades. Ela foi desenvolvida inicialmente para descrever processos

químicos complexos, mas devido à sua clareza, permite a representação de processos químicos em geral, seja em plantas contínuas, semi contínuas ou descontínuas.

A rede estado tarefa (STN) possui dois tipos de nós, os nós estado que representam as matérias primas, produtos intermediários e produtos acabados, e os nós tarefa que representam as operações. Os nós estado são indicados por círculos enquanto as operações são indicadas por retângulos.

A figura 2.7 mostra duas representações STN diferentes que eliminam as ambigüidades citadas anteriormente. No processo representado pela figura 2.7 (a), a operação 1 gera um único estado intermediário que é divido entre as operações 2 e 3, e a operação 4 consome um único estado gerado pelas operações 2 e 5. O processo representado pela figura 2.7 (b) indica que a operação 1 gera dois estados intermediários que alimentam as operações 2 e 3, respectivamente, e a operação 4 consome dois intermediários produzidos pelas operações 2 e 5, respectivamente.

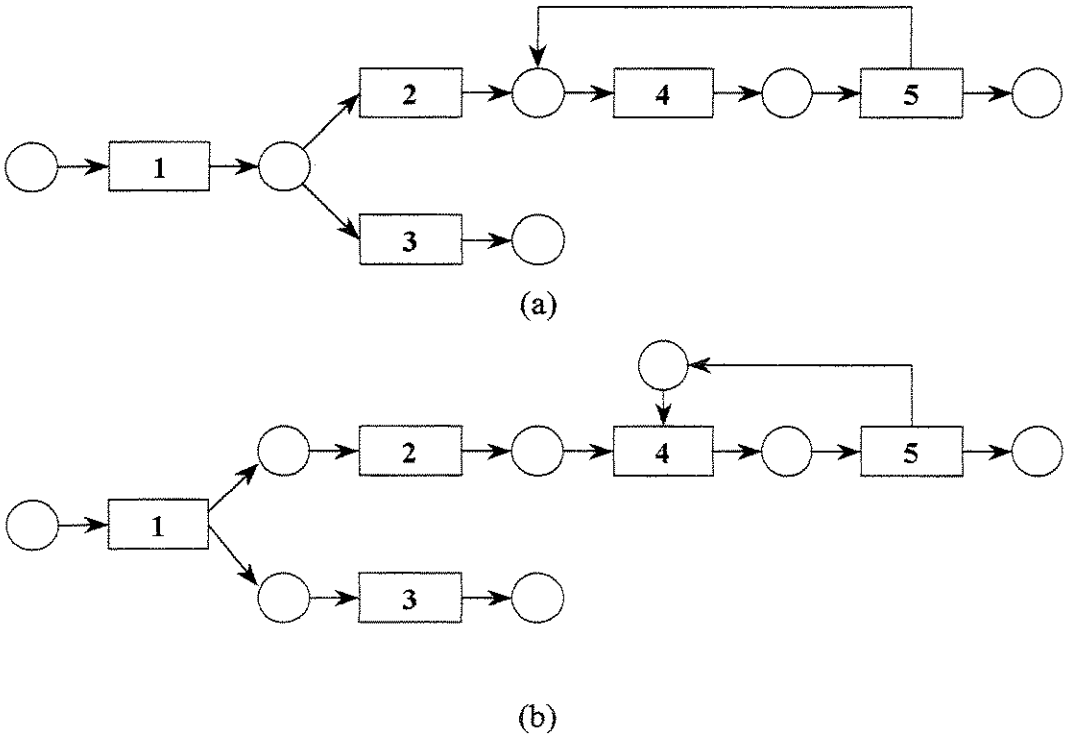


FIGURA 2.7: REPRESENTAÇÃO REDE ESTADO TAREFA DE PROCESSOS QUÍMICOS

A representação STN é igualmente apropriada para plantas de todos os regimes de produção; contínuos, semi contínuos ou batelada. As regras para sua construção são:

I) uma operação tem tantos estados de entrada (saída) quanto forem os materiais diferentes consumidos (produzidos).

II) duas ou mais correntes entrando no mesmo estado são necessariamente da mesma qualidade. Se a mistura de diferentes correntes é envolvida no processo, então esse processo deve constituir uma operação separada.

É importante enfatizar que a representação STN pode ser usada para representar o processo físico, propriamente dito, independente do tipo de problema que se deseja resolver. As unidades de processamento e a sua conectividade não são mostradas explicitamente nesse tipo de representação, assim como nenhum outro recurso disponível.

Com a finalidade de se encontrar uma solução ótima para um problema de planejamento e sequenciamento de produção, assume-se que uma operação recebe material de seus estados de alimentação, em proporções fixas e conhecidas, e que esta operação produz material para os seus estados de saída, também em proporções fixas e conhecidas. Além disso, os tempos de processamento de cada operação independem do tamanho da batelada de um produto e são previamente conhecidos.

No entanto, produtos diferentes de uma dada operação podem ter tempos de processamento diferentes (por exemplo, uma operação de destilação pode gerar vários produtos e com cada um destes produtos sendo gerado após um dado tempo de processamento).

2.5 Métodos de Solução dos Problemas de Programação de Produção

Como já foi mencionado anteriormente, o problema de programação de produção consiste em alocar N operações em M processadores e, de acordo com as características do problema, podem existir $(N!)^M$ soluções possíveis. Ainda que se limite o espaço de soluções, o problema de programação de produção em plantas multiproduto (*flowshop*) é da classe de problemas NP (Não Polinomiais) cuja característica é o aumento exponencial do tempo de solução de acordo com a dimensão do problema. Mesmo para problemas de dimensão reduzida torna-se necessário o uso de técnicas que possam abreviar a busca por uma solução.

A seguir são apresentadas três abordagens para a solução desses problemas. A escolha do método dependerá de fatores como a viabilidade da formulação matemática e informações disponíveis para reduzir a busca pela solução.

2.5.1 Métodos de Busca

A característica combinatorial dos problemas de programação de produção gera um grande número de soluções cuja enumeração explícita foge ao bom senso. Uma alternativa é recorrer a métodos de busca controlada para minimizar uma função de custo estabelecida. Estes métodos se caracterizam pela aplicação de duas atividades:

I) busca, são as maneiras possíveis de busca em árvore,

II) controle, seleção dentre as várias alternativas de caminhos que podem ser percorridos aquele que efetivamente será escolhido.

O problema de sequenciamento de plantas multiproduto vem sendo tradicionalmente tratado pelo método do tipo *Branch and Bound* (BAB), que é considerado uma enumeração parcial inteligente das soluções possíveis. As atividades de busca e controle podem ser realizadas por meio de diferentes tipos de cálculo e da incorporação de informações sobre a evolução do critério de otimização empregado, auxiliando a escolha do melhor caminho para a solução ótima e descartando, ao menos temporariamente, soluções não promissoras.

A sua representação gráfica é a de uma estrutura em forma de árvore, composta de nós que representam soluções parciais do problema, e ramos, que ligam nós de níveis sucessivos. Durante o processo de busca tem-se uma série de nós abertos (soluções parciais), e um custo total associado a cada nó. O custo associado a cada nó aberto é chamado de limitante inferior ou *lower bound*, que é composto por duas parcelas:

I) a primeira é relativa ao custo da solução parcial obtida até esse nó,

II) a segunda é uma estimativa do custo mínimo para se completar a solução.

Em geral, a estimativa da segunda parcela faz uso de uma heurística e é necessário garantir que o custo resultante da soma dessas duas parcelas não diminua ao longo da árvore, o que invalidaria a estratégia de caminho ótimo proposta no método BAB.

A solução vai sendo obtida com a alocação de tarefa por tarefa e o cálculo do custo associado, constituindo um nó. O menor custo associado a um nó indica o melhor caminho a ser seguido na árvore, assim a alocação das tarefas restantes segue até que seja obtida uma solução completa, cujo custo associado é chamado de limitante superior ou *upper bound*.

Uma vez encontrada uma solução completa, o método procura por soluções parciais com custo menor que o limitante superior, repetindo o processo de alocação de tarefas até que

seja encontrada uma outra solução completa ou um custo associado maior ou igual ao limitante superior. Este processo de revisitar nós abertos é também chamado de *backtracking*.

A eficiência do método está associada a estimativa da segunda parcela do custo, o uso de heurísticas que tornem essa estimativa bastante próxima do valor real reduz o número de ciclos necessários à construção da solução (*backtracking*). Por outro lado a heurística não deve se aproximar de uma simulação do caminho a ser percorrido, o que consumirá muito tempo computacional, aproximando a busca a uma enumeração completa.

O método BAB é tradicionalmente aplicado a problemas envolvendo plantas multiproduto devido a uma série de características:

I) neste método é fácil preservar o fluxo unidirecional das operações ao longo dos processadores, não é necessário o estabelecimento de restrições de precedência, bastando decidir qual será a próxima tarefa a ocupar os processadores.

II) pode-se definir os ramos da árvore nos quais se deseja realizar a busca.

III) soluções parciais infactíveis podem ser aceitas pelo usuário através do relaxamento seletivo de restrições ou imposição de penalidades.

Este método não exige uma formulação matemática rigorosa do problema, porém sua eficiência está intimamente relacionada à qualidade do custo mínimo calculado, chamado limitante inferior. A eficiência de busca pode ser aumentada com a construção de um limitante superior permitindo a eliminação de caminhos que possam conduzir a soluções de pior qualidade que serão posteriormente descartadas. A figura 2.8 ilustra as situações descritas acima para um problema de sequenciamento de 4 tarefas A, B, C e D (Rodrigues (1992)).

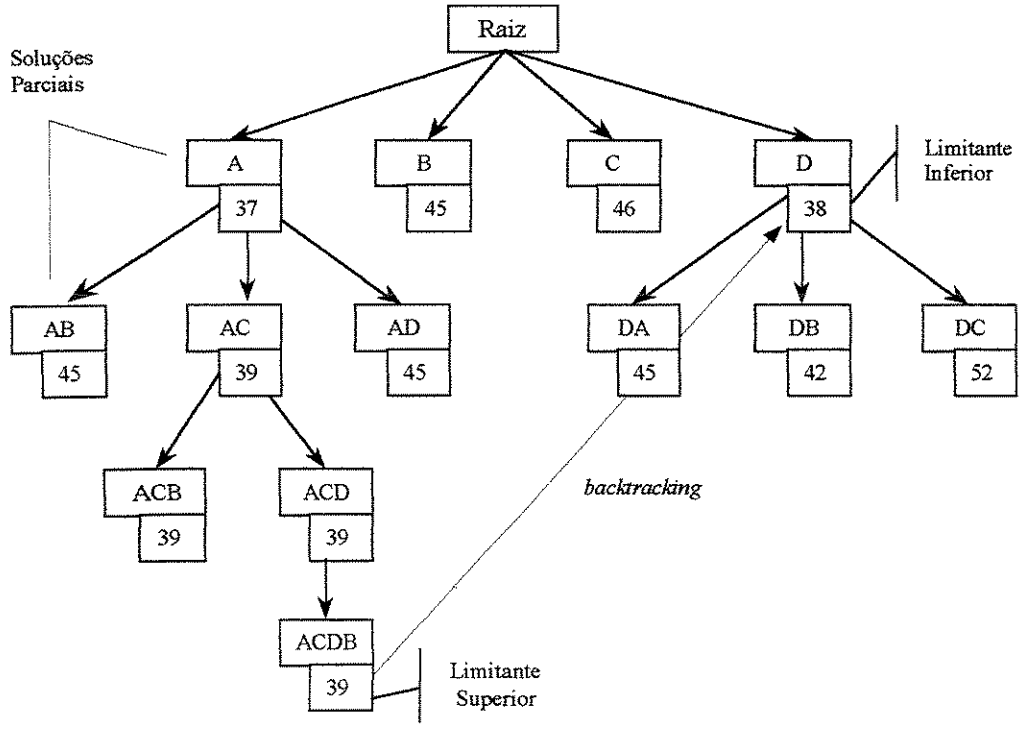


FIGURA 2.8: ÁRVORE DE BUSCA DO MÉTODO BRANCH AND BOUND.

2.5.2 Programação Matemática

O problema de programação de produção também pode ser resolvido através de formulação matemática de acordo com o seguinte procedimento:

I) análise do processo para identificação das variáveis e características específicas de interesse

II) definição do critério de desempenho

III) construção do modelo do processo, definindo as relações matemáticas entre as variáveis envolvidas

A natureza do problema de programação em unidades flexíveis resulta normalmente em um problema do tipo MILP (*Mixed Integer Linear Problem*) ou MINLP (*Mixed Integer Non Linear Problem*).

A estratégia de solução de problemas MILP é através da utilização de um algoritmo de programação linear e uma técnica de busca em árvore.

A resolução do problema é feita com o relaxamento inicial das restrições sobre as variáveis binárias, permitindo que estas últimas assumam valores entre 0 e 1. Esta solução obtém-se através de um algoritmo de programação linear (PL). A solução obtida não tem nenhum significado físico, porém o valor obtido para a função de custo serve como um limitante inferior da solução ótima, tal como na técnica BAB. Se alguma variável binária tem um valor real no resultado do PL, a sua modificação para um valor inteiro necessariamente não diminuirá o valor da função de custo.

O procedimento é a atribuição de valores inteiros para as variáveis binárias seguindo algum critério, por exemplo de custo marginal. Para cada variável são gerados dois subproblemas, um com a variável assumindo o valor 0 e o outro assumindo o valor 1. Cada subproblema é caracterizado como sendo o problema original relaxado com o valor da variável fixado em 0 ou 1 e cada subproblema é resolvido utilizando-se um algoritmo PL.

Tal como na técnica BAB, o valor da função de custo (em problemas de minimização) não diminui à medida que se segue pelos ramos da árvore, dessa forma a abertura de nós é controlada evitando a enumeração completa das soluções possíveis.

A questão da qualidade das soluções dos sucessivos PLs tem a ver com a distância entre o custo obtido em uma solução relaxada e o custo da solução ótima inteira. Esta diferença é conhecida como intervalo de integralidade ou *gap*, que cresce quando é grande o número de nós abertos pelo algoritmo, implicando em maior tempo de cálculo. As soluções relaxadas permitem que as variáveis binárias assumam valores reais com muita liberdade, originando um custo baixo. O resultado é a abertura de muitos ramos que serão eliminados adiante determinando um tempo computacional perdido.

2.5.3 Métodos Heurísticos

A resolução de um problema combinatório de forma exata não é tarefa fácil, na prática porém, é necessário que seja oferecida alguma solução realizável em tempo hábil ainda que distante do ótimo. Alguns algoritmos heurísticos proporcionam soluções razoáveis em tempo muito menor que o requerido pelos métodos exatos, os fatores que tornam seu uso interessante são:

I) inexistência de um método exato de resolução ou este requer muito esforço computacional. Oferece-se um resultado que só pode ser considerado bom frente ao fato de não se ter nenhuma solução.

II) quando há limitação do tempo, obrigando ao emprego de métodos de resposta rápida à custa da precisão.

III) como passo intermediário de outro algoritmo para geração de uma solução inicial para métodos iterativos.

IV) quando os dados são pouco confiáveis ou ainda o modelo não descreve bem a realidade, abre-se mão de soluções exatas já que estas são uma aproximação da realidade.

V) quando não se necessita da solução ótima, se os valores adquiridos pela função objetivo são relativamente pequenos pode não valer a pena o esforço por uma solução ótima. Pode ser suficiente a apresentação de uma solução melhor do que a atual.

Uma grande vantagem apresentada pelas heurísticas sobre as técnicas que buscam soluções exatas é que em geral permitem uma maior flexibilidade para o manejo das características do problema e geralmente oferecem mais de uma solução, permitindo ampliar as possibilidades de quem decide, sobretudo quando existem fatores não quantificáveis que não puderam ser adicionados ao modelo, mas que também devem ser considerados.

Uma desvantagem é o fato de não se poder avaliar a proximidade da solução obtida x_{heu} em relação à solução ótima x^*, se o problema é de minimização, sabemos apenas que $x^* \leq x_{heu}$. Existem métodos que permitem realizar avaliações a respeito da qualidade da solução, que implicam no relaxamento ou eliminação de algumas restrições, tornando mais fácil sua resolução. O relaxamento de restrições torna possível a obtenção de soluções melhores que a ótima, desta forma, se a melhor solução relaxada é x', sabemos que $x' \leq x^* \leq x_{heu}$ e assim, valores de x_{heu} próximos de x' garantem a qualidade da heurística.

Quando não são disponíveis este tipo de método de avaliação, cabe o uso de métodos sensíveis que detectam simplesmente que a heurística não é boa. Uma alternativa é a comparação com soluções de geração aleatória, a similaridade da solução gerada desta forma com a solução heurística coloca em dúvida a eficiência da heurística usada.

2.5.3.1 Tipos de Heurísticas

Os diferentes tipos de métodos heurísticos, muitas vezes usados de maneira conjunta, podem ser classificados de acordo com o modo de busca e construção de soluções, como apresentado a seguir:

I) **Métodos construtivos**. Acrescentam elemento por elemento à solução até obter uma solução factível. Os algoritmos mais populares são os chamados algoritmos devoradores (*greedy*), que buscam o máximo proveito em cada passo da construção da solução.

II) **Métodos de decomposição**. Dividem o problema em subproblemas menores, em geral a solução de um é a informação inicial do seguinte, a solução de todos os subproblemas resulta na solução completa. Alguns autores denominam de **decomposição** quando os subproblemas são resolvidos em série e **partição** quando os subproblemas são independentes entre si.

III) **Métodos de redução**. Buscam identificar características que a solução ótima presumivelmente deva possuir, e dessa forma simplificar o problema. Assim, pode-se detectar que alguma variável assuma sempre o valor 0 ou que outras estão correlacionadas, etc.

IV) **Manipulação do Modelo**. Modificam a estrutura do modelo tornando-o de mais fácil resolução e à partir da sua solução deduzir a solução do problema original. Pode reduzir o espaço de soluções com a linearização de funções e agrupamento de variáveis, ou aumentá-lo com a redução de restrições.

V) **Métodos de busca em vizinhança** ou **Métodos Progressivos**. Nessa categoria enquadram-se a maioria das metaheurísticas, caracterizadas por partirem de uma solução inicial e através de modificações, caminharem de forma iterativa para outras soluções, até que se cumpra um critério de parada, armazenando como ótima a melhor das soluções encontradas. Dois métodos que fazem parte dessa categoria são a busca tabu (*tabu search*) e o *Simulated Annealing*.

A figura 2.9 representa em forma de organograma a classificação dos diferentes algoritmos e técnicas heurísticas (Díaz et al. (1995)).

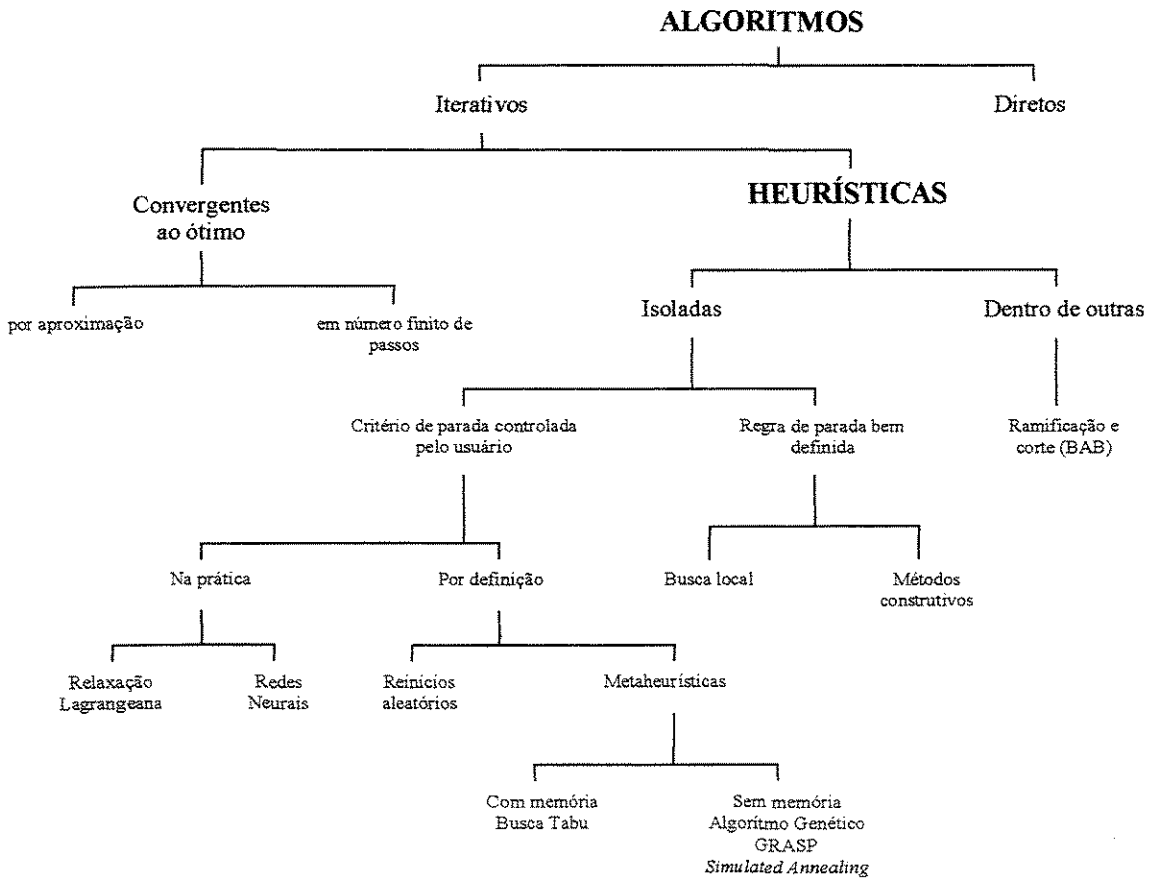


FIGURA 2.9: CLASSIFICAÇÃO DOS DIFERENTES ALGORITMOS E TÉCNICAS HEURÍSTICAS.

2.5.3.2 *Simulated Annealing*

O método *Simulated Annealing* vem sendo eficazmente utilizado na resolução de problemas de otimização global combinatorial, que têm características NP completos. É essencialmente um método progressivo randomizado que aceita soluções que pioram o valor da função objetivo, embora seja teoricamente um método exato, é considerado um método heurístico por não garantir a solução ótima (Ku e Karimi (1991)).

A figura 2.10 apresenta o algoritmo simplificado do método, que é iniciado com a construção de uma seqüência inicial e um valor inicial do parâmetro de controle T, denominado "temperatura". A seqüência inicial é um ordenamento das operações a serem alocadas, à qual, existe um valor associado (FO_1) da função objetivo, que se pretende

otimizar. A melhor seqüência encontrada (Seq_{min}) e a seqüência atual (Seq_1) assumem o valor da seqüência inicial e suas respectivas funções objetivo assumem o valor de FO_1.

O método analisa um número de seqüências candidatas (Seq_2) que são geradas por trocas randômicas dos elementos da seqüência atual, e o número de seqüências analisadas é dado por J, que não tem critério único para sua escolha.

A cada iteração, o valor da função objetivo (FO_2) da seqüência candidata, é comparado com o da melhor seqüência encontrada, substituindo-o em caso de melhoria. Da mesma forma, se FO_2 for melhor ou igual a FO_1, a seqüência candidata e valor da sua função objetivo, são aceitos como os respectivos atuais. Caso contrário, se FO_2 for pior que FO_1, é analisada a probabilidade de aceitação da seqüência candidata como seqüência atual, através de uma função (Algoritmo de Metrópolis) do parâmetro T. É feita, então, uma comparação do valor dessa função, caso este seja maior ou igual a um número randômico entre zero e um, a seqüência candidata e valor da sua função objetivo, são aceitos como os respectivos atuais, caso contrário, a seqüência é rejeitada e uma nova solução é gerada à partir da mesma seqüência atual.

A análise de aceitação da seqüência candidata é uma função que considera os valores de FO_1 e FO_2 e do parâmetro de controle T, que em geral é ajustado a cada vinte iterações, e dispõem de vários algoritmos para o recozimento. A literatura apresenta variantes da expressão proposta por Metrópolis como o algoritmo de Glauber, implicando na maior ou menor aceitação de soluções piores como geradoras de novas soluções, e são analisadas neste trabalho.

A busca é encerrada quando é alcançado um número de iterações ou tempo transcorrido, retornando os perfis de consumo de utilidades, valor da função objetivo e a melhor solução encontrada, que pode ser representada por uma Carta de Gantt.

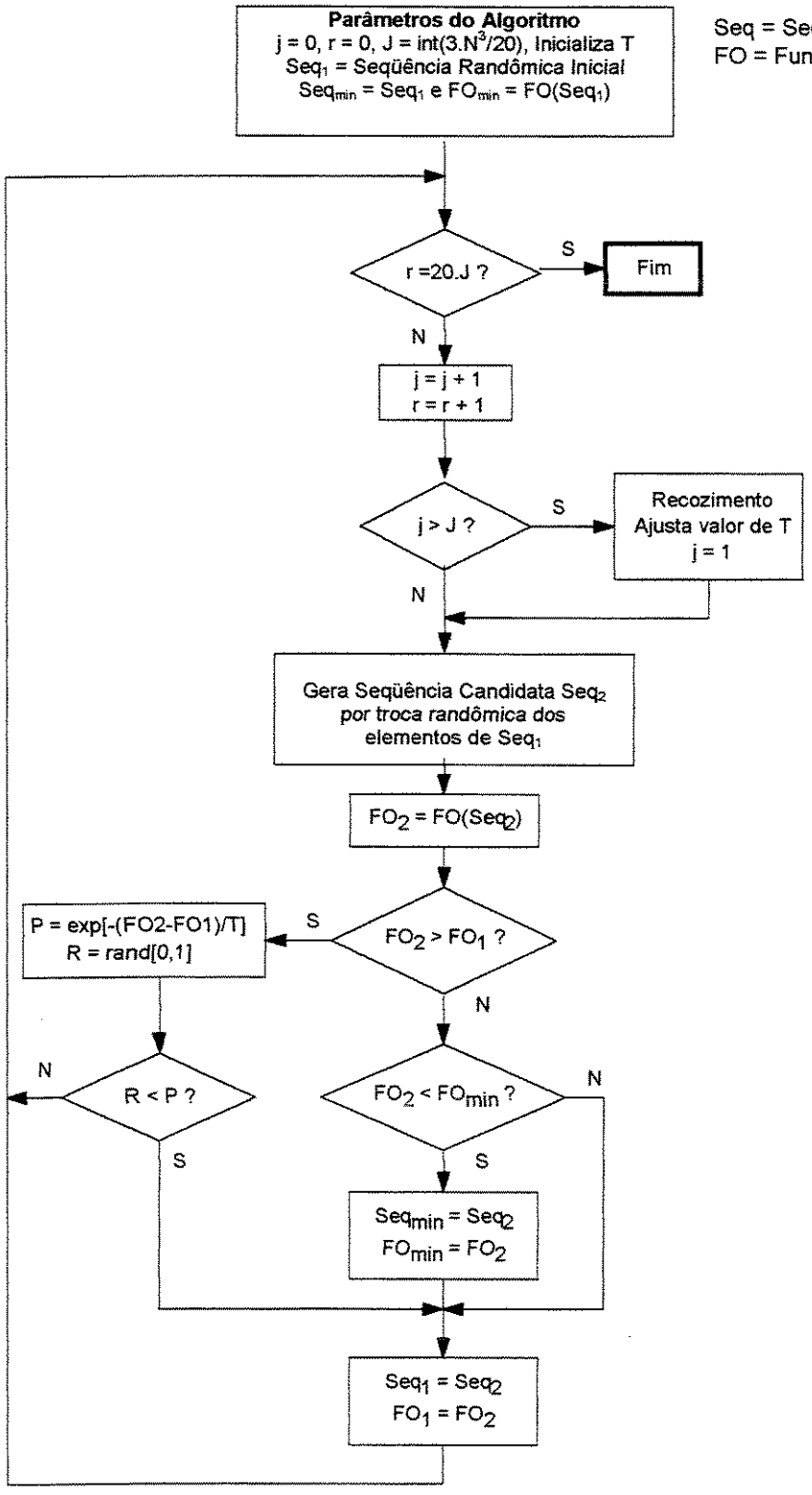


FIGURA 2.10: ALGORITMO SIMPLIFICADO DO MÉTODO SIMULATED ANNEALING.

3. A Abordagem Proposta

3.1 Estrutura Geral do Método Tradicional

O *Simulated Annealing* é um método heurístico de busca em vizinhança sem memória, e consiste basicamente na otimização de uma função objetivo através da geração de novas soluções à partir de uma solução inicial factível e retendo a melhor solução encontrada. A sua principal característica é a aceitação de soluções piores como precursoras de outras soluções que podem vir a ser melhores, escapando assim de mínimos locais. Possui diversas variações dos seus parâmetros básicos e do próprio algoritmo, conferindo versatilidade para aplicação à diversos tipos de problema.

3.2 Características do Método *Simulated Annealing*

O *Simulated Annealing* é baseado no algoritmo Monte Carlo e foi desenvolvido por Metropolis *et al.* (1953) para o cálculo de propriedades de substâncias, que podem ser consideradas como resultado da iteração individual de suas moléculas. Kirkpatrick *et al.* (1983) o aplicaram com êxito na resolução de alguns problemas combinatórios, como o projeto de circuitos de computador e o problema do caixeiro viajante. A estratégia também ofereceu bons resultados no projeto de circuitos integrados e Dolan (1987) o utilizou na síntese de redes de trocadores de calor. Corana (1987) propôs um algoritmo derivado do *Simulated Annealing* para otimização de problemas com variáveis contínuas.

A característica combinatorial e NP completo dos problemas de programação de produção levou outros autores a utilizarem o *Simulated Annealing* na sua resolução. Osman (1989) o aplicou a uma planta multiproduto com seqüências de permutação. Karimi provou seu desempenho comparando seus resultados com os obtidos por um algoritmo randômico, aplicados à minimização do tempo total de processamento de uma planta multiproduto com estocagem intermediária infinita.

Das *et al.* (1990) fez uma comparação diversificada dos parâmetros do método, como algoritmos, modos de ajuste do parâmetro de controle (recozimento) e políticas de armazenagem, e Connolly (1992) obteve bons resultados para problemas de programação linear expressados puramente como 0 – 1.

3.2.1 Definição do Método *Simulated Annealing*

O método SA é constituído pela entrada de parâmetros iniciais, como temperatura inicial, temperatura final e solução inicial. A maior dificuldade é a seqüência inicial, que por si só já constitui uma solução cuja factibilidade deve ser garantida.

As novas seqüências são geradas à partir da troca randômica das posições das operações que compõem a seqüência inicial, e o valor de sua função objetivo é obtida através da alocação temporal das operações que a compõem de modo a satisfazer as restrições do problema tais como, a precedência de operações, balanço de massa, consumo de recursos e instante máximo para término da operação.

O valor da função objetivo da nova seqüência é então comparado com o da seqüência anterior, caso represente uma melhoria na busca, é automaticamente aceita. Caso contrário, quando a seqüência não representa uma melhoria na função objetivo, o SA faz uma análise probabilística se esta pode ser aceita ou não, pois a mesma pode representar uma forma de saída de mínimos locais, realizando a "subida de montanha" para encontrar outro vale e o possível mínimo global. Nestas duas situações, a seqüência atual é substituída pela nova e os parâmetros como temperatura são atualizados à partir desta. Se a nova seqüência é rejeitada, é feita apenas o ajuste da temperatura e contadores de iterações, mantendo-se a seqüência atual.

O ciclo segue com a geração de novas seqüências à partir da atual, repetindo o processo de avaliação até que se atinja algum critério de parada como número de iterações, temperatura final ou tempo de busca, condição em que o SA retorna a melhor solução armazenada.

Seqüências infactíveis podem ainda ser identificadas com testes rápidos que relaxam restrições de consumo de recursos e que consideram apenas a precedência de operações, evitando o esforço de alocação completa que consome a maior parte do tempo de processamento. As seqüências infactíveis ou as mais recentes podem ainda constituir uma lista de soluções tal como na Busca Tabu, evitando a repetição de alocação para uma seqüência e aumentando a exploração do espaço de busca.

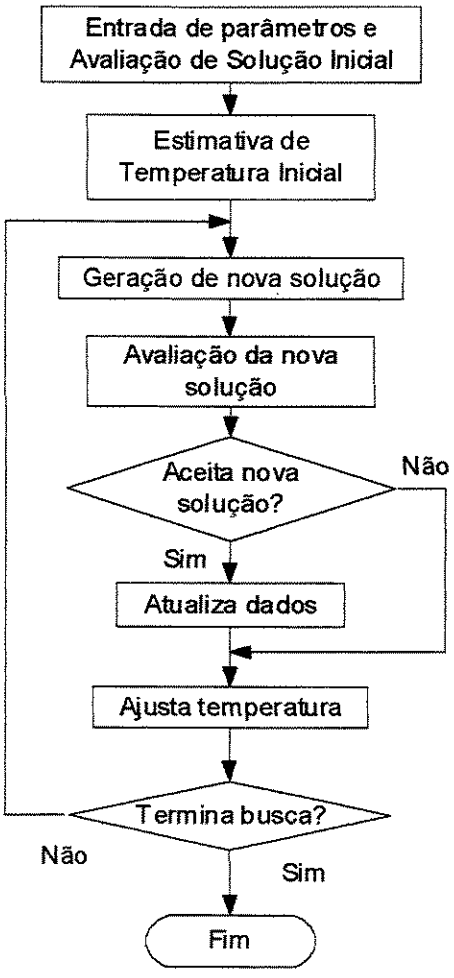


FIGURA 3.1: ESTRUTURA GERAL DO MÉTODO SIMULATED ANNEALING.

3.2.2 Origem do Método *Simulated Annealing*

O método *Simulated Annealing* foi desenvolvido por Metropolis *et al.* (1953) com o intuito de investigar equações de estado para substâncias cuja iteração individual de suas moléculas para uma dada configuração e a uma determinada temperatura, resulta em uma variação de energia ΔE. De acordo com a temperatura, o sistema pode alcançar estados de equilíbrio metaestáveis, contendo ainda grande quantidade de energia.

É um método de busca e tem a característica de poder sair de mínimos locais através da aceitação de configurações ou soluções piores que a anterior, realizando a chamada "subida de montanha" e alcançando outros "vales" onde pode-se encontrar o mínimo global. Uma diferença é que ao invés de se escolher configurações de maneira randômica e ponderá-las

com $\exp^{(-\Delta E/T)}$, escolhe-se configurações com probabilidade $\exp^{(-\Delta E/T)}$, ponderando-as igualmente para a sua aceitação.

Na sua forma original, considera-se um sistema formado por uma cadeia de átomos inicialmente em equilíbrio a uma dada temperatura, ao qual está associada uma quantidade de energia. Em cada passo do algoritmo, a posição de um dos átomos é alterada randomicamente criando uma seqüência candidata e a alteração na energia do sistema, ΔE, é calculada. Se $\Delta E \leq 0$, significa que a energia diminuiu, o sistema caminha para a estabilidade, o movimento é então aceito e a configuração passa a ser a seqüência de origem para o próximo passo.

Se $\Delta E > 0$, o movimento é tratado de maneira probabilística, cuja probabilidade de aceitação é dada por: $P(\Delta E) = \exp^{(-\Delta E/T)}$. A probabilidade de aceitação $P(\Delta E)$ é comparada com um número randômico, distribuído uniformemente no intervalo (0,1). Se $P(\Delta E)$ for maior que este número, o movimento é aceito e a seqüência passa a ser o ponto de partida para o próximo passo, caso contrário a configuração é rejeitada e cria-se outra seqüência à partir da mesma seqüência anterior.

O parâmetro de controle T é ajustado de acordo com critérios escolhidos, inicialmente possui valores grandes que resultam em valores de $P(\Delta E)$ próximos de 1, ou seja, a maioria das seqüências são aceitas. À medida que o método vai analisando o espaço de busca e vão sendo encontradas soluções otimistas, T vai sendo reduzido gradualmente por critérios de recozimento, tornando mais difícil a aceitação de soluções piores e assim, intensificando a busca. A melhor solução encontrada é armazenada e atualizada durante todo o procedimento, que é encerrado por um critério de parada como tempo ou número de iterações.

A repetição dos passos básicos simula o movimento dos átomos em um banho térmico à temperatura T, passando por diferentes níveis de energia, alguns metaestáveis. A analogia com os problemas de otimização é feita com a consideração de uma função objetivo como sendo a energia do sistema e definindo um conjunto de parâmetros $\{x_{i,j}\}$ como configurações. A temperatura é somente um parâmetro de controle com as mesmas unidades da função objetivo e é o que permite o método aceitar soluções piores, saindo assim de mínimos locais.

O desempenho do método pode ser incrementado com o uso de uma lista de seqüências recém visitadas, evitando o processo de alocação de seqüências repetidas. A vantagem do uso de uma lista de seqüências é a economia de tempo com seqüências repetidas, já que o processo de alocação consome em torno de 50% do tempo computacional total.

3.3 Parâmetros do Método

Obviamente não há regras para eleger o melhor conjunto de parâmetros, sendo que a decisão é muito dependente das idéias gerais associadas à esses parâmetros e às características do problema tratado.

3.3.1 Parâmetro de Controle e Temperatura Inicial T_o

No método SA aplicado à problemas de programação de produção, a Temperatura é apenas um parâmetro de controle para a aceitação de novas configurações. De modo geral, um método heurístico de busca não deve ser dependente da solução inicial, isto é conseguido pelo SA através de valores do parâmetro de controle suficientemente grandes para que todas as configurações iniciais ocorram com a mesma probabilidade, realizando uma diversificação da busca. Esta exigência é necessária, mas não suficiente para que o método não caia em um mínimo local. Valores positivos da variação da função objetivo, ΔE, representam a possibilidade da busca sair de mínimos locais, se o valor inicial do parâmetro de controle, T_o, é muito pequeno em relação a ΔE positivo, a nova solução provavelmente não será aceita e consequentemente o processo de otimização ficará preso a um mínimo local.

O valor inicial do parâmetro de controle não possui uma regra específica para sua escolha e diversas opções são apontadas pela literatura. Karimi (1990) usou T_o igual a 1,5 vezes a variação máxima da função objetivo, obtida através da geração de um número de soluções randômicas. Aarts e van Laarhoven (1985) e Das *et al.* (1990) utilizaram, de maneira simplificada ou não, uma formulação mais lógica, considerando P_o a probabilidade de aceitação específica das soluções candidatas e μ a diferença percentual da solução candidata, o parâmetro T_o pode ser escrito em função da variação da função objetivo, ΔE, como:

$$\mathbf{Po = e^{(-\mu.\Delta E / To)} \Rightarrow To = -\mu.\Delta E / Ln(Po)\ ,\ [3.1.a]}$$

que, para a aceitação de 90 % das soluções que sejam 50 % piores que a atual, resulta em:

$$\mathbf{To = -0{,}5.\Delta E / Ln(0{,}9) \Rightarrow To \approx 4{,}8.\Delta E,\ [3.1.b]}$$

ou ainda $T_o \approx 10.\ \Delta E$ para soluções 100 % piores que a atual.

A estimativa de ΔE pode ser obtida por algumas soluções randômicas, retendo o valor máximo, ou mesmo de maneira intuitiva quando já se conhece a amplitude das soluções do problema.

3.3.2 Temperatura Final T_f

Teoricamente, o valor do parâmetro de controle correspondente ao mínimo global assumiria o valor 0, entretanto, a probabilidade de aceitação de uma solução é praticamente nula para valores pequenos de T, portanto a busca pode ser encerrada com valores de T>0. O valor final do parâmetro de controle é geralmente uma função do número de soluções possíveis ou do número de soluções que se deseja analisar para uma mesma temperatura.

Em geral são analisadas 20 ou 30 soluções a uma mesma temperatura, que é corrigida por um fator de ajuste α correspondendo de 80 a 99 % da temperatura anterior. O valor final pode ser escrito como:

$$Tf = To.\alpha^{30}, \quad [3.2]$$

Outros critérios consistem em interromper a busca após analisar um número de soluções sem que haja uma melhora de 2 % na função objetivo ou ainda obedecendo a um critério de parada por tempo de busca.

3.3.3 Ajuste do Parâmetro de Controle

Seguindo a analogia, o número de soluções analisadas a uma mesma temperatura deveria ser suficientemente grande para que o sistema chegasse ao equilíbrio. Se NT é o número total de soluções possíveis, o número de soluções analisadas deveria ser pelo menos igual a NT^2. Tamanho esforço não é recomendável e ainda que factível para problemas de pequena dimensão, deve-se encontrar um meio termo entre o número de iterações à mesma temperatura e a velocidade de decremento da temperatura, ao ser definido o algoritmo.

No ajuste exponencial da temperatura utilizado por Kirkpatrick *et al.* (1983), cada nova temperatura ,T', é obtida multiplicando a temperatura corrente, T, por um fator de ajuste α, sendo $0 < \alpha < 1$. Podemos expressar:

$$T' = T.\alpha, \quad [3.3]$$

Quanto menor o valor de α, mais rápido é o resfriamento e na prática, para um resfriamento lento, o valor de α encontra-se entre 0,8 e 0,99.

Para obter a nova temperatura, Aarts e van Laarhoven (1985), utilizaram um fator de ajuste variável, dependente da proximidade do equilíbrio que se obtém à temperatura atual. Isso implica que a uma dada temperatura, quanto mais próximo do equilíbrio o sistema se encontra, menor é o fator de ajuste e mais rápido é o resfriamento. A idéia básica é fazer o resfriamento a uma taxa que mantenha o sistema próximo ao equilíbrio a cada temperatura. Matematicamente pode ser escrito como:

$$T' = T / \{1 + (Ln(1+\delta).T) / 3.\sigma(T)\}, \quad [3.4]$$

em que $\sigma(T)$ é o desvio padrão da função objetivo, δ é um valor específico da proximidade do equilíbrio desejada e é geralmente assumido como 0,3. Quanto menor o valor de δ, maior é a proximidade do equilíbrio desejada para o sistema e mais lento é o resfriamento. E quanto maior o desvio padrão da função objetivo, mais distante do equilíbrio está o sistema e mais lento é o resfriamento.

Lundy e Mees, (1986), sugerem um resfriamento constante em função da temperatura inicial, temperatura final e do número total de soluções que pretende-se analisar, T_o, Tf e NT, respectivamente, que é expressado como:

$$T' = T / (1 + b.T), \quad [3.5.a]$$

$$b = (To - Tf) / NT.To.Tf, \quad [3.5.b]$$

3.3.4 Critérios de Aceitação de Movimentos

A principal característica do método *Simulated Annealing* é o fato de poder sair de mínimos locais, aceitando soluções piores sob uma probabilidade e assim transpondo pontos de máximo. A literatura é unânime na utilização do algoritmo de Metropolis, (Metropolis *et al.* (1953)), para o qual a probabilidade de aceitação da solução candidata, $P(\Delta E)$, é uma função da variação da função objetivo ΔE, dada por:

$$\begin{cases} P(\Delta E) = 1 & \text{, para } \Delta E \leq 0 \\ P(\Delta E) = e^{(-\Delta E/T)} & \text{, para } \Delta E > 0 \end{cases} \quad [3.6]$$

desta forma, todas as soluções que melhoram a função objetivo são aceitas com probabilidade 1 e soluções que pioram a função objetivo podem ser aceitas com probabilidade entre 0 e 1, decrescendo com o valor do parâmetro T. A variação da probabilidade de aceitação P(ΔE), em função da variação da função objetivo ΔE e da Temperatura para o algoritmo de Metrópolis é apresentada graficamente à seguir.

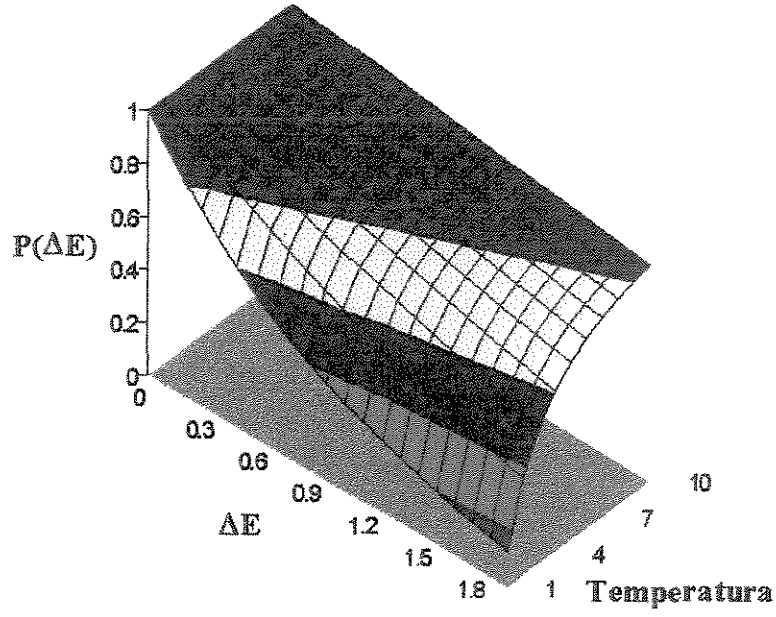


FIGURA 3.2: PROBABILIDADE DE ACEITAÇÃO DE SOLUÇÕES EM FUNÇÃO DA VARIAÇÃO DA FUNÇÃO OBJETIVO E DA TEMPERATURA.

O algoritmo de Glauber, (Glauber (1963)), não aceita necessariamente todas as soluções que melhoram a função objetivo; a probabilidade de aceitação de qualquer movimento pode ser escrita como:

$$P(\Delta E) = e^{(-\Delta E / T)} / (1 + e^{(-\Delta E / T)}), \quad [3.7]$$

Isso implica que para valores altos de T, todas as soluções, melhores ou piores, são aceitas com probabilidade próxima a 0,5. Com a redução de T, a probabilidade de aceitação

de soluções piores decresce de 0,5 a 0, enquanto vai aumentando de 0,5 até 1 para as soluções que melhoram a função objetivo.

O algoritmo de Glauber demora mais para chegar ao mínimo global, quando em sua vizinhança, pois não aceita automaticamente todos os movimentos de "descida de montanha".

Por outro lado, o algoritmo de Metrópolis pode aceitar movimentos que colocam o sistema em um mínimo local, o que o algoritmo de Glauber pode rejeitar. A priori, não é possível dizer qual dos dois algoritmos é mais eficiente sem que se conheça o comportamento do espaço de busca.

3.3.5 Outros Parâmetros

A diversidade de problemas a que o SA vem sendo aplicado faz com que os autores modifiquem e adaptem os parâmetros originais de acordo com sua necessidade. As modificações podem ser sutis como número total de soluções analisadas, número de análises para cada temperatura ou taxa de resfriamento. Outras envolvem a utilização de mais de um método, como o reaquecimento, em que, após um certo número de soluções frustradas retorna-se à melhor solução encontrada e a busca prossegue com o parâmetro de controle aumentado e constante.

O *Simulated Annealing* é um método de busca sem memória, mas pode-se incluir uma lista atualizável das últimas n soluções visitadas, tal como a Busca Tabu, desta forma não é desperdiçado tempo computacional com soluções repetidas ou infactíveis, sem grande consumo de memória.

3.4 A Estratégia Proposta

No Capítulo 2 foi introduzido brevemente o método SA como uma ferramenta para a solução de problemas de diferentes naturezas. Uma característica importante a ressaltar neste tipo de abordagem é que, na sua idéia básica, o SA foi proposto para resolver problemas de otimização sem restrições.

Neste trabalho foi proposta a sua utilização para a solução de problemas de programação de produção na presença de diversas restrições tais como:

- precedência tecnológica, que tem como objetivo garantir a rota de produção;
- restrições de balanço de massa, que têm como objetivo garantir que uma operação somente pode ser realizada se houver massa suficiente;
- restrições sobre prazos de entrega.

O primeiro tipo de restrição tem sido freqüentemente tratada nos problemas de "sequenciamento de tarefas em *flowshops*". Como já se viu anteriormente, o "*flowshop*" estrito consiste em uma estrutura de processamento serial em que todas as tarefas seguem a mesma seqüência. Freqüentemente para este problema, restringe-se sua solução às chamadas seqüências de permutação. Além disso, o critério de desempenho normalmente empregado é o *Makespan*. Consequentemente, o problema tradicionalmente tratado na literatura é determinar a seqüência de produtos que minimiza o tempo total de execução das tarefas. A inexistência de restrições sobre a geração de seqüências de produção tem duas conseqüências:

- qualquer seqüência é factível de modo que a simples permutação de quaisquer produtos gera uma nova seqüência;
- uma vez que uma nova seqüência for definida, seu "custo" pode ser avaliado rapidamente.

Há portanto uma separação entre os problemas de "sequenciamento" e programação (*timing*). Neste caso, pode ser rapidamente avaliado um grande número de soluções em um intervalo de tempo reduzido.

A aplicação deste método a estruturas de processamento complexas, em que haja restrições do 2º e 3º tipos é que movimentos para a geração de seqüências candidatas podem levar a soluções infactíveis do ponto de vista de balanço de massa e/ou que violem os prazos de entrega. Essa pré-qualificação elimina a maior parte das seqüências infactíveis, uma vez que analisa a ordem das operações na seqüência, verificando se a operação que gera um estado está antes de outra que consome, evitando maiores esforços.

Neste trabalho é proposta uma estratégia de solução via SA usando uma abordagem de duas fases:

1ª fase: pré qualificação da seqüência candidata através da sua análise com respeito à factibilidade de balanço de massa.

2ª fase: construção da carta de Gantt para calcular o valor do custo.

O objetivo desta estratégia é a eliminação de soluções potencialmente infactíveis do ponto de vista balanço de massa, de modo a impedir que seja construída a carta de Gantt destas candidatas.

Os dados do problema a ser tratado são:

- representação STN. Conjunto de processadores habilitados para realizar cada operação, relações de consumo e geração de massa entre operações e tempo de processamento das operações;
- número de bateladas de cada operação;
- intervalo de tempo em que elas devem ser realizadas de modo a atender o prazo de entrega (definição de janelas de demanda).

A factibilidade de uma solução candidata é analisada em termos da satisfação do balanço de massa e da possibilidade de realização dentro da sua janela de demanda.

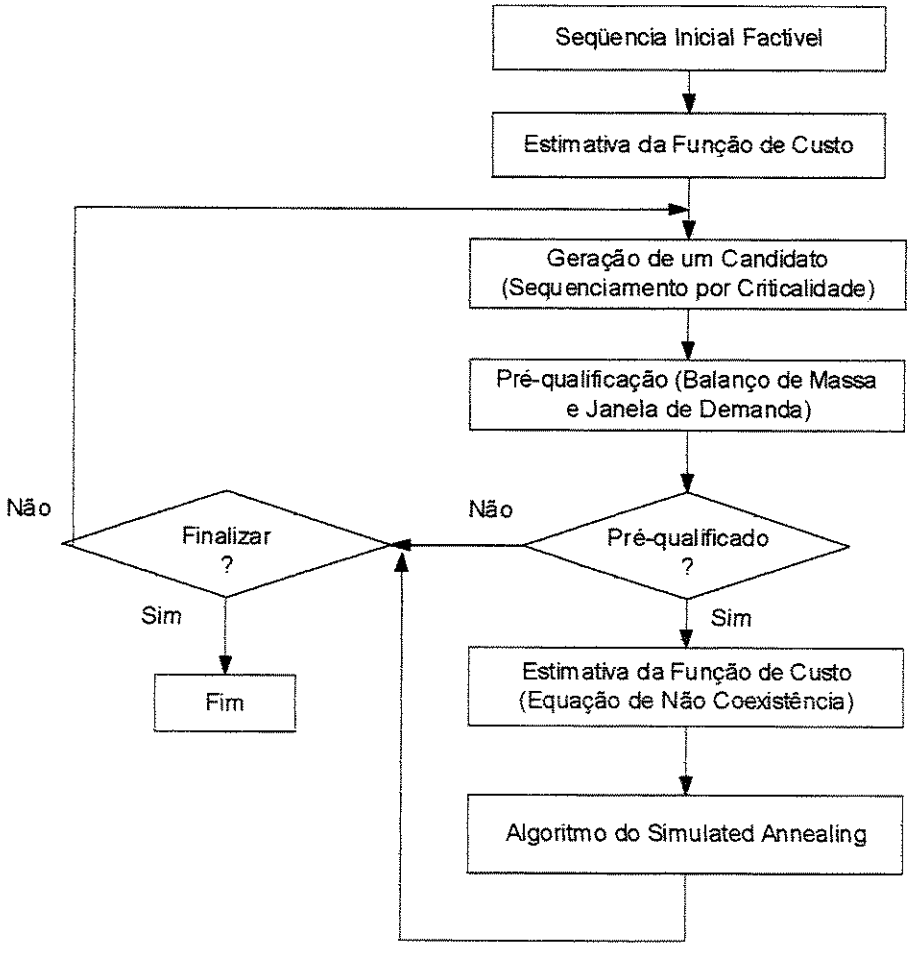


FIGURA 3.3: FLUXOGRAMA DO TESTE DE FACTIBILIDADE DE SEQÜÊNCIAS CANDIDATAS.

3.4.1 Seqüência Inicial

Uma característica do SA é o fato de não depender da qualidade da solução inicial, ainda que esta seja um mínimo local o método possibilita explorar outras áreas do espaço de busca e até obtenção do mínimo global. Essa qualidade é garantida em parte por valores

iniciais grandes do parâmetro de controle T, porém a obtenção de uma solução inicial já não é tarefa fácil dependendo da complexidade do problema dada por restrições de precedência, balanço de massa e janelas de tempo.

O SA é um método heurístico que, como já se disse, tem como objetivo a solução de problemas sem restrições ou com restrições "fracas". À medida que o número de restrições e sua importância na solução aumentam, torna-se mais difícil a geração de uma solução factível inicial. Dependendo do caso, pode ser aceitável a geração de uma solução inicial que viole algumas restrições chamadas "relaxáveis" tais como prazos de entrega. Deve-se, no entanto, ter em mente que se persegue uma solução que satisfaça todas as restrições do problema.

Para problemas mais relaxados, com um grande espaço de soluções factíveis, a construção da seqüência inicial é mais fácil quando é seguido algum critério de ordenamento, como o de criticalidade decrescente. Desta forma, é priorizada a alocação daquelas operações cuja criticalidade se aproxima de 1, ou seja, daquelas cujo tempo de processamento se aproxima da duração da sua janela de alocação, enquanto as que possuem alocação mais flexível assumem as últimas posições na seqüência candidata.

Tanto para a seqüência inicial como para as candidatas, pode ser feito um teste de factibilidade e reordenamento da seqüência por precedência, em que não é considerada a quantidade de matéria gerada ou consumida por operação, mas apenas a precedência de operações que geram matéria-prima para as operações que as consomem. Este teste, por não considerar massa, consumo de recursos ou alocação temporal, é rápido e reduz significativamente o número de seqüências infactíveis enviadas para o processo de alocação.

3.4.2 Janelas de Demanda

O conceito de janelas de demanda é utilizado para a redução do espaço de busca do problema e pode ser definido como o intervalo de tempo no qual uma tarefa deve ser executada. Seus limites são definidos pelo instante mais cedo que a tarefa pode ser iniciada (EBT), determinado por restrições como disponibilidade de matéria-prima ou de processador, e pelo instante mais tarde que a tarefa pode ser finalizada (LFT), determinado por datas de entrega ou níveis de estoque que devem ser repostos.

Através da propagação de restrições por precedência tecnológica à partir da janela de demanda de tarefas, é possível a definição das janelas iniciais das operações que compõem cada tarefa e por conseguinte, as janelas das bateladas que compõem essas operações. Desta

forma, o EBT da tarefa é o início da janela da primeira operação, que somado ao seu tempo de processamento constitui o início da janela da segunda operação e assim por diante.

O LFT da tarefa define o fim da janela da última operação, cujo tempo de processamento subtraído do LFT resulta o fim da janela da operação anterior e assim por diante. A construção de janelas para bateladas é uma extensão desse raciocínio.

A construção dessas janelas é um fator que determina o sucesso da alocação, constituindo uma forte restrição que conduz à soluções otimistas ou indicam prontamente uma infactibilidade.

3.4.3 Conceito de Criticalidade

Uma vez que cada operação possui um tempo de processamento e que deve ser executada dentro de uma janela de tempo, o quociente desses dois termos define a criticalidade da operação. Criticalidade de uma operação é uma medida da flexibilidade da operação para fins de alocação, quanto maior o valor, menos flexível é a operação, então mais crítica é a sua alocação.

Desta forma se o tempo de processamento de uma operação é muito menor que a duração de sua janela, a sua alocação é bastante flexível, por outro lado, se a duração da janela é próxima ou igual ao tempo de processamento, a criticalidade dessa operação tende a 1, havendo uma única possibilidade para sua alocação o que a torna prioritária. Janelas de tempo menores que o tempo de processamento representam infactibilidades.

3.4.4 Construção de Seqüência Candidata

Das *et al.* (1990) testaram cinco tipos de movimentos para a construção de seqüências candidatas à partir da seqüência atual, das quais foram testadas três neste trabalho.

À partir da seqüência inicial, são geradas novas seqüências através da alteração randômica da posição dos seus elementos. Quanto maior a alteração da seqüência inicial mais se obtêm diversificação, explorando uma maior área do espaço de busca obtendo-se referências da função objetivo. Ao longo do processo o método pode ser mais seletivo, realizando alterações mais restritas na seqüência, obtendo-se assim uma intensificação na busca.

Os três tipos aplicados neste trabalho foram escolhidos seguindo sugestão dos autores e comparação de resultados. Ainda são citadas outras técnicas, promissoras para tipos

particulares de problema, como a inversão de subseqüências e a inserção e deslocamento de subseqüências. Alguns autores indicam a utilização de mais de um método para a construção de novas seqüências, de acordo com o estágio do algoritmo; alguns deles são apresentados à seguir :

I)- Inserção e deslocamento de elementos: são escolhidas duas posições randômicas da seqüência, um dos elementos é removido de sua posição e inserido na outra posição, deslocando todo o restante da seqüência.

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

II)- Permutação de elementos: os elementos correspondentes às posições são trocados sem que se altere o restante da seqüência.

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

III)- Permutação de elementos adjacentes: as permutações são mais refinadas, restringindo-se a elementos vizinhos e intensificando a busca; este método não é indicado para a etapa inicial do método.

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

3.4.5 Modelamento do Problema Usando a Representação STN

A representação do tempo é uma consideração básica nos problemas de programação de produção. A formulação a ser utilizada é baseada na discretização do horizonte de alocação em intervalos de tempo de igual duração, também chamados de *slots* de tempo, t. Eventos de qualquer espécie, tais como início e fim de processamento ou mudança de disponibilidade de equipamentos e utilidades só poderão ocorrer nos extremos desses intervalos de tempo.

A utilização dessa representação fornece uma grade de referência de intervalos regulares, na qual são posicionadas todas as operações (processamento, preparação de processador e limpeza) que competem por recursos compartilhados. A duração de um

intervalo de tempo é tomada como sendo o máximo divisor comum entre os tempos de todas as operações envolvidas no problema, portanto, um *slot* pode ter a duração de frações de hora ou até mesmo semanas.

O modelamento do problema considera que em um nível superior de planejamento são definidos o número e tamanho de bateladas, além da habilitação de processadores para a execução de cada operação, janelas de demanda, consumo de recursos, tempo de processamento e política de estocagem intermediária. A função de custo adotada é a minimização do tempo total de processamento, obedecendo às restrições de oferta de recurso e prazos de entrega.

A tabela 3.1 apresenta os índices utilizados nas equações utilizadas no modelamento do problema.

Tabela 3.1. Índices utilizados

i	operação
b	batelada
j	processador
s	estado
t	instante de tempo (*slot*)
u	recursos utilizados

3.4.5.1 Equação de Balanço de Massa

Uma modificação nas equações iniciais propostas por Kondili *et al.* (1993) diz respeito ao balanço de massa. Originalmente o tamanho das bateladas era determinado pelo método de otimização, e era indexado em operação, processador e instante de início. Como nesse trabalho estão sendo consideradas bateladas de tamanho fixo e pré determinado, e a alocação é feita seguindo uma seqüência de bateladas de operação, não há a necessidade da indexação da batelada pelo instante de tempo.

Após a alocação de cada batelada é feita a atualização da quantidade de massa dos estados gerados e consumidos nessa operação, o que é feito por duas equações.

$$S(s,t) = S(s,t) + \sum_{i \in OGs} Ro\_g_{(i,s)} * M(i,b), \quad \text{para } \forall\, s \text{ e } t + TP(j,i) \leq t \leq fim\_horiz \quad [3.8.a]$$

$$S(s,t) = S(s,t) - \sum_{i \in OCs} Ro\_c_{(i,s)} * M(i,b), \quad \text{para } \forall\, s \text{ e } ini\_horiz \leq t \leq fim\_horiz \quad [3.8.b]$$

A equação [3.8.a] atualiza a quantidade de massa dos estados gerados pela operação **i**, à partir do seu instante de término, dado pelo instante de alocação **t** mais tempo de processamento **TP(j,i)**, até o final do horizonte de alocação. A massa dos estados consumidos pela operação é subtraida da quantidade existente ao longo de todo o horizonte de alocação, por meio da equação [3.8.b]. Esse método torna mais simples o procedimento de alocação, bastando verificar se existe a quantidade necessária de um determinado estado no instante em que se pretende alocar a operação, caso contrário, a tentativa passa ao instante t+1 até que a massa necessária esteja disponível ou seja alcançado o fim da janela de operação. A tabela 3.2 fonece a nomenclatura da equação 3.8.

Tabela 3.2. Nomenclatura da equação [3.8]

Variável das equações:	
S(s,t)	Estoque do estado **s** no instante de tempo **t**
Dados das equações:	
M(i,b)	Quantidade de massa da batelada **b** da operação **i**
Ro_g(i,s)	Proporção (fração mássica) do estado **s** na saída da operação **i**
Ro_c(i,s)	Proporção (fração mássica) do estado **s** na entrada da operação **i**
TP(j,i)	Tempo de processamento da operação **i** no processador **j**
Conjuntos utilizados na limitação do domínio das equações:	
OG_s	Conjunto de operações gerando o estado **s**
OC_s	Conjunto de operações consumindo o estado **s**
ini_horiz	Instante de início do horizonte de alocação
fim_horiz	Instante de término do horizonte de alocação

3.4.5.2 Não Coexistência

A discretização pré definida do tempo utilizada por alguns autores (Kondili *et al.*(1993), Shah (1993) e Grossmann (1992)) considera um discriminante **W(j,i,t)** binário, que garante a não coexistência de operações no mesmo processador em cada instante de tempo **t'**. Isto

significa que, se a operação **i** inicia o seu processamento em **j** no instante **t**, outra operação **i'** só poderá ser iniciada em **j** a partir do instante de tempo **t+TP(j,i)**, que pode ser expresso como:

$$\sum_{i \in Ij} \sum_{t'=t}^{t-TP(j,i)+1} W(j,i,t') \leq 1, \quad \text{para } \forall\, j,\ t \in H \quad [3.9]$$

Durante o sequenciamento, as bateladas alocadas vão sendo eliminadas da seqüência candidata, não havendo a possibilidade de alocação da mesma batelada mais de uma vez, e a ocupação dos processadores pode ser indicada por uma variável binária **W(j,t)**. Se uma batelada da operação **i** é iniciada no instante **t** ocupando o processador **j** durante **TP(j,i)** intervalos de tempo, então **W(j,t)** assume o valor 1 (ocupado) durante o intervalo (t,t+TP(j,i)-1). Uma vez definido o instante de alocação, o método analisa o valor de W e caso encontre o valor 1, toma outro instante de alocação dentro da janela de operação e repete o procedimento.

Tabela 3.3. Nomenclatura da equação [3.9]

Variável da equação:	
W(j,i,t)	= 1, indica o início do processamento da operação **i** no processador **j** no instante de tempo **t**; = 0, Caso contrário
Dado da equação:	
TP(j,i)	Tempo de processamento da operação **i** no processador **j**
Conjuntos utilizados na limitação do domínio da equação:	
H	Representa o horizonte de planejamento da produção, conjunto de todos os instantes de tempo
$\mathbf{I_j}$	Conjunto de tarefas que podem ser processadas no processador **j**

3.4.5.3 Consumo de Recursos Compartilhados

As operações a serem realizadas podem consumir recursos compartilhados como processadores, energia elétrica, vapor, mão-de-obra, além do tempo. Este consumo de recursos pode ser fixo para cada operação ou pode ser variável proporcionalmente ao tamanho

da batelada ou durante o processamento. A oferta desses recursos é limitada, constituindo uma restrição no processo de alocação. O método utilizado foi a avaliação da quantidade necessária, verificação de disponibilidade de recurso e débito sobre a oferta durante o período de processamento de cada operação, tal como o consumo de massa.

$$U_{(u,t')} = U_{(u,t')} + C\_fix_{(u,i)} + C\_var_{(u,i)} * M_{(i,b)}, \text{ para } \forall\, u,\ t \le t' \le t + TP(j,i) - 1 \in H \quad [3.10.a]$$

$$U^{min}_{(u,t')} \le U_{(u,t')} \le U^{max}_{(u,t')}, \text{ para } \forall\, u,\ t \le t' \le t + TP(j,i) - 1 \in H \quad [3.10.b]$$

A equação [3.10.a] avalia o consumo de utilidade **u** dentro do intervalo de processamento (t+TP(j,i)-1) da batelada **b** da operação **i**, que é constituído pelo consumo fixo **C_fix(u,i)** e pelo consumo de recurso proporcional ao tamanho da batelada **C_var(u,i)**. A quantia U(u,t) de recurso requerida deve estar dentro dos limites de oferta dados pela equação [3.10.b], caso esta restrição não seja obedecida, procede-se da mesma forma utilizada para o consumo de massa, buscando um instante de alocação que obedeça às restrições.

Tabela 3.4. Nomenclatura da equação [3.10]

Variável das equações:	
U(u,t)	Consumo de recurso **u** no instante de tempo **t**
Dados das equações:	
M(i,b)	Quantidade de massa da batelada **b** da operação **i**
C_fix(u,i)	Consumo fixo de recurso **u** pela operação **i**
C_var(u,i)	Consumo variável de recurso **u** pela operação **i**
$U^{min}_{(u,t)}$	Consumo mínimo do recurso **u** diponível no instante **t**
$U^{max}_{(u,t)}$	Disponibilidade máxima do recurso **u** no instante **t**

4. Estudos de Caso

4.1 Introdução

A literatura não traz muitos exemplos de aplicação do método *Simulated Annealing* voltados a problemas de programação de produção, restringindo-se ao clássico problema do caixeiro viajante e alguns problemas de sequenciamento de operações com poucas restrições. Neste trabalho foi feito um estudo da aplicação do método SA a problemas de sequenciamento de operações de plantas multiproduto e multipropósito, com limite de oferta de recursos, balanço de massa e política de estocagem intermediária mista.

A aplicabilidade do método e o desempenho obtido com cada combinação de parâmetros são demonstrados por quatro problemas escolhidos, tendo sido três deles já tratados por outros autores através de outros métodos, afim de se estabelecer critérios de comparação.

Para todos eles foi utilizado um programa em código Fortran 90, com critério de parada por limite de tempo, de acordo com as características do problema. As novas seqüências candidatas foram obtidas por uma subrotina de geração de números pseudo-randômicos, desta forma, as posições de troca de elementos foram sempre as mesmas, ficando o desempenho à cargo de cada combinação de parâmetros, aceitando ou não a nova seqüência.

A abordagem em nível hierárquico foi adotada, em que as atribuições de processadores, janelas de tempo e disponibilidade de recursos são pré-definidos em um nível superior de Planejamento de Produção, restando ao algoritmo apenas a tarefa de alocação temporal das operações, a Programação de Produção propriamente dita.

A obtenção da seqüência inicial foi feita através do ordenamento por criticalidade de operação decrescente, sendo igual para todas as combinações de parâmetros de um mesmo problema. O desempenho do SA foi também confrontado com resultados obtidos por outro programa que gera e simula a alocação de seqüências randômicas sem o algoritmo de escolha, encontrando as soluções por acaso. As seções seguintes descrevem os detalhes, considerações e resultados de cada problema tratado.

4.2 Aplicação em Plantas Multiproduto

As plantas multiproduto são aquelas em que o fluxo material é unidirecional, a seqüência de utilização dos processadores é a mesma para os diversos produtos, cujas características pouco diferem de um para outro, contudo podendo exigir diferentes tempos de processamento ou consumo de recursos. Este é o tipo de problema mais simples, uma vez que não haja muitas restrições quanto à oferta desses recursos ou alta criticalidades para as operações, sendo freqüentemente utilizado o menor tempo global de processamento para todas as operações como critério de desempenho.

4.2.1 Problema Tratado por Campos (1993)

O problema aqui apresentado foi inicialmente tratado por Campos (1993) aplicando o método de busca *Branch and Bound*. Consiste em sequenciar quatro tarefas (N=4) em três processadores (M=3), utilizando um recurso compartilhado com disponibilidade de 10 unidades por unidade de tempo (ur/ut), não havendo restrições quanto a janelas de alocação. A seqüência inicial obtida corresponde à nomenclatura das operações, assumindo uma posição na seqüência da mesma forma em que é acessada do arquivo de entrada.

Para uma oferta ilimitada de recursos, a melhor solução que pode ser encontrada por uma busca exaustiva, corresponde a 47 unidades de tempo. O método *Branch and Bound* encontrou uma solução igual a 72 unidades de tempo para uma oferta de recursos igual a 10 ur/ut e apenas com o uso de uma relação de recorrência o autor encontrou a solução ótima correspondente a 61 unidades de tempo.

O *Simulated Annealing*, através da simulação da alocação das operações encontrou esse ótimo obedecendo a restrição de oferta de recurso para a maioria das combinações de parâmetros. Os tempos de processamento e consumo de recursos para cada tarefa são apresentados na tabela 4.1.

Tabela 4.1. Tempos de processamento e consumo de recursos

Tarefa	Tempo de processamento por processador			Consumo de recursos por unidade de tempo		
	Proc. 1	Proc. 2	Proc. 3	Proc. 1	Proc. 2	Proc. 3
1	8	9	5	8	6	4
2	6	7	7	6	2	4
3	7	7	12	7	3	6
4	4	7	12	3	4	8

Neste caso, o objetivo é encontrar uma seqüência que minimize o tempo total de processamento, obedecendo a oferta de recursos. Como critério de parada foi utilizado o tempo limite de 120 segundos.

O pequeno número de restrições impostas a esse problema torna grande o espaço de busca, favorecendo os métodos de rearranjo que geram novas seqüências a partir de grandes perturbações da seqüência atual, como a permutação de elementos não adjacentes. E uma vez que o algoritmo tenha explorado suficientemente o espaço de busca, a intensificação da mesma é favorecida, como é o caso do Algoritmo de Metrópolis. Já o método de recozimento não demonstra grande influência para este caso. A tabela 4.2 apresenta o tempo para a obtenção da melhor solução, a porcentagem de tempo utilizada pelo processo de alocação de seqüências factíveis, a exploração do espaço de busca pelo número de soluções analisadas e a eficiência da combinação pela melhor solução encontrada.

Tabela 4.2. Comparação de desempenho

		Glauber				Metropolis			
Recozimento	***Rearranjo***	**Tempo minimo (s)**	**% para alocação**	**Soluções analisadas**	**Melhor solução**	**Tempo minimo (s)**	**% para alocação**	**Soluções analisadas**	**Melhor solução**
Exponencial	***Inserção e deslocamento***	0,05	0,18	10	61	0,06	0,23	9	61
	Permutação	0,05	0,13	5	67	0,05	0,22	8	61
	Permutação de adjacentes	0,06	0,04	3	71	0,28	0,32	10	61
Aarts & van Laarhoven	***Inserção e deslocamento***	0,06	0,18	10	61	0,11	0,23	9	61
	Permutação	0,11	0,14	5	67	0,05	0,22	8	61
	Permutação de adjacentes	0,05	0,04	3	71	0,33	0,32	10	61
Lundy e Mees	***Inserção e deslocamento***	0,06	0,18	10	61	0,06	0,27	9	61
	Permutação	0,06	0,10	5	67	0,05	0,22	8	61
	Permutação de adjacentes	0,05	0,04	3	71	0,27	0,27	10	61
Média		0,06	0,11		66,33	0,14	0,26		61

A eficiência do SA foi também analisada através da solução do mesmo problema com as mesmas técnicas de construção de novas seqüências, retendo a melhor solução encontrada, porém gerando novas seqüências randomicamente à partir da anterior.

O desempenho obtido desta forma é comparável ao uso do algoritmo de Glauber com um pequeno acréscimo de tempo para encontrar a melhor solução, caracterizando a necessidade de um meio de intensificar a busca e a repetição de visitas a soluções factíveis, caracterizando o descontrole do método.

Tabela 4.3. Desempenho do método randômico

	Método Randômico			
Rearranjo	**Tempo minimo (s)**	**% para alocação**	**Soluções analisadas**	**Melhor solução**
Inserção e deslocamento	0,06	0,37	18	61
Permutação	0,11	0,37	16	61
Permutação de adjacentes	0,11	0,13	8	71

As figuras 4.1.a e 4.2.a representam a Carta de Gantt e o perfil de consumo para uma oferta ilimitada de recursos, enquanto as figuras 4.1.b e 4.2.b, foram obtidas para uma oferta de 10 ur/ut.

FIGURA 4.1.A: CARTA DE GANTT PARA UMA OFERTA ILIMITADA DE RECURSOS

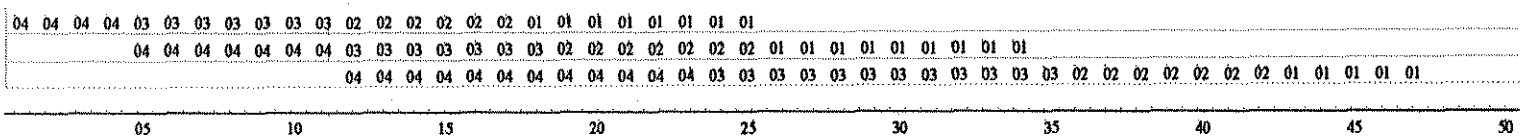
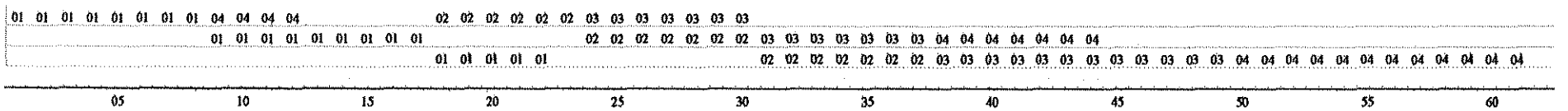


FIGURA 4.1.B: CARTA DE GANTT PARA UMA OFERTA DE 10 UR/UT



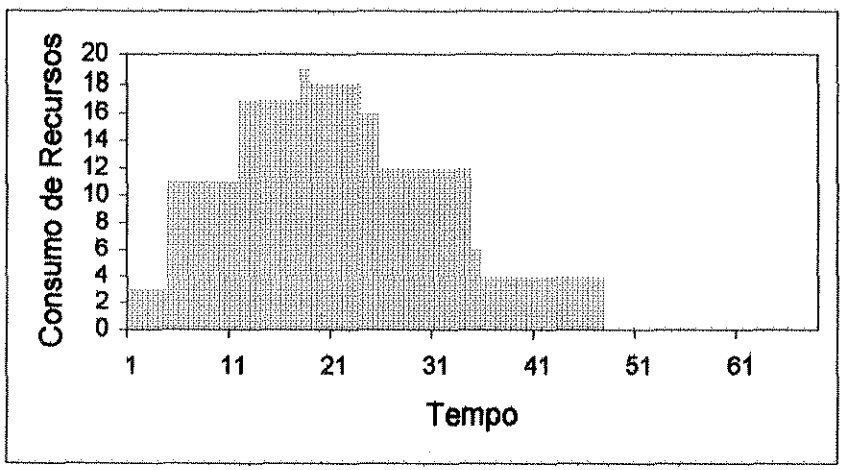


Figura 4.2.a Perfil de consumo de recursos para oferta ilimitada

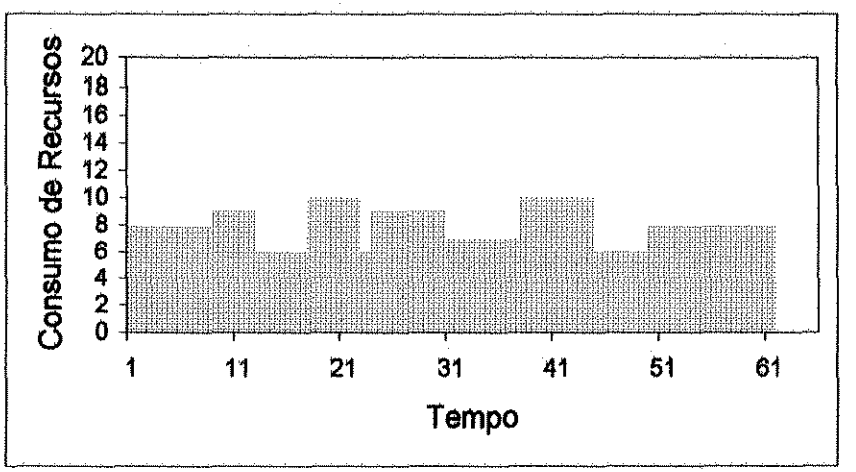


Figura 4.2.b Perfil de consumo de recursos para oferta de 10 ur/ut

4.2.2 Problema Tratado por Gimeno *et al.* (1993)

Este problema, apresentado por Gimeno *et al.* (1993), foi utilizado para demonstrar o uso de regras de despacho sobre um horizonte rolante de alocação consistindo em um problema de seqüências de permutação com grandes restrições quanto a oferta de recursos e a duração das janelas de alocação.

Estas restrições sobre o sequenciamento de oito tarefas (N=8) em três processadores (M=3) foram mantidas para analisar o comportamento do SA para tais situações. A sua representação STN é dada pela Figura 4.3. A oferta de recursos compartilhados é limitada a 10 ur/ut e os tempos de processamento e consumo de recursos são dados pela tabela 4.4.

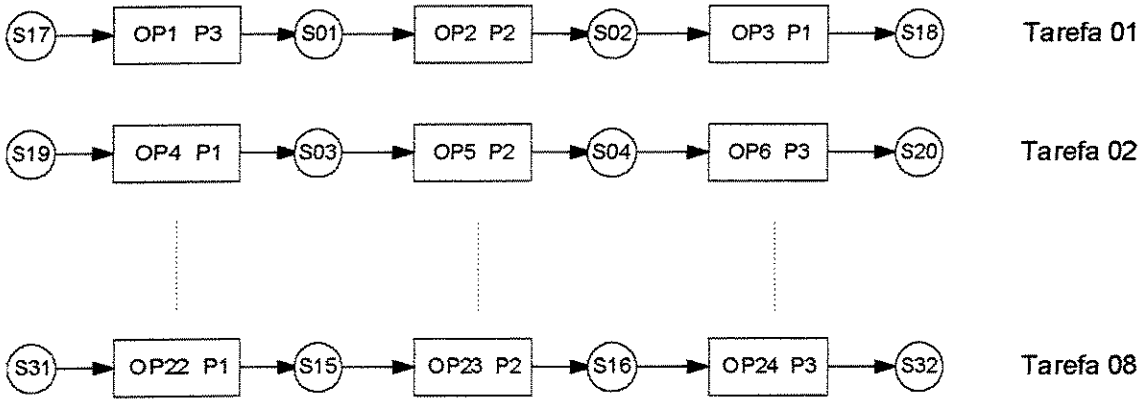


FIGURA 4.3: REPRESENTAÇÃO STN DO PROBLEMA TRATADO POR GIMENO.

Tabela 4.4. Tempos de processamento e consumo de recursos

Tarefa	Tempo de processamento por processador			Consumo de recursos por unidade de tempo		
	Proc. 1	Proc. 2	Proc. 3	Proc. 1	Proc. 2	Proc. 3
1	3	7	2	6	3	4
2	5	7	8	5	6	3
3	8	5	6	4	6	2
4	5	7	3	6	4	2
5	10	4	6	9	2	5
6	6	9	3	7	7	4
7	9	2	5	3	6	3
8	4	4	6	5	6	2

Os limites das janelas de alocação, dados pelo instante mais cedo em que a tarefa pode ser iniciada (EBT) e o instante mais tarde que pode ser finalizada (LFT), são dados pela tabela 4.5.

Tabela 4.5. Janelas de tempo para as tarefas

Tarefa	*EBT*	*LFT*	Tarefa	*EBT*	*LFT*
1	31	61	5	1	41
2	11	36	6	1	76
3	20	51	7	1	81
4	1	61	8	11	51

Neste caso, as operações possuem grande criticalidade e a maior dificuldade encontrada foi a geração de uma seqüência candidata, uma vez que, com uma oferta de recursos limitada a 10 ur/ut, foram encontradas apenas duas soluções factíveis.

A análise de desempenho do método seguiu os mesmos critérios de avaliação anteriores, com um tempo limite de 120 segundos, para a geração da seqüência inicial, partiu-se de uma lista de operações com criticalidades decrescentes, alterando-a até que fosse encontrada alguma factível, que neste caso, já era uma boa solução. Todas as soluções que foram obtidas pelas combinações desses parâmetros são iguais, com um tempo total de processamento de 76 unidades de tempo, um consumo de recursos de 10 ur/ut e duas seqüências encontradas.

Devido às restrições do problema, 99,67 % do tempo total de processamento foi utilizado para a geração e verificação de novas seqüências, enquanto o processo de alocação consumiu apenas uma média de 0,32 % do tempo total. O método de geração de novas seqüências tem fator decisivo em problemas deste tipo, que pode ser melhorado pelo uso da lista de seqüências já visitadas. Nenhuma combinação de parâmetros encontrou alguma solução factível construindo novas soluções por meio de troca de elementos adjacentes que promove uma pequena diversificação na busca. A tabela 4.6 mostra uma comparação dos resultados, apresentando tempo mínimo para a obtenção da melhor solução e a porcentagem de tempo utilizada pelo processo de alocação de seqüências factíveis.

Tabela 4.6. Comparação de desempenho

		Glauber		Metropolis	
Recozimento	***Rearranjo***	**Tempo mínimo (s)**	**% para alocação**	**Tempo mínimo (s)**	**% para alocação**
Exponencial	***Inserção e deslocamento***	0,06	0,37	0,11	0,37
	Permutação	49,54	0,23	46,69	0,32
	Permutação de adjacentes	> 120	-	> 120	-
Aarts & van Laarhoven	***Inserção e deslocamento***	0,11	0,32	0,06	0,41
	Permutação	46,86	0,27	47,07	0,32
	Permutação de adjacentes	> 120	-	> 120	-
Lundy e Mees	***Inserção e deslocamento***	0,06	0,32	0,11	0,55
	Permutação	46,90	0,27	47,35	0,18
	Permutação de adjacentes	> 120	-	> 120	-
Média		47,83	0,29	47,91	0,12

Neste caso, não fica claro qual algoritmo é mais eficiente, pois a solução depende muito da seqüência inicial e o espaço de busca é extremamente restrito beneficiando os rearranjos de Permutação e o de Inserção e deslocamento.

O desempenho oferecido pelo mesmo método randômico demonstra que o método heurístico é pouco eficiente para problemas dessa natureza, com fortes restrições; tendo encontrado as mesmas soluções com uma pequena diminuição do tempo requerido pelo SA, o que é demonstrado na tabela 4.7.

Tabela 4.7. Desempenho do método randômico

	Método Randômico		
Rearranjo	**Tempo mínimo (s)**	**% para alocação**	**Melhor solução**
Inserção e deslocamento	0,05	0,32	76
Permutação	40,65	0,22	76
Permutação de adjacentes	>120	0,04	-

As figuras 4.4.a e 4.5.a representam a Carta de Gantt e o perfil de consumo para uma oferta ilimitada de recursos, enquanto as figuras 4.4.b e 4.5.b, foram obtidas para uma oferta de 10 ur/ut.

Figura 4.4.a: Carta de Gantt para uma oferta ilimitada de recursos

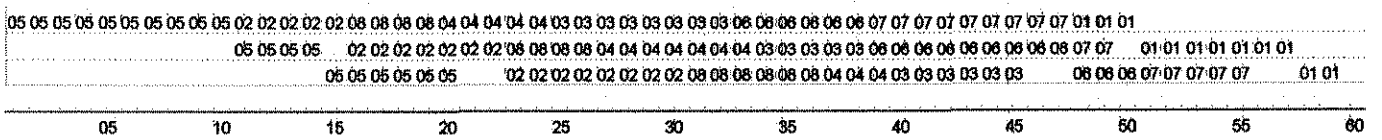
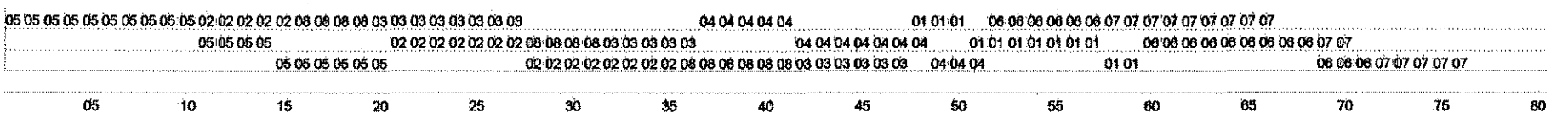


FIGURA 4.4.B: CARTA DE GANTT PARA UMA OFERTA DE 10 UR/UT



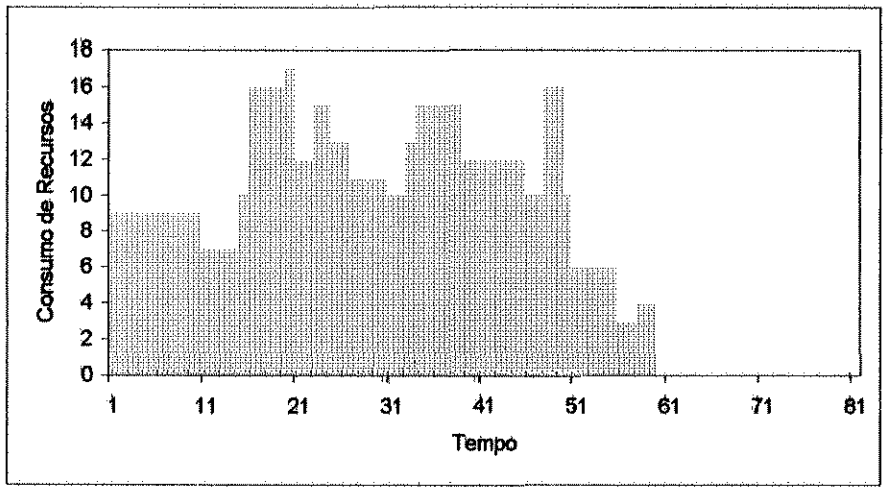


FIGURA 4.5.A PERFIL DE CONSUMO DE RECURSOS PARA OFERTA ILIMITADA

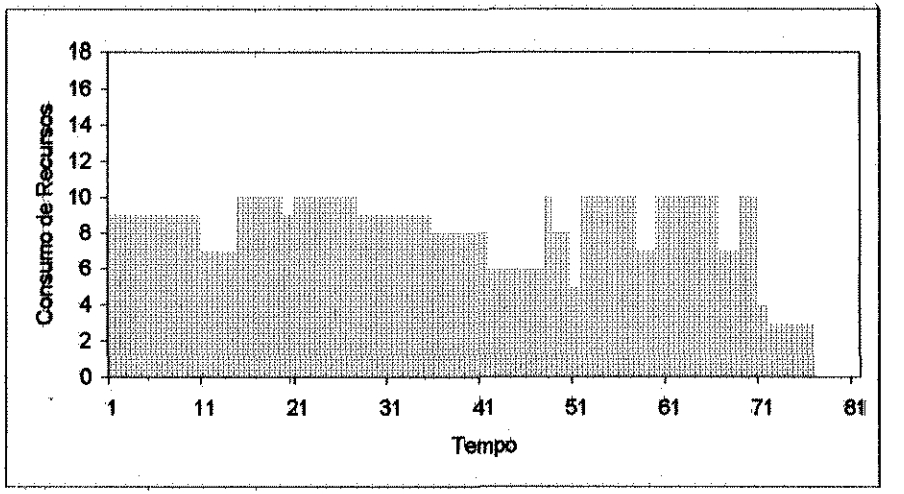


FIGURA 4.5.B PERFIL DE CONSUMO DE RECURSOS PARA OFERTA DE 10 UR/UT

4.3 Aplicação em Plantas Multipropósito

Plantas multipropósito não possuem fluxo de material unidirecional, suas unidades ou processadores possuem muitas interconexões, o que restringe o seu uso. É empregada na produção de produtos de mesmo gênero, porém pouco similares entre si. Por este motivo, a alocação de operações para este tipo de planta também é mais complexa.

Neste caso, também foi utilizado o menor tempo global de processamento para todas as operações como critério de desempenho.

4.3.1 Problema tratado por Pantelides (1996)

Este problema foi apresentado por Pantelides *et al.* (1996) que propuseram uma aproximação por decomposição matemática de problemas de *scheduling* para a otimização de campanhas com processamento em bateladas. A sua representação STN é apresentada na figura 4.6.

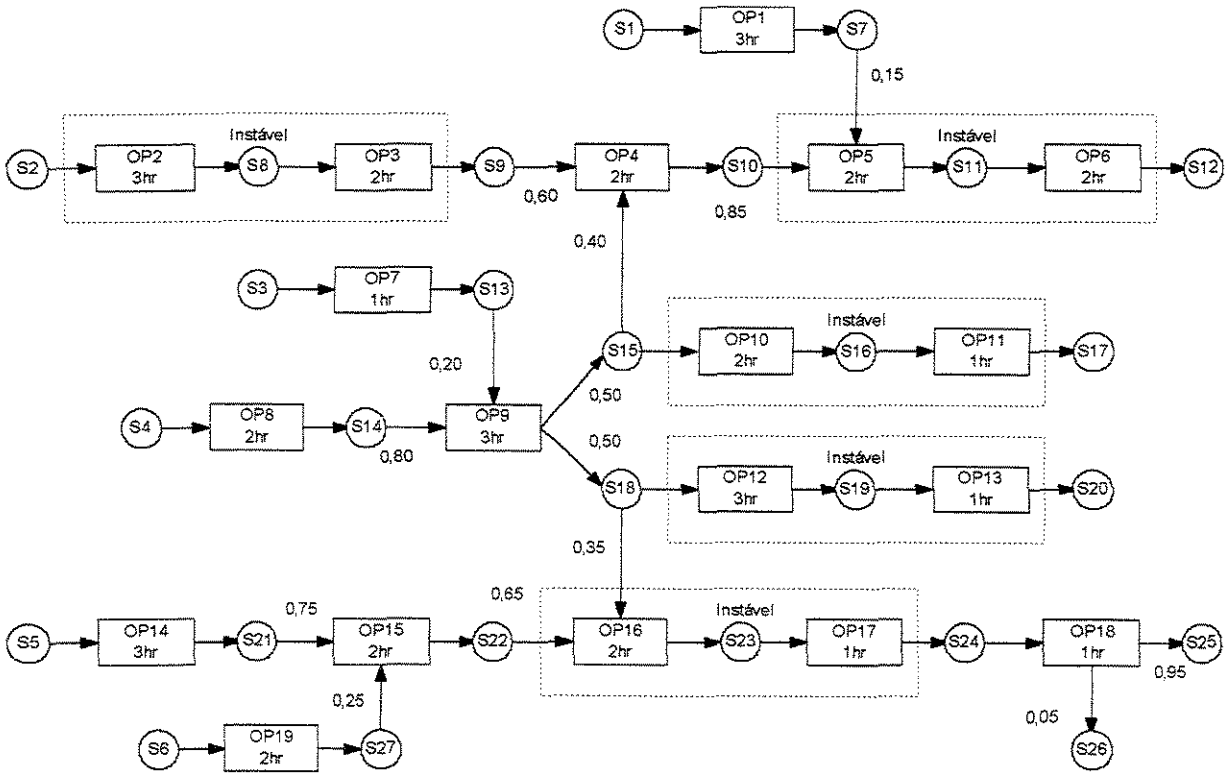


FIGURA 4.6. REPRESENTAÇÃO STN DO PROBLEMA TRATADO POR PANTELIDES (1996)

O problema consiste em atender a demanda final de quatro produtos, (S12, S17, S20 e S25), utilizando para isso oito processadores. A política de estocagem é mista, com estocagem intermediária infinita para alguns estados e do tipo sem espera para os intermediários instáveis (S8, S9, S11, S16, S19 e S23). Portanto, a operação que gera esses últimos só pode ser alocada se ao seu término o processador seguinte estiver disponível e com recursos em quantidade suficiente, caso contrário, a operação é atrasada até que as restrições sejam atendidas.

É assumido que um nível superior de planejamento determinou os tempos de processamento e a habilitação de processadores, que são apresentados na tabela 4.8, enquanto a demanda de produtos finais e plano de compra de matéria prima são apresentados na tabela 4.9. As operações podem exigir mais de uma batelada da operação anterior e essas informações são apresentados na tabela 4.10

Tabela 4.8. Capacidade e habilitação de processadores

Processador	Capacidade	Operações	Processador	Capacidade	Operações
P1	6	OP2.OP4	P5	8	OP1.OP10.OP12
P2	8	OP9	P6	6	OP14.OP16
P3	7	OP8	P7	7	OP3.OP5.OP11
P4	7	OP6.OP7.OP19	P8	8	OP13.OP15.OP17.OP18

Tabela 4.9. Demanda de produtos finais e plano de compra de matéria prima.

Demanda de produtos finais					Plano de compra de MP				
Estado	kg	Prazo de entrega	kg	Prazo de entrega	Estado	kg	Instante	kg	Instante
S12	10	20	10	23	S1	10	1	-	-
S17	10	27	10	32	S2	10	1	10	7
S20	5	27	5	32	S3	20	1	-	-
S25	5	30	5	35	S4	60	1	-	-
					S5	20	1	-	-
					S6	10	1	-	-

Tabela 4.10. Número e massa de bateladas para suprir a demanda.

Seqüência	Nº bateladas	kg	Seqüência	Nº bateladas	kg
T1	1	8	T10,T11	3	7
T2,T3	2	6	T12,T13	2	8
T4	3	6	T14	2	6
T5,T6	3	7	T15	2	8
T7	2	7	T16,T17	3	6
T8	8	7	T18	2	8
T9	8	8	T19	1	7

Uma das simplificações deste problema é a oferta ilimitada de recursos, o que aumenta o número de soluções que satisfazem as restrições, por esse motivo, o tempo limite de busca foi reduzido para 30 segundos obtendo por volta de 360 iterações por combinação de parâmetros. O critério de avaliação adotado é a minimização do tempo total de processamento e o tempo mínimo necessário para encontrar a melhor solução. A comparação de desempenho é apresentada na tabela 4.11, com o tempo mínimo para se obter a melhor solução, a porcentagem de tempo computacional para a alocação das operações e a melhor solução encontrada.

Tabela 4.11. Comparação de desempenho

		Glauber			Metropolis		
Recozimento	***Rearranjo***	**Tempo mínimo (s)**	**% para alocação**	**Melhor solução**	**Tempo mínimo (s)**	**% para alocação**	**Melhor solução**
Exponencial	***Inserção e deslocamento***	> 30	-	-	> 30	-	-
	Permutação	0,55	51,73	29	0,55	50,98	29
	Permutação de adjacentes	0	63,61	32	0,77	66,02	29
Aarts & van Laarhoven	***Inserção e deslocamento***	> 30	-	-	> 30	-	-
	Permutação	0,71	47,90	29	0,50	48,03	29
	Permutação de adjacentes	0,06	62,18	32	0,60	66,11	29
Lundy e Mees	***Inserção e deslocamento***	> 30	-	-	> 30	-	-
	Permutação	0,61	50,71	29	0,49	49,93	29
	Permutação de adjacentes	0	64,58	32	0,77	65,24	29
Média		0,32	56,79	30,5	0,61	57,72	29

Este problema possui um grande espaço de busca, existindo muitas soluções factíveis que aumentam o tempo requerido para sua alocação. A construção da seqüência inicial também foi favorecida pelo sequenciamento em ordem decrescente de criticalidade de operações, exigindo pequenas alterações em sua configuração, encontrando a melhor solução em pouco tempo, o que é prejudicado pelo rearranjo de Inserção e Deslocamento. Deste ponto em diante, o melhor caminho é a intensificação de busca obtida pelo algoritmo de Metrópolis.

As figuras 4.7.a e 4.7.b apresentam cartas de Gantt para duas soluções encontradas dispondo de uma oferta de recursos ilimitada e com uma política de estocagem mista.

FIGURA 4.7.A: CARTA DE GANTT PARA UMA OFERTA ILIMITADA DE RECURSOS

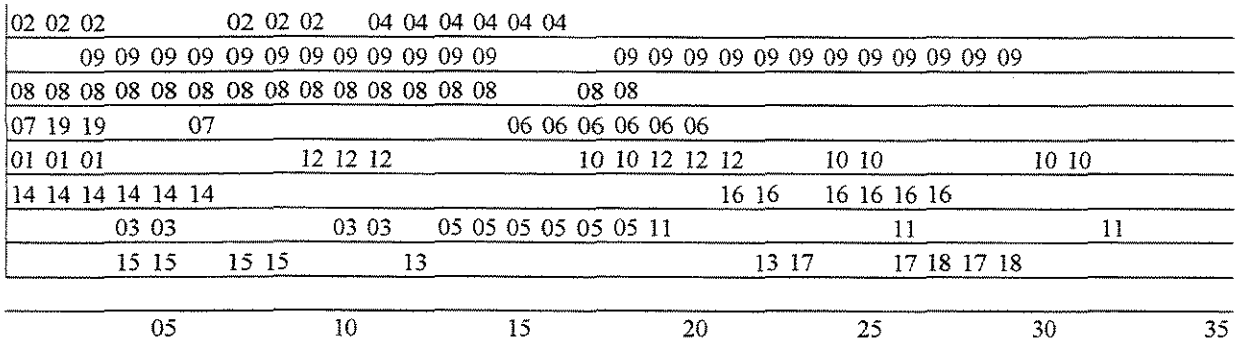
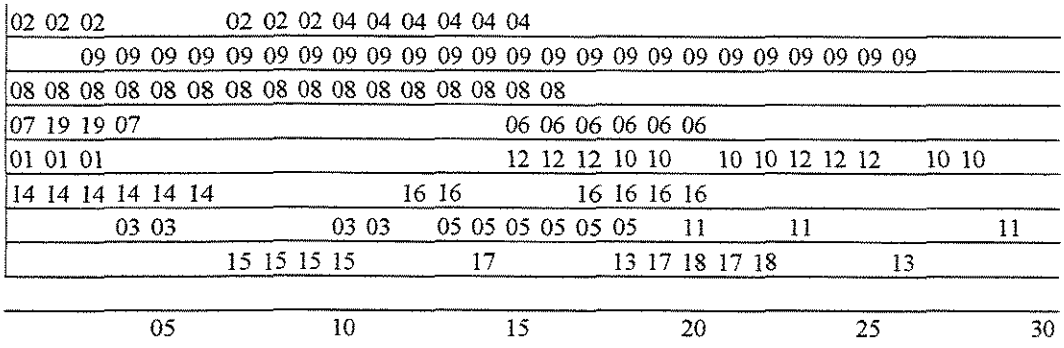


FIGURA 4.7.B: CARTA DE GANTT DA MELHOR SOLUÇÃO OBTIDA



4.3.1 Problema Jobshop 016

Esse problema foi elaborado para testar o método utilizando diferentes tamanhos de batelada da mesma operação, desta forma a alocação de uma batelada pequena pode ser insuficiente para executar a próxima operação, enquanto uma batelada grande pode atender a demanda de mais de uma operação seguinte. Neste caso as janelas de alocação adotadas foram em relação à operação e não em relação à cada batelada. Os demais critérios de construção de seqüência inicial, critério de parada e análise de desempenho foram mantidos.

A tabela 4.12 apresenta o tempo de processamento e consumo de utilidades para cada operação e a 4.13 a habilitação e capacidade de processadores e o número e tamanho de cada batelada. Foi adotada uma política de estocagem intermediária infinita, com uma oferta de 5 ur/ut dos recursos 1 e 2, e 10 ur/ut para o recurso 3.

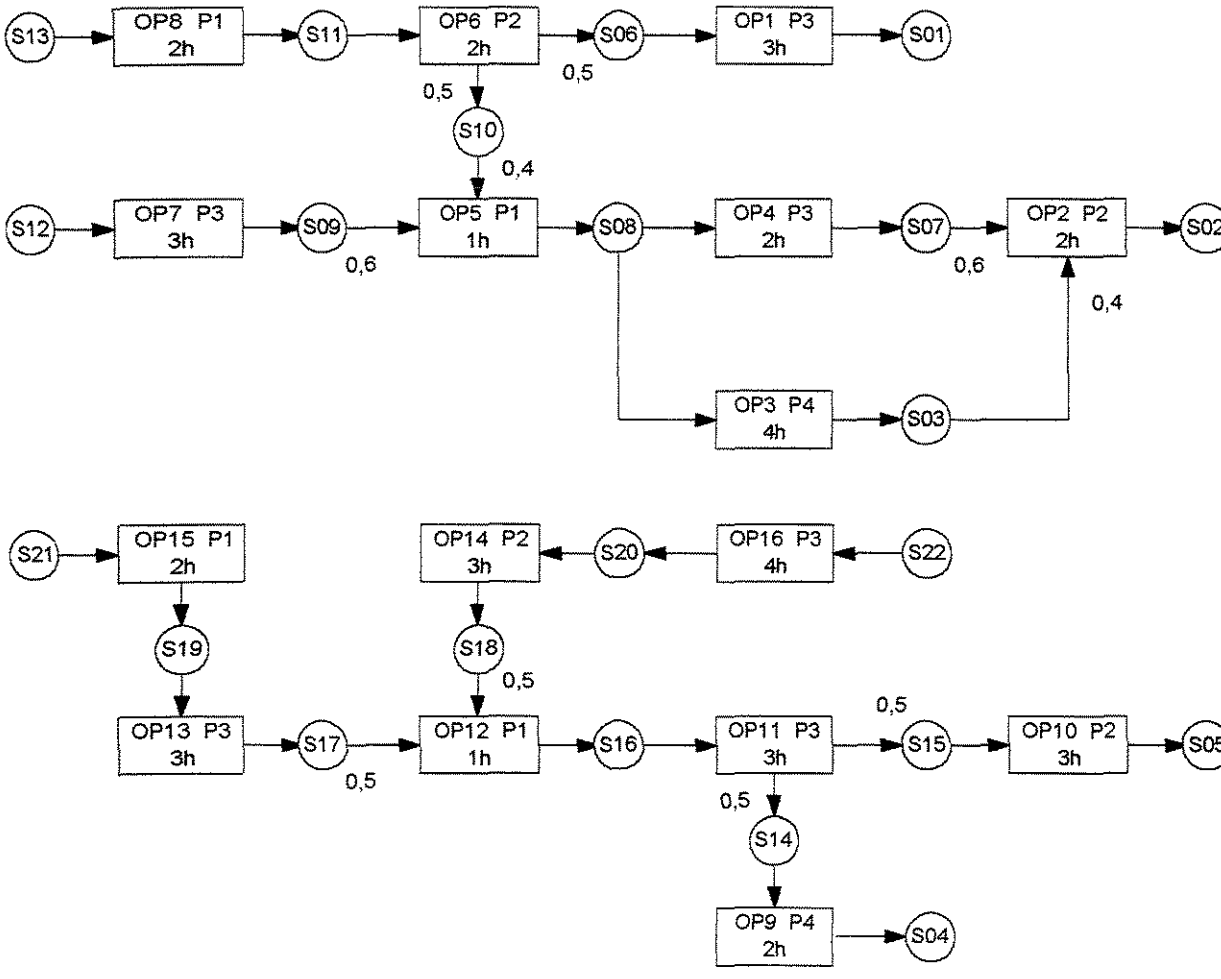


FIGURA 4.8 REPRESENTAÇÃO STN DO PROBLEMA 016

Tabela 4.12 Tempo de processamento e consumo de recursos

Operação	Tempo de processamento	Consumo de Utilidades			Operação	Tempo de processamento	Consumo de Utilidades		
		Util. (1)	Util. (2)	Util. (3)			Util. (1)	Util. (2)	Util. (3)
OP1	3	2	1	3	OP9	2	0	2	5
OP2	2	3	1	2	OP10	3	1	2	3
OP3	4	0	2	5	OP11	3	0	2	4
OP4	2	2	1	2	OP12	1	2	1	2
OP5	1	2	1	2	OP13	3	1	2	2
OP6	2	1	2	1	OP14	3	2	1	2
OP7	3	3	3	0	OP15	2	2	1	2
OP8	2	2	1	1	OP16	4	2	1	2

Tabela 4.13 Habilitação de processadores, número e tamanho de bateladas

Operação	Processador	Capacidade	Bateladas	Tamanho
OP1	P3	5	2	5
OP2	P2	4	2	4
OP3	P4	4	2	4
OP4	P3	5	1	5
OP5	P1	3	3	3
			2	2
OP6	P2	4	5	4
OP7	P3	5	2	4
OP8	P1	3	7	3
OP9	P4	4	2	4
OP10	P2	4	2	4
OP11	P3	5	1	5
			2	4
			1	3
OP12	P1	3	4	3
			2	2
OP13	P3	5	2	4
OP14	P2	4	2	4
OP15	P1	3	2	3
			1	2
OP16	P3	5	1	5
			1	3

A melhor solução obtida é apresentada como carta de Gantt na figura 4.9 e a comparação de desempenho é feita pela tabela 4.14, com tempo necessário para se obter a melhor solução, a porcentagem de tempo necessário para a alocação das seqüências e a melhor solução obtida.

FIGURA 4.9 CARTA DE GANTT DA MELHOR SOLUÇÃO ENCONTRADA

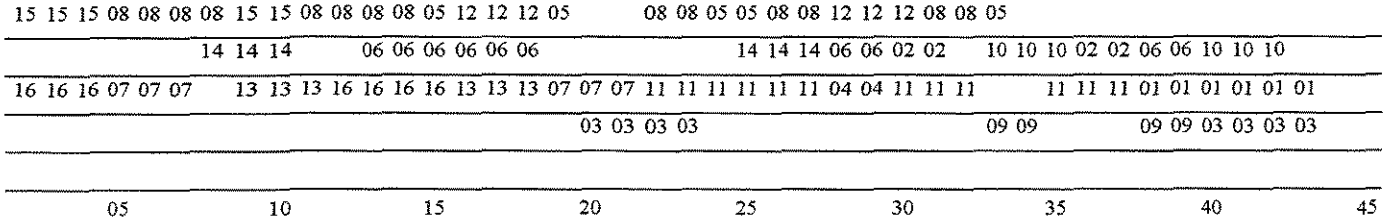
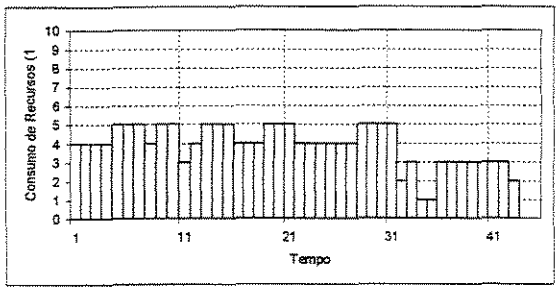
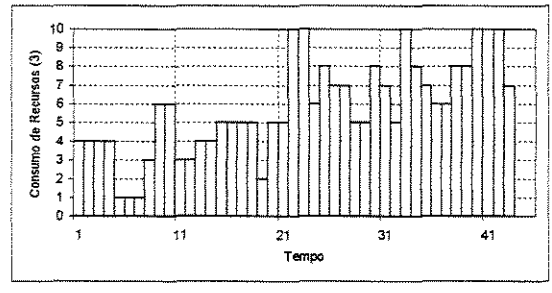


FIGURA 4.10 PERFIL DE CONSUMO DE UTILIDADES





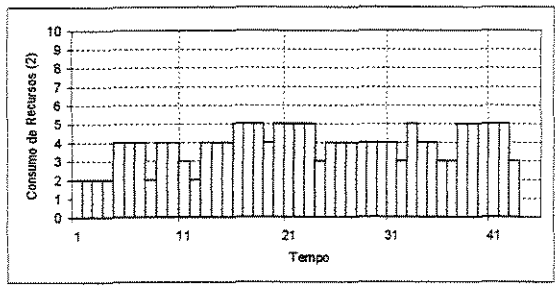


Tabela 4.14. Comparação de desempenho

		Glauber			Metropolis		
Recozimento	***Rearranjo***	**Tempo mínimo (s)**	**% para alocação**	**Melhor solução**	**Tempo mínimo (s)**	**% para alocação**	**Melhor solução**
Exponencial	***Inserção e deslocamento***	> 30	-	-	> 30	-	-
	Permutação	0,11	60,98	43	0,05	62,61	43
	Permutação de adjacentes	> 30	-	-	> 30	-	-
Aarts & van Laarhoven	***Inserção e deslocamento***	> 30	-	-	> 30	-	-
	Permutação	0,05	67,11	43	0,05	63,71	43
	Permutação de adjacentes	> 30	-	-	> 30	-	-
Lundy e Mees	***Inserção e deslocamento***	> 30	-	-	> 30	-	-
	Permutação	0	64,38	43	0,06	61,63	43
	Permutação de adjacentes	> 30	-	-	> 30	-	-
Média		0,05	64,15	43	0,05	62,65	43

O algoritmo de Metropolis apresentou uma pequena vantagem em relação ao de Glauber, mas para ambos os casos só foram encontradas soluções mediante a construção de seqüências por permutação de elementos não adjacentes que promove uma perturbação na seqüência mais intensa que os outros dois métodos.

5 Discussão e Conclusão

Neste trabalho foi proposto o uso do método heurístico *Simulated Annealing* para a resolução de problemas de Programação de Produção para plantas Multiproduto e Multipropósito, sujeitas a restrições de janelas de tempo, balanço de massa, consumo de recursos compartilhados e política de estocagem intermediária.

Trata-se de uma coletânea das variações do método quanto aos algoritmos propostos e métodos de recozimento ou ajuste do parâmetro de controle além da análise de comportamento das formas de construção novas seqüências, tão importantes quanto o algoritmo de aceitação.

Tal como nos problemas reais, não foi possível a abordagem de todas as restrições existentes, o que se consegue é a personalização de um método para cada situação encontrada. Neste trabalho, foram reunidas algumas das restrições mais encontradas na literatura, agregadas na medida em que se obtinha sucesso nas propostas anteriores, contudo sem a pretensão de solucionar todos os problemas. Desta forma, foram abordados problemas como plantas Multipropósito, que ainda possuem pouco material sobre o assunto além do problema de balanço de massa, resultando no problema de não apenas determinar quando produzir, mas também em qual quantidade produzir.

Muitas simplificações foram feitas em relação ao trabalho de outros autores, por exemplo, não foi considerado o tempo de preparação dependente da seqüência, o mesmo foi considerado fixo e incluído no tempo de processamento. Outra simplificação da abordagem multi nível é a definição dos tamanhos de batelada e habilitação de processadores por um nível superior de Planejamento, reduzindo o número de variáveis a serem otimizadas.

A linguagem de programação Fortran foi escolhida quanto à sua simplicidade e por já existir uma grande biblioteca de subrotinas e neste caso, procurou-se usar uma subrotina de geração de números pseudo-randômicos, promovendo as mesmas alterações nas seqüências de operações para todas as combinações de parâmetros, afim de se testar o desempenho de cada uma.

Os problemas tratados foram adaptados dos trabalhos de autores que utilizaram outras técnicas tais como *Branch and Bound* ou Regras de Despacho, afim de se ter uma forma de comparação, sem contudo, questionar o desempenho desses métodos.

Dentre as diversas dificuldades encontradas, a maior delas é a construção da seqüência inicial, o SA necessita de uma solução para que possa melhora-la. Em alguns casos isso é muito difícil, a solução inicial já é um ótimo obtida com o relaxamento de restrições e sucessivas buscas e realimentação do algoritmo.

A construção de seqüências de operações à partir de criticalidades decrescentes contribui para solucionar esse problema, mas deve-se tomar o cuidado com a geração de novas seqüências permitindo que uma operação de alta criticalidade seja alocada no final de sua janela de tempo, do contrário, se a mesma for fixada a uma posição no início da seqüência, a busca pode ficar presa a um mínimo local ou mesmo não encontrar uma solução factível.

Da mesma forma, o retorno à melhor solução encontrada após um determinado número de insucessos pode lançar a busca em um círculo, devendo-se estender a largura da busca. O uso de uma lista de seqüências já analisadas também permite que estas sejam desprezadas numa segunda visita.

O tipo de rearranjo das operações nas seqüências também tem grande importância, conferindo diversificação ou intensificação na busca, a utilização de um critério desses no estágio errado da busca pode manter a busca tão desorientada como uma randômica, para se evitar este problema é necessário o conhecimento do comportamento do espaço de busca, intuitivamente ou por tentativas.

A alocação temporal já é tarefa difícil sem a questão do balanço de massa e neste sentido, um pré teste contribui para se encontrar uma solução factível, uma vez que analisa apenas a ordem das operações na seqüência, verifica se quem gera está antes de quem consome material, caso contrário a seqüência é rejeitada evitando maiores esforços.

Para os problemas propostos, o *Simulated Annealing* demonstrou resultados dentro do esperado, com destaque para o algoritmo de Metrópolis que intensifica a busca na medida que aceita todas as soluções que melhoram a função objetivo. O critério de ajuste do parâmetro de controle, recozimento, não apresentou um fator decisivo, comprovando sua característica intuitiva, ficando a critério do pesquisador qual mais lhe convém quanto a exploração do espaço ou término da busca.

Desta forma, o *Simulated Annealing* pode não ser isoladamente um método de otimização, nem tem a pretensão de o ser, mas pode ser utilizado em conjunto com métodos exatos como o *Branch and Bound* ou MILP na medida que explora de maneira rápida o espaço de busca, retendo informações e transmitindo valores limites da função objetivo para os métodos exatos e estes sim, concentrarem os esforços na busca do ótimo global.

Este trabalho se concentrou em analisar a utilização do *Simulated Annealing* em problemas de Programação de Produção, deixando-se de lado diversos refinamentos que ficam como sugestões para trabalhos futuros como a utilização em conjunto com outros métodos exatos, a abordagem de problemas de busca por restrições, para o qual o SA não se demonstrou tão bem; além de modelagem mais complexa de problemas reais, como tempo de preparação dependente da seqüência, múltipla habilitação de processadores ou estocagem intermediária limitada.

Referências Bibliográficas

Aarts, E.H.L. e Van Laarhoven, P.J.M.; "Statistical Cooling: A General Approach to Combinatorial Optimization Problems", *Philips Journal of Research*, Vol. 40, No. 4, 1985.

Applequist, G.; Samikoglu, O.; Pekny, J.; Reklaitis, G.; "Issues in the Use, Design and Evolution of Process Scheduling and Planning Systems", *ISA Transactions*, Vol. 36, No. 2, pp. 81-121, 1997.

Boctor, F.F.; "Resource-constrained Project Scheduling by Simulated Annealing", *Int. J. Prod. Res.*, Vol. 34, No. 8, pp. 2335-2351, 1996.

Campos, M.F.D.; "Sequenciamento e Alocação de Operações em Flow-shops com Restrições Sobre os Recursos Compartilhados e Sobre os Prazos de Entrega das Tarefas: Umas Abordagem de Busca Orientada por Restrições", *Tese de Mestrado*, Unicamp, Agosto 1993.

Connolly, David; "General Purpose Simulated Annealing", *J. Operational Res. Soc.*, Vol. 43, No. 5, pp. 495-505, 1992.

Corana, A.; Marchesi, M.; Martini, C. e Ridela, S.; "Minimizing Multimodal Functions of Continuous Variables with the Simulated Annealing Algorithm", *ACM Transactions on Mathematical Software*, Vol. 13, No. 3, pp 292-280, Setembro 1987.

Das, H.; Cummings, P.T. e Le Van, M.D.; "Scheduling of Serial Multiproduct Batch Processes via Simulated Annealing", *Computers chem. Engng.*, Vol. 14, No. 12, pp. 1351-1362, 1990.

Díaz, A.; Glover F.; Ghaziri, H.M.; González, J.L.; Laguna, M.; Moscato, P. e Tseng, F.T.; "Optimización Heurística y redes Neuronales", *Editorial Paraninfo S.A.*, Madri, Espanha, 1996.

Dolan, W.B.; Cummings, P.T. e Le Van, M.D.; "Heat Exchanger Network Design by Simulated Annealing", *Foundations of Computer Aided Process Operations*, CACHE Corporation, Elsevier, New York, 1987.

Douglas, J.M.; "Conceptual Design of Chemical Processes", McGraw-Hill, New York, 1988.

Egli, U.M e Rippin, D.W.; "Short-Term Scheduling for Multiproduct Batch Chemical Plants", *Computers & Chemical Engineering*, Vol. 10, No. 4, pp. 303-325, 1986.

Gimeno, L.; Campos, M.D.; Rodrigues, M.T.M. e Passos, C.A.S.; "A Dispatch Rule with Rolling Prediction Horizon for Chemical Flowshops", *Conference on Computer Integrated Manufacturing in the Process Industries*, 1993.

Glauber, Roy J.; "Time-Dependent Statistics of the Ising Model", *Journal of Mathematical Physics*, Vol.4, No. 2, Fevereiro 1963.

Grossmann, I.E.; Queseda, I.; Raman, R. e Voudouris, V.T.; "Mixed-Integer Optimization Techniques for the Design and Schduling of Batch Processes", *Dpartament of Chemical Engineering and Design Research Center,* Carnegie Mellon University, Pittsburg, USA, 1992.

Haddock J. e Mittenthal J.; "Simulation Optimization Using Simulated Annealing", *Computers ind. Engng.*, Vol.22, No. 4, pp.387-395, 1992.

Keng, N.P.; Yun, D.Y.Y. e Rossi, M.; "Interaction-Sensitive Planning System for Job-Shop Scheduling", *Expert Systems and Intelligent Manufacturing*, pp. 57-69, 1988.

Kirkpatrick, S.; Gelatt Jr., C.D.; Vecchi, M.P.; "Optimization by Simulated Annealing", *Science*, Vol. 220, No. 4598, pp. 671-680, Maio 1983.

Kondili, E.; Pantelides, C.C. e Sargent, R.W.H.; "A General Algorithm for Short-Term Scheduling of Batch Operations-I. MILP Formulation", *Computers chem. Engng.*, Vol. 17, No. 2, pp. 211-227, 1993.

Ku, H.M.; Rajagopalan, D. e Karimi, I.; "Scheduling in Batch Processes", *Chemical Engineering Progress*, pp 35-45, Agosto 1987.

Ku, H.M. e Karimi, I.A.; "Scheduling Algorithms for Serial Multiproduct Batch Processes With Tardiness Penalties", *Computers chem. Engng.*, Vol. 15, No. 5, pp. 283-286, 1991.

Lundy, M. e Mees, A.; "Convergence of an Annealing Algorithm", *Math. Prog.*, 34(1), pp. 111-124, 1986.

Medeiros, A.C.G.; "Estratégia de Simulação para a Alocação de Produtos em Plantas Multipropósito", *Tese de Mestrado*, Unicamp, Abril 1995.

Metropolis, N.; Rosenbluth, A.; Rosenbluth, M.; Teller, A. e Teller, E.; "Equation of State Calculations by fast Computing Machines", *J. Chem. Phys.*, 21, 1087-1092, 1953.

Morton, T.E. e Pentico, D.W.; "Heuristic Scheduling Systems"; *Wiley Series in Engineering & Technology Management*, USA, 1993.

Murakami, Y.; Uchiyama, H.; Hasebe, S. e Hashimoto, I.; "application of Repetitive SA Method to Scheduling Problems of Chemical Processes", *Computers chem. Engng.*, vol. 21, Suppl., pp. S1087-S1092, 1997.

Musier, R.F.H. e Evans, L.B.; "Batch Process Management", *Chemical Engineering Progress*, pp.66-77, Junho 1990.

Musser, K.L.; Dhingra, J.S. e Blankenship, G.L.; "Optimization Based Job Shop Scheduling", *Technical Research Report University of Maryland*, TR 91-9, 1991.

Osman, I.H. e Potts, C.N.; "Simulated Annealing for Permutation Flow-Shop Scheduling", *OMEGA Int. of Mgmt Sci.*, Vol. 17, No. 6, pp. 551-557, 1989.

Papageorgiou, L.G. e Pantelides, C.C.; "Optimal Campaign Planning/Scheduling of Multipurpose Batch/Semicontiu\nuous Plants. 2. A Mathematical Decompositions Approach", *Ind. Eng. Chem. Res.*, 35, pp. 510-529, 1996.

Parakrama, Ravi; "Improving Batch Chemical Processes", *The Chemical Engineer*, pp. 24-25, Setembro 1985.

Parthasarathy, S. e Rajendran, C.; "An Experimental Evaluation of Heuristics for Scheduling in a Real-life Flowshop With Sequence-dependent Setup Times of Jobs", *Int. J. Production Economics*, 49, pp. 255-263, 1997.

Pekny, J.F. e Miller, D.L.; "Exact Soluction of the No-Wait Flowshop Scheduling Problem With a Comparison to Heuristic Methods", *Computers chem. Engng.*, Vol. 15, No. 11, pp. 741-748, 1991.

Reklaitis, G.V.; "Review of Scheduling of Process Operations", *The American Inst. of Chemical Engng.*, Vol. 78, No. 214, 1982.

Rippin, D.W.T.; "Simulation of Single and Multiproduct Batch Chemical Plants for Optimal Design and Operation", *Computers and Chemical Engineering*, Vol. 7, No. 3, pp. 137-156, 1983.

Rodrigues, M.T.M.; "Sequenciamento e Alocação de Operações em Flow Shops na Indústria Química com Restrições sobre os Recursos Compartilhados: Uma Abordagem de Busca Orientada por Restrições", *Tese de Doutorado*, Unicamp, Agosto 1992.

Rodrigues, M.T.M.; Gimeno L.; Passos C.A.S. e Campos, M.D.; "Reactive Scheduling Approach for Multipurpose Batch Chemical Plants", *Computers chem. Engng.*, S20, S1215-S1226, 1996.

Shah, N.; Pantelides, C.C. e Sargent, R.W.H; "A General Algorithm for Short-Term Scheduling of Batch Operations-II. Computational Issues", *Computers chem. Engng.*, Vol. 17, No. 2, pp. 229-244, 1993.

ANEXO I

```
      Program Schedule


!     Programação de Produção de plantas Multiproduto e Multipropósito em
Batelada
!     Modelo matemático adaptado de:
!     Kondili,E., Pantelides, C.C. e Sargent, R.W.H., "A General Algorithm
 for
!           Short-Term Scheduling of Batch Operations-I. MILP Formulation
",
!           Computers chem. Engng., Vol. 17, No. 2, pp. 211-227, 1993.

!     Estratégia Simulated Annealing
!     Algoritmos de Metropolis e Glauber
!     Métodos de recozimento Exponencial, Aarts van Laarhoven e Lundy e Me
es
!     Construcao de sequencia inicial por ordem decrescente de criticalida
de
!     Teste de factibilidade por balanço material
!     Opção de realimentação de solução e compra intermediaria de materiai
s
!     Política mista de armazenagem infinita e sem estoque
!     Funcao geradora de numeros pseudo randomicos RAN3(idum)
!     Arquivos de entrada DADOS.E, JAN.E, INIC.E, UTIL.E e REALIM.E
!     Arquivos de saida  Histor_aloc.S,Histor_utilid.S,Histor_result.S e R
EALIM.S

!     DEFINICAO DE VARIAVEIS E FUNCOES
!     NO, SEQ(I) Numero e Indice para sequencia de operacoes
!     NS, EST, Numero e indice de estados
!     NJ, PROC Numero e Indice para Processadores
!     T, Tf Fatores de ajuste analogo a temperatura inicial e final
!     PA Probabilidade de aceitacao da sequencia
!     LIMI e LIMR limitantes de iteracoes e reconfiguracoes para ajuste de
 T
!     SLOT, SLOTL Intervalo de tempo
!     CONTR,CONTM e NF Contadores de reconfiguracoes, melhorias e sequenci
as rejeitadas
!     TP(processador,operacao) matriz tempo de processamento
!     MSPAN, SPAN_AT, SPAN_CAND e SPAN_BEST   Tempo total de processamento
!     N1 e N2 Posicoes na sequencia para alteração de operações
!     SEQ_BEST e SEQ_CAND Sequencias de operações
!     INTER,OP_ALG,OP_SCH,OP_ARR  Opções alimentacao intermediaria, algori
tmo,
!       recozimento e rearranjo
!     C_FIX,C_VAR   Consumo Fixo e Variavel de Utilidade

      USE PORTLIB
      USE MSFLIB

      INTEGER EST,SLOT,NO,NS,NJ,NT,I,CONTR,CONTM,LIMI,LIMR,ITER,IND1,N1,N2
,XX,U,NF,OF(3),DEL,&
              SPAN_AT,SPAN_CAND,SPAN_BEST,INTER,OP_ALG,OP_SCH,OP_ARR,REALI
,NEI,E_INST(20),larg,&
              N_BAT(30),n_max_bat,SEQ(200),SEQ_AT(200),SEQ_CAND(200),SEQ_B
EST(200),SEQ_AUX(200)

      LOGICAL(1) MEM,result

      REAL    M(30,20),AUX,alfa,DC_MAX,equil,sig_t,Tempo_lim,FIM,b,AUX_S(4
0,100)

      REAL(8) EXPR,PA,T,Tf,DELTA,tempo_corrido ,best_time,inst_01,inst_02,
tempo,tmake,tfac

      REAL,ALLOCATABLE::C_FIX(:,:),C_VAR(:,:)
```

```
      COMMON N0,NS,NJ,NT,N_MAX_BAT,N_BAT

      NT=0
      DEL = DELFILESQQ ('Lixo.txt')
      DEL = DELFILESQQ ('Melhor_result.s')
      DEL = DELFILESQQ ('Carta_Gantt.s')

      OPEN  (10,FILE='INIC.E')
      READ  (10,*) N0,NS,NJ,(N_BAT(I),I=1,N0),((M(I,BAT),BAT=1,N_BAT(I)),I
=1,N0),&
                   alfa,DC_MAX,equil,sig_t,PA,larg,LIMI,LIMR,Tempo_lim
      CLOSE (10)

      DO I=1,N0
        N_MAX_BAT=MAX(N_MAX_BAT,N_BAT(I))
        NT=NT+N_BAT(I)
      END DO

      ALLOCATE (C_FIX(3,N0),C_VAR(3,N0))

!     Compra intermediaria de material
      WRITE (*,1000)'Existe compra intermediaria',/'(0) Nao',/'(1) Sim'
      READ *,INTER

      SELECT CASE (INTER)

        CASE (0)
!         Nao ha compra
          GOTO 980

        CASE DEFAULT
!         Estado, Slot e Quantidade de material
          WRITE (*,2000)'Estado, Slot e Quantidade',/'(0) Fim'
910       READ   *,EST
          IF (EST.EQ.0) GOTO 980
         READ   *,SLOT,AUX
          AUX_S(EST,SLOT)=AUX
          GOTO 910

      END SELECT

!     Estocagem intermediária (ZW, NIS)
980   NEI=0
      WRITE (*,1000)'Existe estado instavel (ZW, NIS)',/'(0) Nao',/'(1) Sim'
      READ *,INTER

      SELECT CASE (INTER)

        CASE (0)
!         Nao ha estado instavel
          GOTO 990

        CASE DEFAULT
!         Estado instável
          WRITE (*,2000)'Estado',/'(0) Fim'
920       READ   *,EST
          IF (EST.EQ.0) GOTO 990
          NEI=NEI+1
          E_INST(NEI)=EST
          GOTO 920

      END SELECT
      E_INST(1)=8; E_INST(2)=11; E_INST(3)=16
      E_INST(4)=19; E_INST(5)=23; nei=5

!     Algoritmo
```

```
990   WRITE (*,1000)'Escolha o algoritmo',/(1) Glauber',/(2) Metropolis'
      READ   *,OP_ALG

!     Método de recozimento
      WRITE (*,3000)'Escolha o Sched. Anneal.',/(1) Exponencial',&
            /(2) Aarts van Laarhoven',/(3) Lundy e Mees'
         READ   *,OP_SCH

!     Rearranjo da sequencia
      WRITE (*,3000)'Escolha o rearranjo',/(1) Deslocamento',&
            /(2) Permutacao',/(3) Permutacao de adjacentes'
      READ   *,OP_ARR

!     Realimentação de sequencia inicial
      WRITE (*,1000) 'Realimentacao',/(0) Nao',/(1) Sim'
      READ   *,REALI

1000  FORMAT (/,1X,A,2(/,4X,A))
2000  FORMAT (/,1X,A,/,4X,A)
3000  FORMAT (/,1X,A,3(/,4X,A))

      OPEN  (20,FILE='UTIL.E')
      READ  (20,*) ((C_FIX(U,I),I=1,N0),U=1,3),((C_VAR(U,I),U=1,3),I=1,N0)
,(OF(U),U=1,3)
      CLOSE (20)

!     Arquivos de saida

      OPEN (21,FILE='Histor_aloca.S')
      OPEN (31,FILE='Histor_utilid.S')
      OPEN (41,FILE='Histor_result.S')

      auxm=0; auxf=0; tempo_corrido = TIMEF()

      SELECT CASE (REALI)

        CASE (0)
!       Sem Realimentacao
          CALL INICIAL (SEQ)

        CASE DEFAULT
!       Com Realimentacao

          OPEN  (50,FILE='Realim.E')
          READ  (50,*) (SEQ(I),I=1,NT),SPAN_BEST
          SEQ_BEST=SEQ
          CLOSE (50)

      END SELECT

50    IND1=0; MEM=.FALSE.; Tempo_lim=Tempo_lim*60
      CONTR=0; CONTM=0; NF=0
      FIM=3*NT**3
      T=-DC_MAX/LOG(PA); Tf=0.36*T
      b=(T-Tf)/(FIM*T*Tf)

!     Inicia lista de sequencias analisadas
99    CALL ANTIGAS (NT,SEQ,IND1,MEM)

      IF (MEM) GOTO 109

!     Teste de factibilidade por balanço de massa
      inst_01 = TIMEF()
      CALL TESTE_FACT(M,AUX_S,SEQ,SEQ_AT,MEM)
      inst_02 = TIMEF()
      tempo=inst_02-inst_01; tfac=tempo
      SEQ_AUX=SEQ_AT
```

```
!     Criterio de parada por tempo limite
      tempo_corrido  = TIMEF()
      IF (tempo_corrido .GT.Tempo_lim) THEN
        PRINT *, 'Estouro de tempo limite'
        GOTO 300
      END IF


109   IF (MEM) THEN
!       geracao de numero randomico

        CALL SUBRAND (OP_ARR,NT,N1,N2)

!       geracao de nova sequencia inicial

        CALL GERA_MAT (OP_ARR,N1,N2,NO,SEQ_AT,SEQ,CONTR)
        GOTO 99
      END IF

      ITER=01

!     alocacao da sequencia inicial

      inst_01=TIMEF()
      CALL MAKESPAN (ITER,M,AUX_S,SEQ_AT,C_FIX,C_VAR,OF,NEI,E_INST,SPAN_AT
,MEM)
      inst_02=TIMEF()
      tempo=inst_02-inst_01; tmake=tempo

!     Criterio de parada
      tempo_corrido  = TIMEF()
      IF (tempo_corrido .GT.Tempo_lim) THEN
        PRINT *, 'Estouro de tempo limite'
        GOTO 300
      END IF

      IF (MEM) THEN
        SEQ_AT=SEQ_AUX
        GOTO 109
      END IF

      best_time=inst_01
      SEQ_BEST=SEQ_AT; SPAN_BEST=SPAN_AT
      result=RENAMEFILEQQ('Lixo.txt','Melhor_result.s')

!     arquivo de saida

      WRITE  (41,4001)'M_INI',SPAN_AT,' SEQ',(SEQ_AT(I),I=1,NT)
4001  FORMAT (1X,A,I4,1X,A,200I5.4)

!     inicia laço
      DO ITER=2,FIM

!       Criterio de parada
100     tempo_corrido  = TIMEF()
        IF (tempo_corrido .GT.Tempo_lim) THEN
          PRINT *, 'Estouro de tempo limite'
          GOTO 300
        END IF

!       geracao de numero randomico
        CALL SUBRAND (OP_ARR,NT,N1,N2)

!       geracao de sequencia candidata

        CALL GERA_MAT (OP_ARR,N1,N2,NO,SEQ_AT,SEQ,CONTR)
```

```
!       comparacao com as sequencias anteriores
!       A subrotina antigas retorna MEM=.TRUE. caso a sequencia seja
!       repetida, o que é frequente para pequeno número de operacoes
        CALL ANTIGAS (NT,SEQ,IND1,MEM)
        IF (MEM) GOTO 100

!       teste de factibilidade por balanço de massa
        inst_01 = TIMEF()
        CALL TESTE_FACT(M,AUX_S,SEQ,SEQ_CAND,MEM)
        inst_02 = TIMEF()
        tempo=inst_02-inst_01; tfac=tfac+tempo
        IF (MEM) GOTO 100

!       calculo da sequencia candidata

        inst_01=TIMEF()
        CALL MAKESPAN(ITER,M,AUX_S,SEQ_CAND,C_FIX,C_VAR,OF,NEI,E_INST,SPAN
_CAND,MEM)
        inst_02=TIMEF()
        tempo=inst_02-inst_01; tmake=tmake+tempo
        IF (MEM) GOTO 100

!       arquivo de saida
        WRITE (41,4001) 'M_CAN',SPAN_CAND,' SEQ',(SEQ_CAND(I),I=1,NT)

!       comparacao de funcoes
        DELTA=(SPAN_BEST-SPAN_CAND)

        SELECT CASE (OP_ALG)

          CASE (1)
!         Algoritmo de Glauber
            EXPR=(DEXP(DELTA/T))/(1+DEXP(DELTA/T))

          CASE DEFAULT
!         Algoritmo Metropolis
            EXPR=DEXP(DELTA/T)

        END SELECT

!       se a candidata substitui a melhor
        IF (DELTA.GT.0) THEN
          best_time=inst_01
          SPAN_BEST=SPAN_CAND
          SEQ_BEST=SEQ_CAND
          DEL = DELFILESQQ ('Melhor_result.s')
          result=RENAMEFILEQQ('Lixo.txt','Melhor_result.s')
          XX=ITER; CONTM=CONTM+1; NF=0
        END IF

!       se a candidata e rejeitada
        IF (PA.GE.EXPR) THEN
          NF=NF+1
          IF (NF.GT.larg) THEN
            SEQ_AT=SEQ_BEST; SPAN_AT=SPAN_BEST
          END IF
          GOTO 200
        END IF

!       renomeando sequencia atual
        SEQ_AT=SEQ_CAND

!       intensificacao de busca (Recozimento)
200     IF (CONTM.GE.LIMI .OR. CONTR.GE.LIMR) THEN

          SELECT CASE (OP_SCH)
```

```
            CASE (1)
!             Ajuste Exponencial
              T=T*alfa

            CASE (2)
!             Aarts e van Laarhoven   EQUIL=0.3   SIGMA(T)=2.2
              T=T/(1+(LOG(1+equil)*T/(3*sig_t)))

            CASE DEFAULT
!             Lundy e Mees
              T=T/(1+b*T)

          END SELECT

          CONTM=0; CONTR=0
        END IF
      END DO

300   CLOSE (21)
      CLOSE (31)

!     Estatísticas
      tempo_corrido = TIMEF()
      WRITE (41,4011) 'Melhor solução após ',XX,' iterações','Consumo em s
eg. e % em ',&
      tempo_corrido,' (s) de execução'
      WRITE (41,4012) 'Tempo total        ',tmake,(tmake/tempo_corrido )*1
00
      WRITE (41,4012) 'Tempo p/ alocação  ',tfac,(tfac/tempo_corrido )*100
      WRITE (41,4012) 'CPU Time(s) =      ',tempo_corrido ,(1-(tmake+tfac)
/tempo_corrido )*100

4011  FORMAT (1x,a,I4,a,1X,A,F10.4,A)
4012  FORMAT (1X,A,F10.4,2X,F10.4)

!     final do laco e saida dos resultados

      WRITE (*,5000) 'M ',SPAN_BEST,'Iter ',XX,'nt',nt,'Tempo',best_time,'
 SEQ',&
      (SEQ_BEST(I),I=1,N0)
      WRITE (41,5001) 'Melhor solução',SPAN_BEST,' encontrada após',best_
time,' (s)','Total de iterações',ITER
5000  FORMAT (/,1X,A,I5,2(2X,A,I4),2X,A,1X,D14.4,1X,A,/,200(2X,I5.5))
5001  FORMAT (1x,A,I5,a,1X,D14.4,A,/,1x,a,I4)

      CLOSE (41)

      OPEN  (51,FILE='Realim.E')
      WRITE (51,6001) (SEQ_BEST(I),I=1,NT),SPAN_BEST
6001  FORMAT (1X,200I5.4,/,1X,A,2X,I5)
      CLOSE (51)

      CALL SUB_GANTT ()

      DEL = DELFILESQQ ('Lixo.txt')

      END PROGRAM Schedule

!**************FIM DO PROGRAMA PRINCIPAL*********************************

!   SUBROTINA PROGGANTT
!   Saída de resultados na forma de carta de Gantt
      subroutine sub_gantt()

      integer n_max_bat,i,col,no,nt,n_bat(30)
```

```
      integer,allocatable::gan(:,:),proc(:),oper(:),slot(:),tp(:),tempo(:)
,&
                          ini_jan(:,:),fim_jan(:,:),mjan(:,:)

      COMMON N0,NS,NJ,NT,N_MAX_BAT,N_BAT

      allocate (gan(100,nj),proc(nt),oper(nt),slot(nt),tp(nt),tempo(100),&
                ini_jan(no,n_max_bat),fim_jan(no,n_max_bat),mjan(100,nt))

      gan='.'; mjan='.'

      do i=1,100
        tempo(i)=i
      end do

      open   (40,file='Jan.E')
      read   (40,*) ((ini_jan(i,bat),bat=1,n_bat(i)),i=1,no),&
                    ((fim_jan(i,bat),bat=1,n_bat(i)),i=1,no),&
                    ini_horiz,fim_horiz
      close  (40)

      open   (71,file='Melhor_result.s')
      read   (71,*) (proc(i),oper(i),slot(i),tp(i),i=1,nt)
      close  (71)

      do i=1,nt
        do col=slot(i),slot(i)+tp(i)-1
          gan(col,proc(i))=oper(i)
        end do
      end do

      col=0
      do i=1,no
        do bat=1,n_bat(i)
          col=col+1
          do aux=ini_jan(i,bat),fim_jan(i,bat)
            mjan(aux,col)=i
          end do
        end do
      end do

      open   (81,file='Carta_Gantt.s')

      write  (81,100) gan
      write  (81,100) tempo
      write  (81,150)
      write  (81,100) mjan

100   format (/,100(1x,I2.2))
150   format (/)

      close (81)

      return
      end
!     FIM DA SUBROTINA PROGGANTT


!     INICIO SUBROTINA PARA GERACAO DE SEQUENCIA INICIAL
!     Sequenciamento das operações por ordem decrescente de criticalidade


      SUBROUTINE INICIAL (SEQ)

      INTEGER  N0,NJ,NT,PROC,I,IL,IJ,IM,BAT,N_BAT(30)

      INTEGER(4) TAR,SEQ(100)
```

```
      REAL(8) AUXC

      COMMON NO,NS,NJ,NT,N_MAX_BAT,N_BAT

      REAL(8),ALLOCATABLE::CRIT(:),TP(:,:),INI_JAN(:,:),FIM_JAN(:,:)

      ALLOCATE(CRIT(NT),TP(NJ,NO),INI_JAN(NO,N_MAX_BAT),FIM_JAN(NO,N_MAX_B
AT))


      OPEN  (30,FILE='DADOS.E')
      READ  (30,*) ((TP(PROC,I),I=1,NO),PROC=1,NJ)
      CLOSE (30)

      OPEN  (40,FILE='JAN.E')
      READ  (40,*) ((INI_JAN(I,BAT),BAT=1,N_BAT(I)),I=1,NO),&
            ((FIM_JAN(I,BAT),BAT=1,N_BAT(I)),I=1,NO)
      CLOSE (40)

      IJ=0

      DO I=1,NO
        DO PROC=1,NJ
          IF (TP(PROC,I).NE.0) THEN

            DO BAT=1,N_BAT(I)
              AUXC=TP(PROC,I)/(FIM_JAN(I,BAT)-INI_JAN(I,BAT))
              IJ=IJ+1;TAR=(100*I)+BAT

              DO IL=1,IJ
                IF (AUXC.GT.CRIT(IL)) THEN

                  DO IM=NT-1,IL,-1
                    CRIT(IM+1)=CRIT(IM)
                    SEQ(IM+1)=SEQ(IM)
                  END DO

                  CRIT(IL)=AUXC
                  SEQ(IL)=TAR
                  GOTO 100
                END IF
              END DO
100         END DO
            GOTO 200
          END IF
        END DO
200   END DO

      RETURN
      END
!     FIM DA SUBROTINA PARA GERACAO DE SEQUENCIA INICIAL


!     INICIO SUBROTINA PARA TESTE DE FACTIBILIDADE
!     Teste de factibilidade por precedência de operações, balanço de mass
a

      SUBROUTINE TESTE_FACT (M,AUX_S,SEQ,SEQ_CAND,MEM)

      INTEGER  NO,NS,NJ,NT,EST,PROC,BAT,SLOT,SLOTL,I,IT,IN,V,AUXV(10),W(10
,100),&
               SEQ(200),SEQ_CAND(200),OGs(30,40),OCs(30,40),&
               INI_HORIZ,FIM_HORIZ,INI_JAN(30,20),FIM_JAN(30,20),N_BAT(30)


      REAL TP(10,30),M(30,20),RO_G(30,40),RO_C(30,40),V_MAX(10),AUXS(40),S
```

```
(40,100),AUX_S(40,100)

      LOGICAL(1) MEM

      COMMON N0,NS,NJ,NT,N_MAX_BAT,N_BAT

      OPEN  (30,FILE='DADOS.E')
      READ  (30,*) ((TP(PROC,I),I=1,N0),PROC=1,NJ),&
            ((0Gs(I,EST),EST=1,NS),I=1,N0),((0Cs(I,EST),EST=1,NS),I=1,N0),
&
            ((R0_G(I,EST),EST=1,NS),I=1,N0),((R0_C(I,EST),EST=1,NS),I=1,N0
),&
            (V_MAX(PROC),PROC=1,NJ),(S(EST,1),EST=1,NS)
      CLOSE (30)

      OPEN  (40,FILE='JAN.E')
      READ  (40,*) ((INI_JAN(I,BAT),BAT=1,N_BAT(I)),I=1,N0),&
            ((FIM_JAN(I,BAT),BAT=1,N_BAT(I)),I=1,N0),INI_HORIZ,FIM_HORIZ

      CLOSE (40)



      IN=0
      DO SLOT=INI_HORIZ,FIM_HORIZ
        IF (SLOT.GT.1) THEN
          DO EST=1,NS
            S(EST,SLOT)=S(EST,SLOT-1)+AUX_S(EST,SLOT)
          END DO
        END IF
      END DO

      DO EST=1,NS
        AUXS(EST)=S(EST,1)
      END DO

100   DO IT=1,NT
        IF (SEQ(IT).NE.0) THEN
          I=SEQ(IT)/100; BAT=SEQ(IT)-I*100
          DO PROC=1,NJ
            IF (TP(PROC,I).NE.0) THEN

              V=0; AUXV=0
              DO EST=1,NS
                IF (0Cs(I,EST).EQ.1 .AND. AUXS(EST)+1E-5.LT.M(I,BAT)*R0_C(
I,EST)) GOTO 200
                V=V+1; AUXV(V)=EST
              END DO

              DO EST=1,V
                AUXS(AUXV(EST))=AUXS(AUXV(EST))-M(I,BAT)*R0_C(I,AUXV(EST))
              END DO

              DO EST=1,NS
                IF (0Gs(I,EST).EQ.1) AUXS(EST)=AUXS(EST)+M(I,BAT)*R0_G(I,E
ST)
              END DO

              IN=IN+1; SEQ_CAND(IN)=SEQ(IT); SEQ(IT)=0

              GOTO 300
            END IF
          END DO
        END IF
200   END DO

300   IF (IN.LT.NT) GOTO 100
```

```
      MEM=.FALSE.; W=0

400   DO IT=1,NT
        I=SEQ_CAND(IT)/100; BAT=SEQ_CAND(IT)-I*100
        DO PROC=1,NJ
          IF (TP(PROC,I).NE.0) THEN
            SLOT=INI_JAN(I,BAT)

500         IF ((SLOT+TP(PROC,I)-1).GT.FIM_JAN(I,BAT)) THEN
                MEM=.TRUE.
                RETURN
            END IF

            DO SLOTL=SLOT,(SLOT+TP(PROC,I)-1)

              IF (W(PROC,SLOTL).EQ.1) THEN
                SLOT=SLOT+1
                GOTO 500
              END IF

            END DO

            DO EST=1,NS
              IF (OCs(I,EST).EQ.1) THEN
                IF (S(EST,SLOT).LT.(RO_C(I,EST)*M(I,BAT)-1E-5)) THEN
                  SLOT=SLOT+1
                  GOTO 500
                END IF
              END IF
            END DO

            DO SLOTL=SLOT,(SLOT+TP(PROC,I)-1)
              W(PROC,SLOTL)=1
            END DO

            DO EST=1,NS

              IF (OGs(I,EST).EQ.1) THEN
                DO SLOTL=(SLOT+TP(PROC,I)),FIM_HORIZ
                  S(EST,SLOTL)=S(EST,SLOTL)+RO_G(I,EST)*M(I,BAT)
                END DO
              END IF

              IF (OCs(I,EST).EQ.1) THEN
                DO SLOTL=INI_HORIZ,FIM_HORIZ
                  S(EST,SLOTL)=S(EST,SLOTL)-RO_C(I,EST)*M(I,BAT)
                  IF (S(EST,SLOTL).LT.0) S(EST,SLOTL)=0
                END DO
              END IF

            END DO

          GOTO 600
          END IF
        END DO
600   END DO

700   RETURN
      END
!     FIM DA SUBROTINA PARA TESTE DE FACTIBILIDADE


!     INICIO SUBROTINA PARA CALCULO DO TEMPO DE PROCESSAMENTO
!     Alocação temporal das operações sujeita a todas as restrições

      SUBROUTINE MAKESPAN (ITER,M,AUX_S,SEQ_AUX,C_FIX,C_VAR,OF,NEI,E_INST,
```

```
MSPAN,MEM)

      INTEGER N0,NS,NJ,NT,EST,PROC(10),PL,PK,MSPAN,SLOT(30),SLOTL,AUX1,I(3
0),IT,ITL,BAT(30),&
              U,OF(3),A,C,NEI,E_INST(20),ITER,W(10,100),SEQ(200),SEQ_AUX(2
00),TP(10,30),&
              OGs(30,40),OCs(30,40),N_BAT(30),INI_JAN(30,10),FIM_JAN(30,10
),DEL(10),&
              INI_HORIZ,FIM_HORIZ

      REAL C_FIX(3,30),C_VAR(3,30),DUT(3),UT(3,100),AUX_S(40,100),S(40,100
),RO_G(30,40),&
              RO_C(30,40),V_MAX(10),M(30,20)

      LOGICAL(1) MEM,MEM2

      COMMON N0,NS,NJ,NT,N_MAX_BAT,N_BAT

      OPEN   (30,FILE='DADOS.E')
      READ   (30,*) ((TP(PROCL,IT),IT=1,N0),PROCL=1,NJ),&
             ((OGs(IT,EST),EST=1,NS),IT=1,N0),((OCs(IT,EST),EST=1,NS),IT=1,
N0),&
             ((RO_G(IT,EST),EST=1,NS),IT=1,N0),((RO_C(IT,EST),EST=1,NS),IT=
1,N0),&
             (V_MAX(PROCL),PROCL=1,NJ),(S(EST,1),EST=1,NS)
      CLOSE  (30)

      OPEN   (40,FILE='JAN.E')
      READ   (40,*) ((INI_JAN(IT,C),C=1,N_BAT(IT)),IT=1,N0),&
             ((FIM_JAN(IT,C),C=1,N_BAT(IT)),IT=1,N0),INI_HORIZ,FIM_HORIZ

      CLOSE (40)

!     Inicializacao de W, UT e S
!     Janelas de tempo, disponibilidade de utilidades e estados
      SEQ=SEQ_AUX; W=0; UT=0; MSPAN=0; MEM=.FALSE.

      DO SLOTL=INI_HORIZ,FIM_HORIZ
        IF (SLOTL.GT.1) THEN
          DO EST=1,NS
            S(EST,SLOTL)=S(EST,SLOTL-1)+AUX_S(EST,SLOTL)
          END DO
        END IF
      END DO

!     alocacao
      OPEN   (61,FILE='Lixo.txt')
      WRITE  (21,2011) ITER
2011  FORMAT (/,10X,I5)

      DO IT=1,NT
        IF (SEQ(IT).NE.0) THEN
          C=1; I(C)=SEQ(IT)/100; BAT(C)=SEQ(IT)-I(C)*100
          DO PL=1,NJ
            IF (TP(PL,I(C)).NE.0) THEN
              PROC(C)=PL; SLOT(C)=INI_JAN(I(C),BAT(C))-1

100           C=1; SLOT(C)=SLOT(C)+1; DEL(C)=IT

!             verificacao de janela de alocacao
              IF ((SLOT(1)+TP(PROC(1),I(1))-1).GT.FIM_JAN(I(1),BAT(1))) TH
EN
                WRITE (21,2031) 'Fim da janela',SEQ(IT),'Estado ',EST,&
                'Slot ',INI_JAN(I(1),BAT(1)),'Demanda ',(M(I(1),BAT(1))-S(
EST,INI_JAN(I(1),BAT(1)))),&
!               'Demanda Utilidades',((DUT(U)-OF(U)),U=1,3),IT,(SEQ_AUX(IT
),IT=1,NT)
```

```
                  'Demanda Utilidades',((DUT(U)-OF(U)),U=1,3),(IT,(SEQ_AUX(j
i)),ji=1,NT)
2031              FORMAT (1X,A,I5,2(1X,A,I3),/,1X,A,F5.2,1X,A,3F6.2,/,I4,3X,
5OI5)
                  MSPAN=1000; MEM=.TRUE.
                  RETURN
                END IF

200             CALL ALOCACAO (I,PROC,BAT,SLOT,M,TP,W,OCs,RO_G,RO_C,S,C_FIX,
C_VAR,OF,UT,DUT,C,MEM2)

                IF (MEM2) GOTO 100

                IF (NEI.NE.0) THEN
                  DO A=1,NEI
                    DO EST=1,NS
                      IF (OGs(I(C),EST).EQ.1 .AND. EST.EQ.E_INST(A)) THEN
                        C=C+1
                        DO ITL=IT+1,NT
                          IF (SEQ(ITL).NE.0) THEN
                            I(C)=SEQ(ITL)/100; BAT(C)=SEQ(ITL)-I(C)*100
                            DO PK=1,NJ
                              IF (TP(PK,I(C)).NE.0 .AND. OCs(I(C),EST).EQ.1)
 THEN
                                PROC(C)=PK
                                SLOT(C)=(SLOT(C-1)+TP(PROC(C-1),I(C-1)))
                                DEL(C)=ITL
                                GOTO 200
                              END IF
                            END DO
                          END IF
                        END DO
                      END IF
                    END DO
                  END DO
                END IF

!               atualizacao de estoque
                DO AUX1=1,C
                  SEQ(DEL(AUX1))=0

!                 Alocacao do processador
                  DO SLOTL=SLOT(AUX1),(SLOT(AUX1)+TP(PROC(AUX1),I(AUX1))-1)
                    W(PROC(AUX1),SLOTL)=1
                  END DO

!                 Arquivo de saida histor_aloca.S
                  WRITE (21,2041) PROC(AUX1),I(AUX1),SLOT(AUX1),TP(PROC(AUX1
),I(AUX1))
                  WRITE (61,2041) PROC(AUX1),I(AUX1),SLOT(AUX1),TP(PROC(AUX1
),I(AUX1))
2041              FORMAT(4(3X,I3))

                  DO EST=1,NS

!                   verificacao de estado instavel

                    MEM2=.FALSE.
                    DO A=1,NEI
                      IF (EST.EQ.E_INST(A)) MEM2=.TRUE.
                    END DO

!                   GERACAO
                    IF (OGs(I(AUX1),EST).EQ.1) THEN
                      IF (MEM2) THEN
!                       estado instavel
```

```
                        SLOTL=SLOT(AUX1)+TP(PROC(AUX1),I(AUX1))
                        S(EST,SLOTL)=S(EST,SLOTL)+R0_G(I(AUX1),EST)*M(I(AUX1
),BAT(AUX1))
                      ELSE
!                       sem estado instavel
                        DO SLOTL=(SLOT(AUX1)+TP(PROC(AUX1),I(AUX1))),FIM_HOR
IZ
                          S(EST,SLOTL)=S(EST,SLOTL)+R0_G(I(AUX1),EST)*M(I(AU
X1),BAT(AUX1))
                        END DO
                      END IF

!                   CONSUMO
                    ELSE IF (OCs(I(AUX1),EST).EQ.1) THEN
                      IF (MEM2) THEN
!                       estado instavel
                        S(EST,SLOT(AUX1))=S(EST,SLOT(AUX1))-R0_C(I(AUX1),EST
)*M(I(AUX1),BAT(AUX1))
                      ELSE
!                       sem estado instavel
                        DO SLOTL=INI_HORIZ,FIM_HORIZ
                          S(EST,SLOTL)=S(EST,SLOTL)-R0_C(I(AUX1),EST)*M(I(AU
X1),BAT(AUX1))
                          IF (S(EST,SLOTL).LT.0) S(EST,SLOTL)=0
                        END DO
                      END IF
                    END IF

                  END DO

!                 atualizacao de makespan
                  MSPAN=MAX(MSPAN,(SLOT(AUX1)+TP(PROC(AUX1),I(AUX1))-1))

                END DO

                GOTO 400
              END IF
            END DO
          END IF
400     END DO

!       Arquivo de saida Histor_utilid.S e lixo.txt
        WRITE (31,3001) ' SLOT  ','UTIL(1)','UTIL(2)','UTIL(3)',ITER
        WRITE (61,3002) ' SLOT  ','UTIL(1)','UTIL(2)','UTIL(3)',MSPAN,ITER

        DO SLOTL=1,MSPAN
          WRITE (31,3011) SLOTL,UT(1,SLOTL),UT(2,SLOTL),UT(3,SLOTL)
          WRITE (61,3011) SLOTL,UT(1,SLOTL),UT(2,SLOTL),UT(3,SLOTL)
        END DO

        WRITE (61,1012) (EST,EST=1,NS)
        WRITE (61,1013) (S(EST,MSPAN),EST=1,NS)

1012    FORMAT (/,1X,40I5)
1013    FORMAT (/,1X,40F5.1)
3001    FORMAT (/,4(3X,A),3X,I4)
3002    FORMAT (/,4(3X,A),2(2X,I4))
3011    FORMAT (3X,I4,3(3X,F7.2))
        CLOSE (61)

        RETURN
        END
!       FIM DA SUBROTINA PARA CALCULO DO TEMPO DE PROCESSAMENTO


!       INICIO DA SUBROTINA PARA ALOCAÇÃO DE OPERAÇÕES COM INTERMEDIÁRIOS IN
```

```
STÁVEIS
      SUBROUTINE ALOCACAO (I,PROC,BAT,SLOT,M,TP,W,OCs,RO_G,RO_C,S,C_FIX,C_
VAR,OF,UT,DUT,C,MEM2)

      INTEGER NO,NS,NJ,NT,EST,PROC(10),SLOT(30),SLOTL,I(30),BAT(30),U,OF(3
),C,&
              W(10,100),TP(10,30),OCs(30,40)

      REAL C_FIX(3,30),C_VAR(3,30),DUT(3),UT(3,100),AUX_U,S(40,100),RO_G(3
0,40),&
           RO_C(30,40),AUX,M(30,20)

      LOGICAL(1) MEM2

      COMMON NO,NS,Nd,NT,N_MAX_BAT,N_BAT

      MEM2=.FALSE.; M(I(0),BAT(0))=0


!     verificacao de ocupacao de processador
      DO SLOTL=SLOT(C),(SLOT(C)+TP(PROC(C),I(C))-1)
        IF (W(PROC(C),SLOTL).EQ.1) THEN
          MEM2=.TRUE.
          RETURN
        END IF
      END DO

!     verificacao de presenca de materia prima
      DO EST=1,NS
        IF (OCs(I(C),EST).EQ.1) THEN
          IF (C.EQ.1 .AND. S(EST,SLOT(C))+1E-5.LT.RO_C(I(C),EST)*M(I(C),BA
T(C))) THEN
            MEM2=.TRUE.
            RETURN
          ELSE IF (C.NE.1) THEN
            AUX=(RO_C(I(C),EST)*M(I(C),BAT(C))-1E-5)-(RO_G(I(C-1),EST)*M(I
(C-1),BAT(C-1)))
            IF (S(EST,SLOT(C)).LT.AUX) THEN
              MEM2=.TRUE.
              RETURN
            END IF
          END IF
        END IF
      END DO

!     Consumo de Recursos
      DO U=1,3
        DO SLOTL=SLOT(C),SLOT(C)+TP(PROC(C),I(C))-1
          AUX_U=UT(U,SLOTL)+C_FIX(U,I(C))+C_VAR(U,I(C))*M(I(C),BAT(C))
          IF (AUX_U.GT.OF(U)) THEN
            DUT(U)=MAX(DUT(U),AUX_U)
            MEM2=.TRUE.
            RETURN
          END IF
          UT(U,SLOTL)=AUX_U
        END DO
      END DO

      RETURN
      END
!     FIM DA SUBROTINA PARA ALOCAÇÃO DE OPERAÇÕES COM INTERMEDIÁRIOS INSTÁ
VEIS

!     INICIO DA SUBROTINA PARA GERACAO DE SEQUENCIA CANDIDATA

      SUBROUTINE GERA_MAT (OP_ARR,N1,N2,NO,SEQ_AT,SEQ,CONTR)
      INTEGER  OP_ARR,OPER,NO,N1,N2,CONTR,SEQ_AT(200),SEQ(200),AUX1
```

```
      print *, 'geramat'

      SELECT CASE (OP_ARR)

      CASE(1)
!       Deslocamento
        DO OPER=1,N1-1
          SEQ(OPER)=SEQ_AT(OPER)
        END DO

        DO OPER=N1,N2-1
          SEQ(OPER+1)=SEQ_AT(OPER)
        END DO

        DO OPER=N2+1,NO
          SEQ(OPER)=SEQ_AT(OPER)
        END DO

        SEQ(N1)=SEQ_AT(N2)

      CASE(2)
!       Permutação
        SEQ=SEQ_AT
        AUX1=SEQ(N1)
        SEQ(N1)=SEQ(N2)
        SEQ(N2)=AUX1

      CASE DEFAULT
!       Permutação de Adjacentes
        SEQ=SEQ_AT
        AUX1=SEQ(N1)
        SEQ(N1)=SEQ(N2)
        SEQ(N2)=AUX1

      END SELECT

      CONTR=CONTR+1

      RETURN
      END
!     FIM DA SUBROTINA PARA GERACAO DE SEQUENCIA CANDIDATA


!     LISTA DE SEQUENCIAS ANTERIORES
!     OLD Ultimas cinco sequencias analisadas
!     Primeira coluna = MSPAN, seguintes = operacoes
      SUBROUTINE ANTIGAS (NT,SEQ,IND1,MEM)
      INTEGER  I,TAR,IND1,NT,SEQ(200),OLD(30,200)
      LOGICAL(1) MEM
      SAVE OLD

      MEM=.FALSE.

!     Inicia lista com sequencia inicial e zera demais linhas
      IF (IND1.EQ.0) THEN

        OLD=0
        DO TAR=1,NT
          OLD(1,TAR)=SEQ(TAR)
        END DO

        IND1=1
        RETURN

      ELSE
        DO I=1,30
```

```
            MEM=.TRUE.
            DO TAR=1,NT
!             Rejeita sequencias iguais
              IF (SEQ(TAR).NE.OLD(I,TAR)) MEM=.FALSE.
            END DO
            IF (MEM) RETURN
          END DO
        END IF

!       Atualiza lista de sequencias

        DO I=30,2,-1
          DO TAR=1,NT
            OLD(I,TAR)=OLD(I-1,TAR)
          END DO
        END DO

        DO TAR=1,NT
          OLD(1,TAR)=SEQ(TAR)
        END DO

        RETURN
        END
!       FIM DA SUBROTINA ANTIGAS

!       Funcao geradora de numeros randomicos RAN(IDUM)
!       Fonte Numerical Recipes in Fortan
!       (C) Copr. 1986-92 Numerical Recipes Software #0).
        SUBROUTINE SUBRAND (OP_ARR,NT,N1,N2)
        INTEGER  OP_ARR,IND,NT,P(2),N1,N2,IDUM
        DOUBLE PRECISION RAN3

        P(2)=0; IDUM=0

        DO IND = 1,2
100       P(IND) = AINT(100*RAN3(IDUM))
          P(IND) = MOD(P(IND),NT+1)

          IF (P(IND).EQ.0)  GOTO 100

          IF (OP_ARR.EQ.1 .OR. OP_ARR.EQ.2) THEN
            IF (P(2).EQ.P(1)) GOTO 100
          ELSE
            IF (P(1).LT.NT) THEN
              P(2)=P(1)+1
              GOTO 200
            ELSE
              GOTO 100
            END IF
          END IF
200     END DO

        N1=MIN(P(1),P(2))
        N2=MAX(P(1),P(2))

        RETURN
        END

        FUNCTION ran3(idum)
        INTEGER  idum
        DOUBLE PRECISION MBIG,MSEED,MZ
        DOUBLE PRECISION ran3,FAC
        PARAMETER (MBIG=4000000.,MSEED=1618033.,MZ=0.,FAC=1./MBIG)
        INTEGER  i,iff,ii,inext,inextp,k
        DOUBLE PRECISION mj,mk,ma(55)
        SAVE iff,inext,inextp,ma
        DATA iff /0/
```

```
      if(idum.lt.0.or.iff.eq.0)then
        iff=1
        mj=MSEED-iabs(idum)
        mj=mod(mj,MBIG)
        ma(55)=mj
        mk=1
        do 11 i=1,54
          ii=mod(21*i,55)
          ma(ii)=mk
          mk=mj-mk
          if(mk.lt.MZ)mk=mk+MBIG
          mj=ma(ii)
11      continue
        do 13 k=1,4
          do 12 i=1,55
            ma(i)=ma(i)-ma(1+mod(i+30,55))
            if(ma(i).lt.MZ)ma(i)=ma(i)+MBIG
12        continue
13      continue
        inext=0
        inextp=31
        idum=1
      endif
      inext=inext+1
      if(inext.eq.56)inext=1
      inextp=inextp+1
      if(inextp.eq.56)inextp=1
      mj=ma(inext)-ma(inextp)
      if(mj.lt.MZ)mj=mj+MBIG
      ma(inext)=mj
      ran3=mj*FAC

      return
      END
!     FIM DA FUNCAO
```