UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ADMINISTRAÇÃO
Programa de Pós-Graduação em Administração
Grupo de Estudos em Sistemas de Informação e de Apoio à Decisão
Dissertação de Mestrado

Leonardo Rosa Rohde

REPRESENTAÇÃO NA FORMA NORMAL DISJUNTIVA PARA A FLEXIBILIDADE DE SEQÜÊNCIA NA MANUFATURA

Dissertação de Mestrado, apresentada ao Programa de Pós-Graduação em Administração da Universidade Federal do Rio Grande do Sul como requisito parcial para a obtenção do título de Mestre em Administração.

Orientadores:
Prof. Denis Borenstein, Ph.D.
Prof. João Luiz Becker, Ph.D.

Porto Alegre, 2002.

“The reasonable man adapts himself to the world;
The unreasonable one persists in trying to adapt the world to himself.
Therefore all progress depends
on the unreasonable man.”

George Bernard Shaw

Agradecimentos

- Primeiramente a Deus, pois sem ele nada seria possível;
- Aos meus pais, Mario e Circe, pela educação e apoio;
- Aos meus irmãos, Leandro, Adriana, Daniel e Rodrigo, pelo apoio e descontração;
- Aos meus amigos pelas festas nos momentos oportunos e paciência nos restantes;
- Aos meus orientadores Denis Borenstein e João Luiz Becker, pela paciência, dedicação, e, é claro, orientação;
- A todo o GESID pelo convívio, aprendizado e oportunidade;
- Aos professores do PPGA, por terem contribuído com a minha formação;
- Muitos foram aqueles que me apoiaram nas horas difíceis e me conduziram até aqui. A eles fica meu sincero agradecimento.

Sumário

Lista de Figuras

Lista de Quadros

Lista de Tabelas

Resumo

A crescente demanda por produtos de melhor qualidade, diferenciados e com custos competitivos tem forçado as manufaturas a se tornarem flexíveis, capacitando-as a responder às mudanças impostas pelo mercado. A flexibilidade permite que as empresas alcancem a customização desejada através da capacitação do sistema de responder e atuar em tempo real, mesmo em um ambiente de incertezas. Para atuar em tempo real, os sistemas de manufatura precisam de representações eficientes dos planos de produção.

Muitas vezes, a atuação em tempo real torna-se inviável devido ao crescimento exponencial no número de planos de produção para cada máquina ou operação adicionada ao sistema. Uma possível solução para este problema é uso de representações adequadas para o espaço de estados. A escolha de uma representação adequada para o espaço de estados influencia na capacidade de reposta em tempo real, pois determina o desempenho computacional do sistema através da utilidade e eficiência dos algoritmos desenvolvidos, tornando possível explorar problemas clássicos de flexibilidade, tais como, seqüenciamento, otimização, etc. Entretanto, a geração de uma representação que trabalhe com o espaço de estados completo de uma manufatura é considerada um problema não polinomial (NP). Esta particularidade dificulta o desenvolvimento de algoritmos que trabalhem com uma manufatura flexível. Assim, a geração de uma representação, que trabalhe com pouca memória computacional e permita o desenvolvimento de heurísticas eficientes, é um importante desafio para uma avaliação efetiva da flexibilidade.

Este trabalho objetiva o desenvolvimento de uma representação para o espaço de estados de uma manufatura com flexibilidade de seqüência. Na construção desta representação são aplicadas técnicas de modelagem baseadas na teoria dos grafos e nos princípios de álgebra booleana. Inicialmente, os grafos são utilizados para representar todas as seqüências de operações de uma manufatura, posteriormente estas seqüências são convertidas em formas normais disjuntivas (FND). Por fim, é apresentada uma possível aplicação da representação na FND em modelos de programação linear.

Palavras Chaves: espaço de estados, representação, FND, grafo, manufatura, flexibilidade e programação linear.

Abstract

Best quality and customised products with competitive cost have been increasingly demanded, what have forced manufactures to become flexible and enable to respond the market changes. Flexibility allows organisations to achieve the desirable customisation through capability of the system to respond and to perform in real-time given external and internal disturbances. However, for working in real-time, the production plans of manufacturing systems need an efficient representation.

Frequently, working in real-time becomes impracticable due to an exponential increase in the number of production plans for each increase in the number of machines or operations in the manufacturing system. One possible solution for this problem is to use an adequate state space representation. The choice of an appropriate representation for state space will influence the ability of the manufacturing system to respond in real time, for it determinates the computational performance of the systems through useful and efficient algorithms developed for adequate representation, making possible to exploit flexibility issues in classical industrial problems such as scheduling, optimisation of manufacturing plans, etc. However, the complete state representation of manufacturing plans in a flexible manufacturing environment is a NP problem. This particularity makes the development of algorithms involving state space of manufacturing systems difficult. Thus, the generation of a state space representation that leads to the use of appropriate low computer memory and allowing the development of efficient heuristics is a an important challenge towards the effective evaluation of flexibility.

This work aims the development of a state space representation for manufacturing systems with sequence flexibility. Methods and models supported by graphs and Boolean algebra theories are applied in the construction of this representation. First, the graphs are used to depict the manufacturing operations sequences and second they are converted to disjunctive normal form (DNF). Finally, the representation in DNF applied to linear programming models is reported.

Keywords: state space, representation, DNF, graph, manufacturing, flexibility and linear programming.

1. INTRODUÇÃO

Atualmente, o mercado tem exigido produtos de melhor qualidade, diferenciados e com custos competitivos, obrigando as manufaturas a buscarem a flexibilização da produção. A flexibilidade permite a customização dos produtos e melhor utilização da capacidade industrial, fornecendo maior eficiência no tratamento das incertezas impostas pelo mercado (GERWIN, 1986; BERNARDO e MOHAMED, 1992; SARKER *et al.* 1994; SHAFER e CHARNES, 1997;).

Os sistemas convencionais de manufatura, baseados em processos tradicionais, apresentam rotinas fixas de produção, onde cada máquina é capaz de executar um único tipo de operação. Contrariamente, o avanço da tecnologia permitiu o desenvolvimento de sistemas flexíveis de manufatura, nos quais as operações não necessitam ser executadas em máquinas dedicadas, numa ordem fixa e predeterminada. Por este motivo, os sistemas de manufatura tornaram-se aptos a trabalhar com uma grande variedade de produtos e alternativas para processá-los (FENSTERSEIFER, 1990).

Entre as vantagens fornecidas por uma manufatura flexível, cita-se a melhoria no tratamento de incertezas, capacidade de adaptação às exigências impostas pelo ambiente, melhor controle sobre os processos de manufatura e a capacidade de unir, em um único sistema, produção customizada e custos competitivos (KOCHIKAR e NARENDRAN,1992; DAST e GENDRA, 1993; CHEN e CHUNG, 1996; BOYER, 1999; KIM e EGBELU, 1999). Devido a estas vantagens, a procura por manufaturas flexíveis cresceu e milhões de dólares foram gastos nestes sistemas no mundo todo. Como resultado destes investimentos, nas últimas décadas, o problema de flexibilidade recebeu a atenção de diversos estudiosos no assunto (BORENSTEIN, 1999; GUPTA e GOYAL, 1992).

Apesar das vantagens fornecidas pela flexibilidade, uma manufatura eficiente precisa utilizar metodologias adequadas no tratamento dos parâmetros de produção. Para garantir essa eficiência, a escolha de um plano de produção adequado é muito importante, principalmente em organizações onde múltiplos produtos são fabricados e existem diversas alternativas para processar tais produtos (KIM e EGBELU, 1999).

Muitas vezes, a escolha de um plano torna-se inviável, devido ao crescimento exponencial no número de alternativas de produção para cada máquina ou operação adicionada ao sistema. Este aumento exponencial nas alternativas de produção dificulta a criação de algoritmos que trabalhem com o espaço de estados de uma manufatura flexível. A geração destes espaços de estados é considerada um problema não polinomial (NP), justificando a utilização de representações adequadas que trabalhem com pouca memória computacional e permitam o desenvolvimento de heurísticas eficientes (CAMPELLO e MACULAN, 1994).

Desta forma, este trabalho objetiva o desenvolvimento de uma representação, que utilize o mínimo possível de memória computacional e gaste pouco tempo de processamento, para armazenar o espaço de estados de uma manufatura com flexibilidade de seqüência. Na construção desta representação são aplicadas técnicas de modelagem baseadas na teoria dos grafos e nos princípios de álgebra booleana. Inicialmente, os grafos são utilizados para representar todas as seqüências de operações de uma manufatura, posteriormente estas seqüências são convertidas em formas normais disjuntivas (FND). Por fim, é apresentada uma possível aplicação da representação na FND em modelos de programação linear.

Com isso, espera-se que o desenvolvimento desta representação possibilite tratar com maior eficiência os sistemas flexíveis de manufatura, auxiliando na tomada de decisão e na aplicação de simulações e mensurações dos processos de produção. No capítulo 2 é apresentada a contextualização da pesquisa. Nos capítulos 3 e 4 encontram-se respectivamente a metodologia de pesquisa e os conceitos sobre técnicas de modelagem. O desenvolvimento da representação é apresentado no capítulo 5. Finalizando, uma possível aplicação da representação desenvolvida é apresentada no capítulo 6, seguido pela conclusão e recomendações.

2. CONTEXTUALIZAÇÃO DA PESQUISA

As empresas de manufatura começaram a experimentar fortes mudanças no meio da década de 60 devido ao desenvolvimento tecnológico e aumento da competitividade internacional. Estas companhias, baseadas em dogmas de fábricas gigantes, economia de escala e força de trabalho, enfrentavam as repercussões da competitividade, tais como, vida curta do produto, demandas incertas, customização da produção, etc. Pouco depois, nesta mesma década, exigiu-se destas organizações produtos com melhor qualidade e maior confiabilidade. Na década de 70, em reposta às novas exigências do mercado, começaram a surgir os conceitos de manufaturas flexíveis (EVANS e HADDOCK, 1992; TINCKNELL e RADCLIFFE, 1996).

A convencional forma de manufatura se baseia em dois tipos de processos. O primeiro, com máquinas dedicadas, é apropriado para produção em grande escala de um único tipo de peça. Este processo especializado permite baixos custos, mas também inibe a flexibilidade. O segundo processo, com máquinas não integradas, é mais apropriado para produções em pequena escala de peças personalizadas. Neste segundo tipo de processo o custo por unidade tende a ser alto, mas a flexibilidade oferecida permite mudanças no *design* das peças, melhor atendimento às flutuações na demanda e mudanças no *mix* do produto. Hoje, os sistemas avançados de manufatura, tais como FMS (do inglês, *flexible manufacturing system*), oferecem uma terceira escolha: a união da flexibilidade com um baixo custo unitário (GERWIN, 1986).

Esta nova perspectiva pode ser melhor compreendida através de uma pesquisa *survey* realizada com um grande número de empresas da América do Norte, Europa e Japão. A pesquisa sobre estratégia em manufatura relatou que: “a próxima batalha competitiva será através da competência das manufaturas para superar o velho *trade-off* entre eficiência e flexibilidade” (MEYER *et. al*, 1989 apud GUPTA e GOYAL, 1992, p. 528). Desde então, os administradores da produção concordam com o quadro geral, onde os baixos custos e a alta qualidade não são mais garantias de sucesso. Nestas circunstâncias as manufaturas vêm, gradativamente, concentrando seus esforços na flexibilidade, como forma de vantagem competitiva (UPTON, 1995). Para Chen e Chung (1996), a flexibilidade, por estar diretamente relacionada com os custos, qualidade e tempo de produção, torna-se

a dimensão chave das empresas em relação às prioridades de competitividade. Chen e Chung (1996) ainda afirmam que a manufatura flexível foi desenvolvida para responder rapidamente e com custos competitivos às mudanças desconhecidas no mercado e na tecnologia de produção. Assim, tal importância e o desenvolvimento tecnológico, colocaram a flexibilidade, junto com qualidade e custos, como a principal meta a ser alcançada pelas manufaturas.

2.1. MANUFATURAS FLEXÍVEIS

As manufaturas flexíveis têm trazido significativa vantagem competitiva sobre a tradicional manufatura em lote, devido à redução dos níveis de estoque, melhoramento na utilização dos recursos organizacionais e reutilização das máquinas em sucessivas gerações de produtos. Além disso, a tecnologia também fornece muitos benefícios, tais como melhoria na qualidade do produto, menor tempo entre os processos e maior agilidade na resposta às mudanças do mercado (FENSTERSEIFER, 1990).

A seguir enumeram-se as principais vantagens fornecidas por uma manufatura flexível (GUPTA e GOYAL, 1989):

1. permite a execução do plano de manufatura em um prazo mais curto e com menor custo (maior competitividade);

2. mostra o inter-relacionamento das diversas etapas e operações do plano de manufatura;

3. permite a distribuição ótima dos recursos disponíveis e facilita a sua redistribuição em caso de modificações posteriores;

4. fornece diversas alternativas para a execução do plano de manufatura, facilitando a tomada de decisão a respeito;

5. identifica as tarefas ou operações críticas, ou seja, aquelas que não oferecem folga de tempo para sua execução. As tarefas ou operações críticas são aquelas que afetam diretamente o prazo para o término do projeto global, exigindo que a administração concentre nelas a sua atenção;

6. estabelece clara definição da responsabilidade de todos os setores de produção.

Pelos motivos apresentados, a flexibilidade tornou-se um dos objetivos chave de qualquer manufatura e um fator crítico de mensuração da performance produtiva. A flexibilidade garante que a manufatura possa ser simultaneamente customizada e eficiente em custos. Como o tempo de configuração (*setup*) diminui, pequenos lotes podem ser produzidos com os mesmos custos de manufaturas em economias de grande escala. Assim, a organização pode somar a sua estratégia competitiva a capacidade de atendimento personalizado e exclusivo, mesmo em mercados bastantes dinâmicos. Segundo Upton (1995), o valor competitivo da flexibilidade na manufatura baseia-se na sua capacidade de neutralizar os efeitos de demandas incertas em determinados mercados. Sob tais condições, a habilidade de um sistema de manufatura de responder apropriadamente a esta incerteza determinará a estabilidade e lucratividade da unidade de negócio, pois cabe ao sistema o exame das soluções possíveis e a decisão daquelas que prometem ter caráter ótimo com a máxima eficiência numa rede complexa de interações.

2.2. FLEXIBILIDADE NA MANUFATURA

A flexibilidade é essencial para lidar com as contingências num plano de manufatura, pois permite a produção de peças levando-se em consideração o comportamento estático e dinâmico do ambiente. Deste modo, é a flexibilidade que fornece à manufatura a capacidade de manter a execução da produção sob condições de mudanças e incertezas.

As mudanças podem ser externas ou internas à organização. As mudanças externas, como o próprio nome sugere, são aquelas oriundas do ambiente externo, portanto, geralmente assumem características mercadológicas. Estas mudanças influenciam na demanda e *mix* do produto, definem o progresso da tecnologia e a adoção de políticas estratégicas. Por sua vez, as mudanças internas relacionam-se com aspectos geralmente operacionais tais como, quebra de máquinas, falhas em equipamentos, variações no tempo de processamento, necessidade de incorporação de novas peças ao sistema, entre outros (BARAD e SIPPER, 1988; BARAD, 1992).

Como as mudanças afetam diretamente a produção, diversos autores definem flexibilidade como a habilidade de um sistema de manufatura enfrentar as circunstâncias de mudança ou instabilidade causadas pelo ambiente (GUPTA e GOYAL, 1992; KOCHIKAR e NARENDRAN, 1992; TINCKNELL e RADCLIFFE, 1996). Brill e Mandelbaum (1989) relacionam a flexibilidade com a capacidade do sistema de absorver as mudanças no ambiente de manufatura, tais como demanda por produtos, variação nos custos de

produção e venda, introdução de novas peças e ferramentas, entre outros fatores. Barad e Sipper (1990) definiram manufatura flexível como um sistema de produção adequadamente equipado e operado para responder, o mais rápido possível, a qualquer mudança ou distúrbio se uma decisão apropriada for tomada.

Independente do conceito adotado, é possível perceber através de suas definições, que a flexibilidade está diretamente associada à incerteza, uma vez que, o futuro não é deterministicamente previsível (Kochikar e Narendran, 1992). Deste modo, é possível afirmar que o objetivo de uma manufatura flexível é responder às incertezas decorrentes de mudanças internas e externas. Sob este ponto de vista, é possível associar cada incerteza a um tipo de flexibilidade, conforme exemplificado a seguir.

- a incerteza quanto à aceitação de produtos por parte dos clientes gera a necessidade de flexibilidade de mix do produto;
- a incerteza sobre os tempos de processamentos das máquinas gera a necessidade de flexibilidade de roteamento;
- a incerteza sobre a quantidade de produtos demandados gera a necessidade de flexibilidade de volume de produção; etc.

Para lidar com a incerteza, Browne *et al.* (1984) identifica oito tipos distintos de flexibilidade: máquinas, processos, produto, roteamento, volume, capacidade de expansão, seqüência e produção.

Flexibilidade de máquina: a flexibilidade de máquina refere-se aos vários tipos de operações que cada máquina pode executar de forma eficiente sem acarretar em um esforço proibitivo na troca de uma operação para outra (BROWNE, 1984; DAST e GENDRA, 1993; SARKER, 1994)

Flexibilidade de processo: é a habilidade de variar as etapas necessárias para completar uma tarefa. Isto permite que diferentes tarefas sejam completadas no sistema, usando uma variedade de máquinas.

Flexibilidade de produto: é a capacidade de acrescentar um novo produto ao sistema ou alterar o *mix* de produtos já existente, para atender às exigências do mercado (geralmente os produtos apresentam características comuns).

Flexibilidade de roteamento: é a habilidade de variar a rotina de produção dentro de um sistema em resposta as incertezas do ambiente (por exemplo, quebra de uma máquina).

Flexibilidade de volume: é a capacidade do sistema de se adaptar às flutuações de volume de produção de uma peça, modificando os ritmos e os tempos de operação e de ocupação das ferramentas (BORENSTEIN, 1991).

Flexibilidade de expansão: é a capacidade de construir um sistema e expandí-lo se necessário (geralmente encontrada em sistemas que comportam módulos e células).

Flexibilidade de seqüência: é a habilidade de alterar a ordem das operações em uma peça.

Flexibilidade de produção: é a capacidade de, economicamente e rapidamente, alterar a produção de uma peça por outra, dentro de um sistema de manufatura.

Segundo Browne *et al*. (1984), as flexibilidades de produto, processos e seqüências dependem da flexibilidade de máquinas, enquanto as flexibilidades de volume e expansão dependem da flexibilidade de roteamento. Assim, os trabalhos desenvolvidos nesta área abordam, essencialmente, as flexibilidades de máquinas e roteamento, e as mesmas podem ser consideradas como os principais tipos de flexibilidade, uma vez que são requisitos para as demais (CHEN e CHUNG, 1996).

Ambos os tipos de flexibilidade, de máquinas e de roteamento, definem como as peças fluem pelo sistema de manufatura. Deste modo, para que um ambiente de manufatura possa ser descrito como flexível, não deve possuir operações que sejam executadas em um único local, em uma única seqüência predefinida e com máquinas que se dediquem inteiramente a execução de um único tipo de operação (BORENSTEIN, 1999). De um modo geral, a flexibilidade pode ser definida como a aptidão para executar mais de uma operação numa determinada máquina e a possibilidade de permuta na ordem em que tais operações serão executadas.

Uma operação pode ser definida como a transformação ocorrida em uma peça visando alterar seu estado atual. Assim, uma peça é caracterizada por uma seqüência de operações que deve ser executada de acordo com um plano pré estabelecido (BORENSTEIN, 1999). Estas operações podem incluir passos como fresamento, torneamento, aquecimento, etc.

O espaço de estados de uma manufatura flexível cresce exponencialmente com o acréscimo de operações ou máquinas no sistema. Este crescimento é resultado da grande quantidade de alternativas de planos de produção, tornando a escolha deste plano um problema bastante complexo. Deste modo a grande quantidade de planos de produção

exige modelos ou algoritmos que trabalhem com representações eficientes do espaço de estados, para não comprometer a tratabilidade da flexibilidade.

Muitas são as formas de representação da flexibilidade e cada uma delas apresenta peculiaridades distintas. Estas peculiaridades influenciam na elaboração de heurísticas e na programação de algoritmos capazes de trabalhar com uma determinada representação, uma vez que os algoritmos que trabalham eficientemente com um tipo de representação podem ser ineficientes no tratamento de outras. Assim, entende-se que a agilidade na tomada de decisão em processos de manufatura depende da capacidade dos modelos desenvolvidos para representar a flexibilidade do sistema.

2.3. TRABALHOS ANTERIORES RELACIONADOS À REPRESENTAÇÃO DA FLEXIBILIDADE

Outros trabalhos já foram desenvolvidos utilizando diferentes representações por espaço de estados no tratamento de manufaturas flexíveis. Kusiak (1987), Yao e Pei (1987), Kusiak e Cho (1992) desenvolvem algoritmos que escolhem planos de produção para diferentes aplicações, tais como *lay-out* e seqüenciamento. Entretanto, estes algoritmos apenas escolhem os planos durante a execução da produção, sem enumerar todas as possíveis seqüências. A enumeração de todas as seqüências deveria ser executada antes da utilização dos algoritmos, para uma análise completa das alternativas de produção.

Hancock (1989) utiliza redes (*network*) para representar a flexibilidade em manufatura, onde cada nó indica uma máquina, cada arco indica o fluxo entre os nós, e cada rota representa uma possibilidade de processamento. Contudo, o modelo não explicita todas as seqüências de manufatura e não permite a distinção entre máquinas e operações. Para solucionar este problema, Lin e Solberg (1991) utilizam uma representação na forma de dígrafos AND-OR. Nesta representação, cada arco representa uma relação de precedência e cada nó uma operação numa determinada peça. Além da operação, cada nó é classificado como AND ou OR. Um nó AND significa que todas as operações “filhos” devem ser executadas, enquanto um nó OR indica que apenas uma das operações “filhos” deve ser executada. A desvantagem desta representação é não indicar as operações que podem ser executadas e as máquinas que estão disponíveis, podendo gerar planos de produção incoerentes com a situação do sistema.

Barad e Sipper (1990) apresentaram artigos que trabalham com representações em redes Petri para manufaturas flexíveis. Através das redes é possível avaliar a performance do sistema sob condições instáveis, tais como, falhas, flutuações na demanda, re-

roteamento, etc. Deste modo, as redes foram usadas para comparar sistemas distintos com o objetivo de auxiliar na escolha de flexibilidades para a manufatura. Todavia, o tamanho da rede Petri crescia rapidamente conforme aumentava a complexidade do sistema modelado. Assim, a representação em redes Petri pode se tornar impraticável devido à quantidade de recursos computacionais requeridos. Kochikar e Narendram (1993) também utilizaram redes Petri para representar a transição de estados de uma peça na manufatura. Contudo, esta representação foi desenvolvida apenas para análises qualitativas dos estágios de um sistema de manufatura, restringindo sua utilização prática.

Borenstein (1999) publicou um artigo utilizando árvores para representar a flexibilidade de roteamento em sistemas de manufatura. A representação em árvores permite o planejamento de operações e máquinas em diferentes condições de fabricação de uma peça, auxiliando na avaliação dos efeitos da flexibilidade num sistema de manufatura. Como nas redes Petri, a representação na forma de árvore utiliza considerável recurso computacional conforme se aumenta a complexidade da manufatura.

Kim e Egbelu (1999) trabalham com a flexibilidade em sistemas de manufatura utilizando programação linear. A modelagem realizada por estes autores apresenta um aumento exponencial no número de variáveis e restrições, conforme se aumenta o número de operações ou máquinas no sistema. Deste modo, o aumento exponencial verificado limita o uso da programação linear para problemas muito pequenos. Reveliotis (1999) também utilizou programação linear para roteamento em sistemas de manufatura. Seu trabalho leva em consideração o comportamento estático de um sistema de manufatura para definir políticas de roteamento. Estas políticas são utilizadas como critério de re-roteamento em caso de contingências enfrentadas pelo sistema. Entretanto, o modelo desenvolvido trabalha com um número limitado de seqüências de manufatura.

Embora as representações citadas nesta seção tenham utilidade e princípios interessantes, elas não estão aptas a incluir, simultaneamente, as características exigidas pelos sistemas flexíveis de manufatura. As características de uma representação eficiente no tratamento de manufaturas flexíveis estão enumeradas na próxima seção.

2.4. SITUAÇÃO PROBLEMÁTICA

Em virtude do que foi exposto nas seções anteriores, pode-se resumir a situação problemática da seguinte maneira:

a) o mercado tem exigido produtos de melhor qualidade, diferenciados e com custos competitivos;

b) o avanço da tecnologia permitiu o desenvolvimento de manufaturas flexíveis, para responder às mudanças impostas pelo mercado;

c) apesar das vantagens fornecidas pelas manufaturas flexíveis, a escolha de um plano de manufatura torna-se difícil, devido ao crescimento exponencial no número de planos de produção;

d) a complexidade do espaço de estados das manufaturas flexíveis exige representações adequadas para o sistema;

e) a escolha de uma representação adequada auxilia a tratabilidade de manufaturas flexíveis.

Em resposta, as representações devem apresentar algumas peculiaridades que são listadas a seguir:

- considerar o comportamento estático e dinâmico do ambiente, fornecendo à manufatura a capacidade de manter a produção sob condições de mudança e incerteza;

- capacidade de resposta em tempo real;

- possuir um ótimo desempenho computacional em virtude da complexidade do espaço de estados (pouca memória e tempo requerido pelo algoritmo);

- representar o espaço de estados completo de um sistema flexível de manufatura;

- apresentar custos acessíveis de desenvolvimento e implementação.

Infelizmente, nenhuma das representações descritas na seção 2.3 apresenta estas capacidades simultaneamente, justificando o desenvolvimento de uma nova representação.

2.5. JUSTIFICATIVA

Em virtude da crescente necessidade de manufaturas flexíveis e ausência de representações adequadas capazes de trabalhar com a complexidade destes sistemas, avaliou-se pertinente a construção de uma representação capaz de tratar este problema. A tarefa não é simples, uma vez que a construção deste espaço de estados é um problema considerado não polinomial (CAMPELLO e MACULAN, 1994).

Problemas não-polinomiais são aqueles cujo tempo de execução do algoritmo ou a memória computacional requerida, em função de sua entrada, não pode ser calculado por equações polinomiais. Desta forma, o tempo de execução ou memória requerida por um algoritmo estende-se, impossibilitando uma resposta imediata do sistema ou até mesmo a tratabilidade do problema (TOSCANI e VELOSO, 2001). Como os sistemas flexíveis são responsáveis pelo andamento e controle da produção, eles necessitam responder e atuar em tempo real. Esta particularidade do sistema exige representações compactas e eficientes do espaço de estados da manufatura.

A reposta em tempo real é fundamental, pois a escolha de um plano de manufatura depende da dinâmica da situação. Durante a produção de uma peça diversos eventos inesperados podem causar o desvio do curso de ação desenhado. Assim, se um evento inesperado ocorre (por exemplo, o dano em uma máquina), a nova seqüência escolhida dependerá do atual estágio de manufatura (BORENSTEIN, 1999). O comportamento dinâmico de um sistema de manufatura incorpora, dia-a-dia, eventos tais como: mudança de máquinas, manuseio de material, manutenção, etc.

Tais eventos aumentam a complexidade operacional e os custos de produção, devido à necessidade de transporte e reconfiguração do sistema. Estas peculiaridades implicam em sistemas de manufatura bastante complexos, tornando a escolha de um plano de manufatura um dos mais difíceis problemas decisórios (BORENSTEIN, 1999).

Deste modo, o desempenho computacional de uma manufatura flexível depende fortemente da representação escolhida para sua flexibilidade. A escolha desta representação determinará a eficiência e utilidade dos algoritmos de busca e heurísticas, capacitando ou não o sistema a responder em tempo real. Assim, a eficiência da representação é um dos fatores que indicará o tempo e a capacidade das manufaturas flexíveis de responder às mudanças no ambiente, tornando-se essencial para o desenvolvimento da mesma.

Contudo, apesar da existente diversidade de modelos, ainda são poucas as representações capazes de trabalhar com a flexibilidade na manufatura. Isso ocorre, porque uma representação adequada para tratar com uma determinada aplicação pode ser inadequada quando utilizada para outra finalidade (MELLO e SANDERSON, 1991). Ante o exposto, julgou-se oportuno o desenvolvimento de uma representação que apresente as características necessárias para trabalhar com a flexibilidade de seqüência - pouca necessidade de memória computacional e a capacidade de ser facilmente integrada com outros modelos analíticos e de simulação, direcionados para a análise e avaliação de sistemas flexíveis de manufatura.

2.6. OBJETIVOS

Os objetivos deste trabalho são:

a) Objetivo geral

Desenvolver uma representação, na forma normal disjuntiva (FND), para o espaço de estados de um sistema de manufatura com flexibilidade de seqüência, possibilitando sua aplicação para melhorar a escolha de um plano de produção, levando-se em consideração o comportamento estático e dinâmico do sistema.

b) Objetivos específicos

- desenvolver uma representação da flexibilidade de seqüência na forma de grafos direcionados acíclicos e na FND, estabelecendo a bijeção entre estas representações;

- desenvolver algoritmos computacionais capazes de transformar a representação na forma de grafos direcionados acíclicos em uma representação na FND;

- validar e avaliar os modelos e algoritmos desenvolvidos;

- demostrar as vantagens da representação na FND;

- exemplificar as possíveis aplicações da representação na FND.

3. METODOLOGIA DE PESQUISA

Neste trabalho é desenvolvida uma representação para a flexibilidade de seqüência na manufatura, com vistas a permitir a geração e análise de alternativas de planos de produção, tanto antes (*off-line*) como durante (*on-line*) o processo de fabricação. Assim, espera-se fornecer às manufaturas uma representação que trabalhe melhor com a flexibilidade de seqüência e permita a escolha da ação mais conveniente e eficiente a ser tomada. Isso significa que um sistema de manufatura poderá simular possíveis problemas e testar soluções alternativas, procurando responder as questões do tipo: o que acontecerá se (*What-if*)?

A pesquisa se apoiará nos preceitos metodológicos de pesquisa operacional para o desenvolvimento de um modelo matemático. Tal modelo caracteriza-se pela representação do espaço de estados utilizando-se de FNDs que, posteriormente, serão aplicadas em modelos lineares. Desta forma, espera-se melhorar o tratamento da flexibilidade de seqüência com o uso de uma representação mais apropriada e eficiente.

3.1. MÉTODO DE PESQUISA

Uma das abordagens matemáticas mais conhecidas para modelagem é a pesquisa operacional. O termo pesquisa operacional foi usado na segunda guerra mundial para descrever um método nascido de grupos interdisciplinares, que pretendia resolver problemas estratégicos e táticos de administração militar. Após a segunda guerra, este método tornou-se um tipo de abordagem comum na solução de problemas estruturados (SHAMBLIN e STEVENS, 1979).

A pesquisa operacional é, em si, considerada uma metodologia de pesquisa. De acordo com Andrade (1998), a pesquisa operacional é uma metodologia administrativa que agrega, em sua teoria, quatro ciências fundamentais para o processo decisório: economia, matemática, estatística e informática. Por apresentar estas características, neste trabalho foram aplicados os conceitos de pesquisa operacional para desenvolver os modelos capazes de trabalhar com a flexibilidade de seqüência em um sistema de manufatura. Para

tanto, foram necessárias três etapas distintas, a saber: contextualização da pesquisa, geração das representações do espaço de estados e, por fim, aplicação da representação desenvolvida. A Figura 1 resume a pesquisa, enquanto suas etapas são apresentadas a seguir.

Primeira Etapa - Contextualização

Nesta fase foi realizada a pesquisa bibliográfica que serviu de base para todo trabalho desenvolvido. A contextualização e o referencial conceitual já foram apresentados nos capítulos anteriores, enquanto alguns conceitos relacionados a representação ainda serão abordados no próximo capítulo. Além de fornecer o suporte teórico necessário, nesta fase foram definidas e selecionadas as representações para o espaço de estados. Por fim, alguns exemplos de peças manufaturadas foram coletados para posterior avaliação, mensuração e aplicação das representações desenvolvidas.

Segunda Etapa - Geração das Representações do Espaço de Estados

Uma das grandes vantagens na utilização de algoritmos é a facilidade de encontrar soluções onde se necessite tratar com restrições não lineares ou variáveis discretas. Assim, a segunda fase da pesquisa (capítulo 5) consiste no desenvolvimento de algoritmos computacionais capazes de gerar e trabalhar com as representações escolhidas na primeira fase. Deste modo foram desenvolvidos três algoritmos para representação do espaço de estados: o primeiro na forma matricial, o segundo em grafo e, por fim, o terceiro na FND. A representação na FND é gerada a partir do grafo e este, por sua vez, é derivado da matriz de precedência. Dando continuidade a pesquisa, os algoritmos desenvolvidos nesta etapa foram avaliados e validados conforme descrito na seção 5.4.

Terceira Etapa - Aplicação da Representação

Pela natureza da pesquisa foi necessário criar outros modelos que permitissem a aplicação e análise da representação na FND. Deste modo, nesta etapa foram criados modelos de programação linear, tornando perceptíveis as vantagens da representação. Como os modelos de programação linear baseiam-se em critérios de otimização, também foi preciso desenvolver um módulo capaz de gerar uma matriz de custos. O capítulo 6 apresenta a terceira etapa, seguido pelos resultados, contribuições e recomendações da pesquisa.

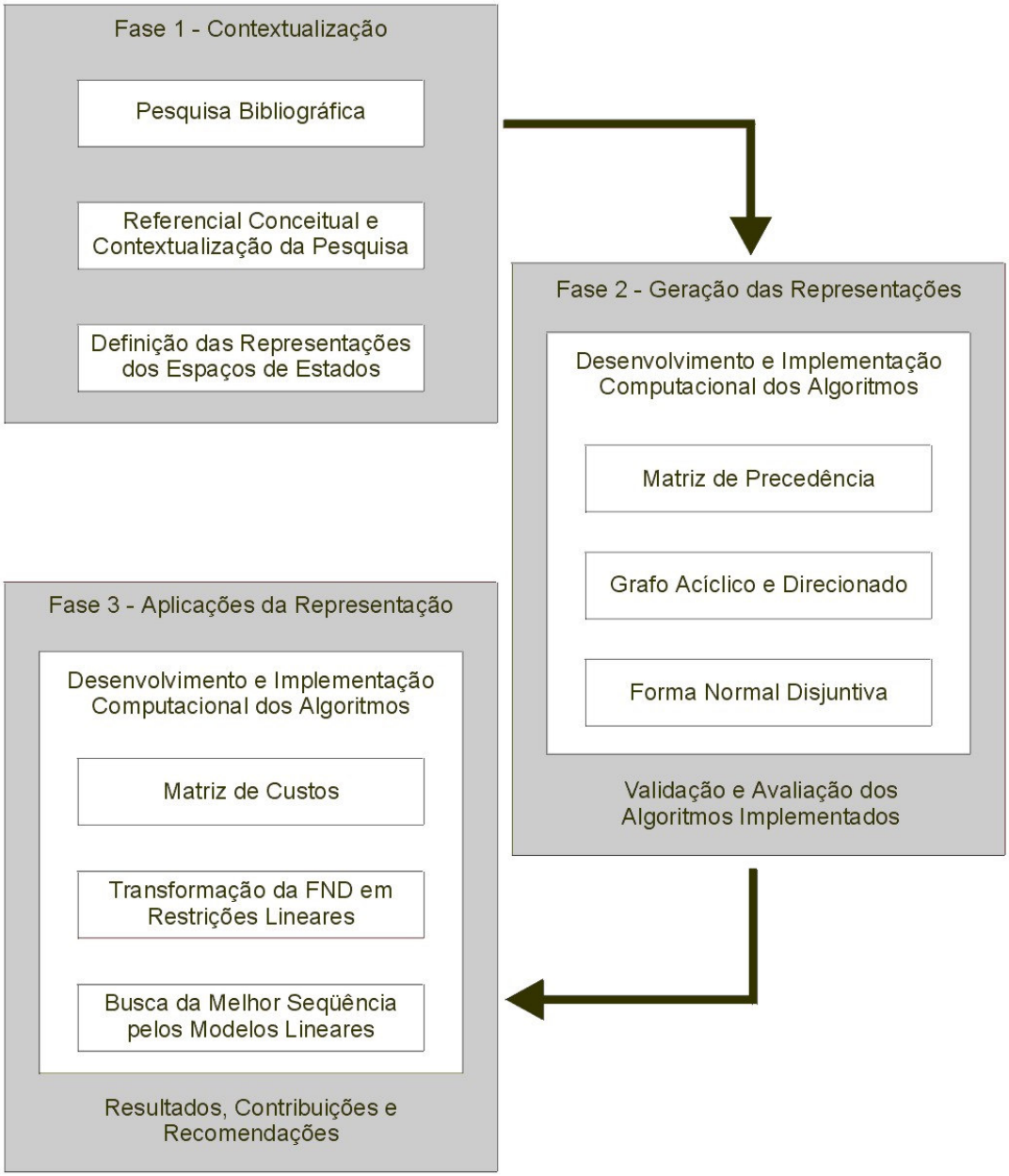

Figura 1 - Desenho da pesquisa.

4. CONCEITOS RELACIONADOS COM A REPRESENTAÇÃO

Segundo Rothenberg (1989), um modelo é a representação ou abstração da realidade para um determinado fim, uma vez que não pode representar todos os seus aspectos. Assim, os modelos permitem-nos tratar com o mundo de uma maneira simples, evitando a complexidade, os danos e a irreversibilidade da realidade (PIDD, 1998). Ackoff e Sasieni (1977) definiram modelos como representações da realidade e acrescentaram:

> Se os modelos fossem tão complexos e difíceis de controlar como a realidade, não haveria nenhuma vantagem em utilizá-los. Felizmente, podemos em geral construir modelos que são muito mais simples que a realidade e ainda assim conseguimos empregá-los para prever e explicar fenômenos com alto grau de precisão. A razão disso é que, embora seja necessário um grande número de variáveis para prever um fenômeno com exatidão perfeita, um pequeno número de variáveis explica geralmente a maior parte dele. O truque, evidentemente, é achar as variáveis certas e a relação correta entre elas (ACKOFF e SASIENI, 1977).

A modelagem é um dos principais processos da mente humana, é a nossa habilidade de pensar e imaginar, usando sinais e linguagens, para comunicar e tratar com o inesperado (ROTHENBERG, 1989). Deste modo, de uma maneira ou de outra, somos forçados a tratar com a complexidade dos sistemas em todas as áreas do conhecimento (BERTALANFFY, 1975). Este capítulo dedica-se a abordagem de alguns conceitos na área de modelagem. Primeiramente, o conceito de espaço de estados é apresentado e, após, são mostradas duas formas de representação: grafos e equações booleanas.

4.1. ESPAÇO DE ESTADOS

Um sistema é representado por um conjunto de elementos relacionados entre si para formar o todo. Estes elementos indicam algum fenômeno num estado do sistema, assim, os estados do sistema são percebidos nas características ou atributos assumidos por seus elementos (FLOOD e CARSON, 1993).

Os atributos dos elementos de um sistema são chamados de variáveis de estado (*state variables*). Como a maioria dos modelos trabalham com mais de um atributo, muitas vezes, diversas variáveis de estados são necessárias para determinar o estado de um

sistema num determinado momento. A este conjunto de variáveis de estados dá-se o nome de vetor de estado (*state vector*).

Os diferentes valores assumidos por um vetor de estados no tempo formam uma trajetória, denominada trajetória de estados. Esta trajetória mostra o comportamento do sistema no tempo, assumindo todos os estados pelo qual o sistema passou. A Figura 2 exemplifica os conceitos abordados nos dois últimos parágrafos.

$$X = \begin{vmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{vmatrix} \qquad X = \begin{vmatrix} x_1(t) & x_1(t+s) & x_1(t+2s) & \dots & x_1(t+ns) \\ x_2(t) & x_2(t+s) & x_2(t+2s) & \dots & x_2(t+ns) \\ x_3(t) & x_3(t+s) & x_3(t+2s) & \dots & x_3(t+ns) \\ \vdots & \vdots & \vdots & \dots & \vdots \\ x_n(t) & x_n(t+s) & x_n(t+2s) & \dots & x_n(t+ns) \end{vmatrix}$$

Vetor de Estado — Trajetória de Estados

Figura 2 - Vetor de Estado e Trajetória de Estados (FLOOD e CARSON, 1993).

Contudo, a trajetória de estados não inclui todas as possibilidades de estados que o sistema poderia assumir. O conjunto de todas as possibilidades, ou trajetórias, que um sistema pode assumir é denominado espaço de estados (*state space*).

A complexidade do espaço de estados ressalta a importância da escolha de sua representação, antes do desenvolvimento de um modelo ou de sua implementação computacional. Por esta razão a escolha da representação deve estar baseada em suas vantagens e desvantagens, levando-se em consideração o objetivo do modelo que será desenvolvido. Entre as representações mais escolhidas encontram-se os grafos, sentenças booleanas, matrizes, equações lineares, entre outras.

Muitas vezes, a geração de um espaço de estados torna-se bastante exaustiva, devido à sua representação e pelo crescimento exponencial característico de problemas não polinomiais. Nestes casos, algumas técnicas podem ser aplicadas para simplificar o espaço de estados (MORGAN e RAZOUK, 1987):

- seleção: consiste em concentrar-se numa área selecionada do espaço de estados;
- projeção: agrupa e pesquisa estados seguindo critérios predeterminados, preservando alguns aspectos do sistema e ignorando outros;
- redução: transforma o espaço de estados, obtendo um outro menor que preserve e exiba o mesmo comportamento do original.

4.2. TEORIA DOS GRAFOS

Entre os modelos de pesquisa operacional, encontram-se aqueles oriundos da teoria dos grafos. A teoria dos grafos surgiu em auxílio a problemas que envolvam propriedades estruturais ou topológicas de sistemas. Matematicamente, esta teoria liga-se à álgebra das matrizes e, em forma de modelos, à teoria dos sistemas (BERTALANFFY, 1975). Um grafo é um diagrama geralmente utilizado para representar um sistema e sua complexidade no mesmo tempo em que são mostradas e mensuradas as relações entre as variáveis quantitativas que o compõem (FLOOD e CARSON, 1993).

Da teoria dos grafos derivam as técnicas bastante utilizadas de planejamento e programação por redes nas atividades de construção civil e de montagem industrial. Os grafos são intensamente aplicáveis em projetos que envolvam várias operações e etapas, vários recursos, diferentes órgãos, prazos e custos mínimos, onde tudo deve ser articulado, coordenado e sincronizado da melhor maneira possível. Existe uma gama de conceitos referentes à teoria dos grafos, todavia, para os modelos desenvolvidos nesta dissertação os conceitos abordados nesta seção são satisfatórios.

O conceito de grafo é bastante simples e pode ser definido como uma série de vértices ou nós que são ligados por arestas. Estas arestas representam uma relação de interdependência entre os nós. Pela notação matemática, um grafo G = (V,A) é um conjunto não-vazio, onde V é o conjunto formado por elementos denominados *vértices*, e A é um conjunto de *arestas*. Uma aresta é um par não-ordenado (v_i,v_j), onde v_i e v_j são elementos de V. Eis um exemplo de grafo com a sua ilustração e notação matemática.

$V = \{v_1, v_2, v_3, v_4, v_5\}$

$A = \{a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8\}$

$a_1=(v_1,v_2)$, $a_2=(v_1,v_3)$, $a_3=(v_1,v_3)$, $a_4=(v_2,v_3)$, $a_5=(v_2,v_5)$, $a_6=(v_5,v_5)$, $a_7=(v_3,v_5)$, $a_8=(v_3,v_4)$

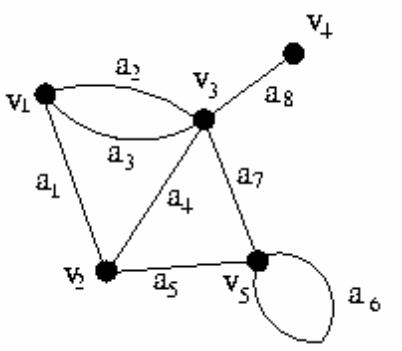

Figura 3 - Exemplo de grafo rotulado.

Os grafos fornecem um modelo confortável para a interpretação de situações reais (GOLDBARG e LUNA, 2000). Esta facilidade de interpretação é oriunda da capacidade de geração de caminhos dentro de um grafo. Um caminho ou rota, é o meio de mover-se de um nó a outro, sem reincidir qualquer nó. Assim, os problemas de roteamento consistem em encontrar caminhos entre dois ou mais nós que minimizem (ou maximizem) alguma medida de desempenho pré determinada pelas arestas do grafo (ACKOFF e SASIENI, 1977).

Contudo, para que problemas de otimização possam ser resolvidos, um grafo deve conter informações em seus vértices e arestas. Quando um grafo recebe atribuições (numéricas ou alfabéticas) associadas a seus vértices dizemos que este grafo é *rotulado*, enquanto que um grafo que recebe atribuições numéricas em suas arestas é chamado de *ponderado*. A Figura 3 é um exemplo de grafo rotulado e quando são associados valores numéricos a suas arestas, ponderado.

Os grafos ainda podem ser *direcionados*. Neste caso, o sentido das arestas é relevante para definição de um caminho. Geralmente as arestas direcionadas de um grafo são chamadas de *arcos*. Pela definição matemática, um grafo G = (V,A) é direcionado, se cada aresta (v_j,v_k) possui uma única direção de v_j para v_k. A aresta (v_j,v_k) é dita *divergente* de v_j e *convergente* a v_k. Quando as arestas de um grafo permitem a reincidência de um vértice, diz-se que este grafo é cíclico (ou em circuito), caso contrário este grafo será acíclico. A Figura 4 ilustra um grafo direcionado e cíclico.

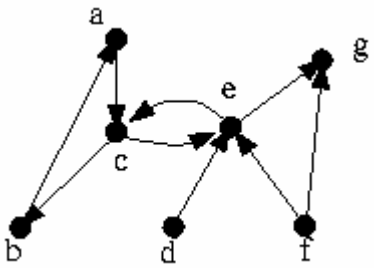

Figura 4 - Exemplo de grafo direcionado e cíclico.

4.3. FORMA NORMAL DISJUNTIVA

Uma sentença booleana é qualquer expressão formada por um conjunto finito de variáveis booleanas (também chamadas de *letras sentenciais*) onde são aplicadas as operações $\wedge$ (&), $\vee$ (ou) e $\neg$ (negação). Assim, a forma normal disjuntiva (FND) pode ser definida da seguinte maneira:

> Por literais, entendem-se as letras sentenciais A, B, C.... e as negações destas letras sentenciais ¬A, ¬B, ¬C... Por conjunção fundamental entende-se ou (i) uma literal, ou (ii) uma conjunção de duas ou mais literais, duas das quais nunca envolvem a mesma letra sentencial. Diz-se que uma conjunção fundamental A está incluída em outra B, se todas as literais de A são também literais de B. Assim, diz-se que uma forma sentencial A está na forma normal disjuntiva se, ou (i) A é uma conjunção fundamental, ou (ii) A é uma disjunção de duas ou mais conjunções fundamentais, nenhuma das quais está incluída na outra (MENDELSON, 1977).

Nesta definição, os quadros 1 e 2 exemplificam conjunções fundamentais e formas normais disjuntivas, respectivamente. É importante salientar que, diversos autores utilizam uma notação que suprime a operação ∧ (&), tornando, por exemplo, (A∧B∧C) igual a (ABC). Esta notação será utilizada em todos os exemplos e aplicações desta dissertação.

Conjunções fundamentais
A
¬B
AB
¬BAC
D¬AB

Quadro 1 - Exemplos de conjunções fundamentais.

Forma normal disjuntiva
B
¬C∨C
A∨(¬BC)
(A¬B)∨(¬A¬BC)
(B¬A)∨(¬A¬BD)∨A∨(BC¬D)

Quadro 2 - Exemplos de FNDs.

De maneira mais clara, a FND pode ser entendida como a melhor simplificação de uma equação booleana. Deste modo, basta verificar se, necessariamente, toda equação booleana tem uma FND logicamente equivalente. A prova de que, em álgebra booleana, toda forma sentencial que não seja uma contradição é, logicamente, equivalente a uma FND pode ser encontrada em Mendelson (1977). Uma contradição é uma forma sentencial que sempre resulta em falso. O exemplo a seguir, para a forma sentencial ((A∨B) ¬A ¬B), é uma contradição.

A	**B**	**A∨B**	**¬A**	**¬B**	**(A∨B)¬A**	**(A∨B)¬A¬B**
V	V	V	F	F	F	F
F	V	V	V	F	V	F
V	F	V	F	V	F	F
F	F	F	V	V	F	F

Quadro 3 - Exemplo de contradição.

Algumas sentenças booleanas podem apresentar mais de uma FND. Considere *K* sendo a sentença ((ABC)∨(ACD)∨(A¬BD)∨(¬AB¬C)∨(B¬CD)). As sentenças listadas a seguir são logicamente equivalentes a *K*, e também estão na FND.

(ABC)∨(A¬BD)∨(¬AB¬C)∨(B¬CD)
(ABC)∨(AD)∨(¬AB¬C)∨(B¬CD)
(ABC)∨(AD)∨(¬AB¬C)

Apesar das sentenças estarem na FND, a sentença ((ABC)∨(AD)∨(¬AB¬C)) é considerada a mais simples. A simplicidade de uma FND é determinada pelo número de literais e disjuntas em uma sentença, conforme se segue:

> Dada qualquer FND *K*. Seja Lk o número de literais de *K*, e Dk o número de disjuntas de *K*. Para as FNDs *K* e *G*, dizemos que *K* é mais simples do que *G* se, e somente se, Lg≥Lk e Dg≥Dk e, pelo menos uma destas igualdades é estrita (>). (MENDELSON, 1977)

A FND mais simples será chamada de FND mínima. No exemplo acima, ((ABC)∨(AD)∨(¬AB¬C)) é a FND mínima da sentença *K*. O processo de simplificação de uma sentença para sua equivalente na FND mínima pode passar por duas etapas.

Na primeira etapa uma sentença booleana qualquer é transformada para uma sentença logicamente equivalente de disjuntas (ou conjuntas). Esta transformação é realizada através da aplicação de um conjunto de leis (ou fundamentos) em álgebra booleana. O Quadro 4 lista as principais leis de transformação em álgebra booleana.

Sentença 1	Sentença 2	Lei
¬¬A	A	Dupla negação
AA	A	Idempotência
AB	BA	Comutatividade
A∨B	B∨A	Comutatividade
(AB)C	A(BC)	Associatividade
(A∨B) ∨C	A∨(B∨C)	Associatividade
¬(A∨B)	¬A¬B	De Morgan
¬(AB)	¬A∨¬B	De Morgan
A(B∨C)	(AB)∨(AC)	Distributivas
A∨(BC)	(A∨B)(A∨C)	Distributivas
A∨(AB)	A	Absorção
A(A∨B)	A	Absorção
(AB)∨¬B	A∨¬B	Absorção
(A∨B)¬B	A¬B	Absorção
A⇒B	¬B⇒¬A	Contrapositivo
A⇒B	¬A∨B	Eliminação de condicionais
A⇔B	(AB)∨(¬A¬B)	Eliminação de bicondicionais

Quadro 4 - Leis e fundamentos em álgebra booleana.

A segunda etapa para obtenção da FND consiste em encontrar os implicativos diretos da sentença disjunta (ou conjunta). Existem três métodos mais difundidos à obtenção de implicativos diretos: Quine-McCluskey, Consenso e Mapas de Karnaugh (MENDELSON, 1977). Os três métodos possuem características peculiares quanto às suas aplicações, todavia só descreveremos o de Quine-McCluskey, por ter sido o método utilizado nesta pesquisa. O método de Quine-McCluskey mostrou-se o mais apropriado, pois trabalha com conjunções completas, conforme veremos mais adiante na seção 5.2.

4.3.1. Método de Quine-McCluskey

A seguir são mostrados os passos do método de Quine-McCluskey (MENDELSON, 1977) para encontrar todos os implicativos diretos de uma equação booleana não tautológica. Considere a sentença $K = (G_1 \vee ... \vee G_n)$, onde G_i é uma conjunção de *K*, então:

Passo 1: enumera-se $G_1, ..., G_n$

Passo 2: se duas conjunções fundamentais G_i e G_j na enumeração são as mesmas, exceto que G_i contém uma certa letra não negada, enquanto G_j contém a mesma letra negada, acrescentemos à enumeração a conjunção fundamental obtida pela eliminação da letra de G_i em que G_i difere de G_j.

Passo 3: marca-se G_i e G_j.

Passo 4: repete-se o processo indicado nos passos 2 e 3 até que eles não possam mais ser aplicados. As conjunções fundamentais que já tiverem sido verificadas podem ser usadas de novo nas aplicações do passo 2.

As conjunções fundamentais não marcadas na enumeração resultante são os implicativos diretos de *K*. Uma vez obtidos todos os implicativos diretos de *K* é preciso descobrir que disjunções de implicativos diretos formam a FND mínima. Essa verificação pode ser feita com o uso de tabelas de implicativos diretos. O exemplo a seguir ilustra o método de Quine-McCluskey e a aplicação da tabela de implicativos diretos. Seja *K*, $(AB\neg C) \vee (A\neg BC) \vee (\neg ABC) \vee (\neg A\neg BC) \vee (\neg A\neg B\neg C)$. O primeiro passo pede que se enumere as conjunções de *K*.

$AB\neg C$
$A\neg BC$
$\neg ABC$
$\neg A\neg BC$
$\neg A\neg B\neg C$

O Passo dois acarreta em:

AB¬C		¬BC
A¬BC	✓	¬AC
¬ABC	✓	¬A¬B
¬A¬BC	✓	
¬A¬B¬C	✓	

Nota-se que, ao aplicar o passo dois a A¬BC e ¬A¬BC tem-se ¬BC, a ¬ABC e ¬A¬BC tem-se ¬AC, e por fim, a ¬A¬BC e ¬A¬B¬C tem-se ¬A¬B. Deste modo, os implicativos diretos de *K* são AB¬C, ¬BC, ¬AC e ¬A¬B, pois o passo dois não é mais aplicável.

Uma vez que os implicativos diretos (G_i) foram encontrados, deve-se colocá-los em uma coluna e as conjunções de *K* em uma linha, de modo que se construa um quadro. Logo em seguida, marca-se com uma cruz (X) todas as intersecções de uma linha correspondente a um implicativo direto G_i e uma coluna correspondente a uma conjunção de *K* de tal modo que a conjunção de *K* inclua G_i.

	AB¬C	A¬BC	¬ABC	¬A¬BC	¬A¬B¬C
AB¬C	X				
¬BC		X		X	
¬AC			X	X	
¬A¬B				X	X

Quadro 5 - Exemplo de seleção de implicativos diretos.

O passo seguinte consiste em selecionar as colunas que apresentam uma única marca (X), ou seja, selecionar os implicativos diretos que equivalem logicamente e exclusivamente a uma conjunção de *K*. Por exemplo, nota-se que apenas ¬BC corresponde a A¬BC, logo ¬BC precisa, obrigatoriamente, estar na FND mínima.

Os implicativos que necessariamente estão na FND mínima são chamados de núcleo da FND. No exemplo acima, todos os implicativos diretos fazem parte do núcleo, portanto temos (AB¬C) ∨ (¬BC) ∨ (¬AC) ∨ (¬A¬B) como FND mínima.

Muitas vezes, percebe-se que alguns implicativos diretos equivalem logicamente a mais de uma conjunção fundamental. Assim, nem sempre todos os implicativos diretos são necessários na obtenção da FND de uma sentença booleana qualquer. O Quadro 6, referente a outra sentença booleana, exemplifica a questão.

	ABCD	AB¬CD	AB¬C¬D	A¬BCD	¬ABCD	¬ABC¬D	¬AB¬CD	¬A¬B¬CD
BD	X	X			X		X	
¬A¬CD							X	X
¬ABC					X	X		
AB¬C		X	X					
ACD	X			X				

Quadro 6 - Exemplo de seleção de implicativos diretos.

Neste exemplo o implicativo direto BD não foi necessário, pois os implicativos pertencentes ao núcleo da FND foram suficientes para equivaler logicamente a todas as conjunções da sentença booleana original. Deste modo, obteve-se a seguinte FND mínima: $(\neg A \neg CD) \vee (\neg ABC) \vee (AB \neg C) \vee (ACD)$.

5. REPRESENTAÇÃO DA FLEXIBILIDADE DE SEQÜÊNCIA

Primeiramente, efetuou-se nesta pesquisa um estudo sobre sistemas de manufaturas. Nas seções 2.1 e 2.2 os conceitos de manufatura e flexibilidade são revistos, bem como sua importância para o planejamento da produção. Também foram revistos modelos para planejamento de manufatura desenvolvidos em trabalhos anteriores e as representações adotadas por alguns destes modelos, conforme apresentado na seção 2.3.

Esta seção define a representação do espaço de estados de uma manufatura, levando em consideração sua flexibilidade de seqüência. Na primeira parte são apresentadas as representações matricial, grafo e FND. A segunda parte é dedicada à implementação computacional das representações. Enfim, a última seção mostra a forma de validação dos algoritmos implementados.

5.1. CONSTRUÇÃO DA REPRESENTAÇÃO POR ESPAÇO DE ESTADOS

Manufaturas flexíveis são sistemas dinâmicos discretos e, por esta razão, a representação por espaço de estados é uma das técnicas mais apropriadas para analisar tais sistemas (KOCHICAR e NARENDRAN, 1993). Nesta abordagem, o sistema é representado pelos estados, aos quais correspondem as possíveis situações reais que a manufatura pode assumir. Contudo, a geração do espaço de estados de uma manufatura é considerada um problema NP, o que exige severo esforço computacional na construção de uma representação. Deste modo, a idéia por trás de cada representação é exigir o menor esforço computacional possível selecionando um mínimo de elementos para armazenar o espaço de estados, obtendo-se um modelo de dimensão reduzida que se aproxime do comportamento original do sistema (HWANG, *et al.*, 1991). A representação por espaço de estados também já foi utilizada com sucesso em outros trabalhos em diferentes áreas, tais como excitação do nervo (NEVIS e MAJUNDAR, 1977), sistemas dinâmicos lineares (ISIDORI e RUBERTI, 1976; HWANG, *et al.*, 1991) e processos de reprodução da fala (MORIKAWA e FUJISAK, 1984).

Esta pesquisa apresenta uma representação por espaço de estados para todas as seqüências de manufatura. A representação adotada na pesquisa desenvolvida trabalha com variáveis booleanas que formam conjunções fundamentais. Cada conjunção representa um estado de um sistema de manufatura que, posteriormente, será transformada para a FND. Antes de se obter a representação na FND, são necessárias três etapas conforme listado a seguir:

1) *primeira Etapa*: refere-se à construção do grafo de precedência (Figura 6);

2) *segunda Etapa*: consiste em montar uma matriz binária, baseando-se no grafo de precedência (Quadro 8);

3) *terceira Etapa*: consiste no desenvolvimento do grafo de seqüências de manufatura na forma booleana, caracterizado como uma representação por estados (Figura 7).

Para melhor entendimento das etapas desta pesquisa, foi utilizado um exemplo real de peça manufaturada. As etapas apresentadas a seguir baseiam-se na peça produzida pela Hidráulica Parker em uma de suas plantas no Brasil. A peça em questão é a engrenagem motriz de uma bomba hidráulica e pode ser visualizada na Figura 5.

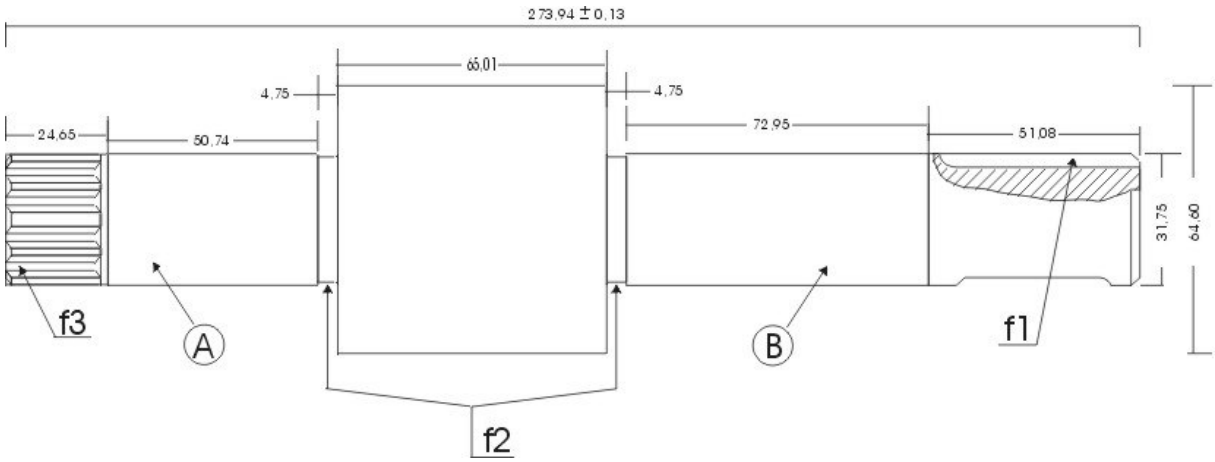

Figura 5 - Engrenagem motriz de uma bomba hidráulica.

Primeira Etapa - Grafo de Precedência

Nem todas as seqüências de manufaturas são possíveis devido às restrições geométricas e mecânicas de uma peça. Restrições geométricas referem-se ao fato de alguma operação ter que ser executada antes de outra, para evitar danos ou colisões em superfícies já trabalhadas em um componente. As restrições mecânicas referem-se ao esforço mecânico, exercido sobre um componente, que impede o processo de manufatura.

Para simplificar a pesquisa, o grafo de precedência que será construído inclui ambas as restrições de uma peça - geométricas e mecânicas.

Inicialmente, é preciso determinar quais operações necessitam obrigatoriamente ser realizadas antes de uma outra operação. A precedência de operação será indicada por uma flecha, onde sua raiz indica a operação que necessita ser executada com antecedência (precedente) e sua seta indica a operação que será executada posteriormente (procedente). Assim, é preciso que o engenheiro responsável pelo desenho da peça informe quais operações procedem as demais. O Quadro 7 mostra as especificações para a peça exemplificada na Figura 5.

Operação	Descrição	Op. precedentes
1	Corte da engrenagem	-
2	Aparar dentes	1
3	Serrilhar f2	-
4	Serrilhar f1	-
5	Serrilhar superfícies A e B	-
6	Tratamento de calor	2,3,4,5

Quadro 7 - Especificações para a engrenagem motriz.

O grafo de operações precedentes pode ser definido como um grafo direcionado (O,P), onde O = {1,2,...,n} é um conjunto de operações e P é um conjunto de precedentes definidos como $\mathrm{P} = \{(i,j) \in O\mathrm{x}O \mid i \; precede \; j\}$. A Figura 6 mostra o grafo de precedência onde a operação 2 só pode ser executada após a operação 1 e a operação 6 é, obrigatoriamente, a última a ser realizada. É importante enfatizar que no grafo de precedência existe sempre, no mínimo, uma operação que pode ser executada sem ter que esperar que outra operação seja completada.

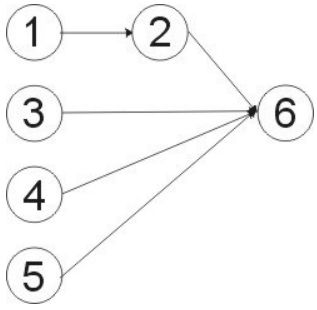

Figura 6 - Grafo de Precedência.

Segunda Etapa - Matriz de Precedência

A segunda etapa consiste na elaboração da matriz de precedência com o objetivo de facilitar o desenvolvimento ou reutilização de algoritmos computacionais. Os elementos matriciais são binários, pois a utilização de variáveis binárias reduz o espaço de memória

necessário para implementação de *softwares*. O Quadro 8 mostra a simplicidade da representação em uma matriz $A_{n \times n}$, onde A_{ij} é 1 se, e somente se, $(i,j) \in P$.

		Operação					
		1	2	3	4	5	6
Precedência							
1		0	1	0	0	0	1
2		0	0	0	0	0	1
3	A =	0	0	0	0	0	1
4		0	0	0	0	0	1
5		0	0	0	0	0	1
6		0	0	0	0	0	0

Quadro 8 - Matriz Binária de Precedência.

Ambas as representações, matriz e grafo de precedência, fornecem uma idéia singular da seqüência de manufatura, uma vez que não representam, simultaneamente, todas as possibilidades de seqüenciamento. Sem a representação completa do espaço de estados, não é possível planejar e analisar eficientemente as alternativas de manufatura, tanto antes (*off-line*) como durante (*on-line*) a produção.

Terceira Etapa - Grafo de Seqüências de Manufatura

Embora as representações anteriores consigam informar quais operações necessitam ser executadas com precedência, não são capazes de fornecer diretamente as possíveis seqüências de manufatura e, portanto, não capacitam a pesquisa de alternativas à produção. Assim, a terceira etapa transforma a matriz de precedência em um grafo capaz de representar todas as seqüências de uma manufatura. O objetivo desta representação é identificar "Qual será a próxima operação executada e por qual caminho?". A resposta para esta questão permite a construção de todas as possíveis seqüências de manufatura.

O grafo de seqüência é formado por nós que representam os estados discretos do sistema e arcos que representam a transição entre estes estados. Deste modo, as seqüências de operações na manufatura são obtidas enumerando-se todos os caminhos desde a raiz até as folhas do grafo. Neste processo, uma operação é sempre facilmente identificada através dos nós e dos arcos, possibilitando alternar as seqüências de manufatura de uma determinada peça levando em consideração suas restrições e especificações. A Figura 7 mostra o grafo de seqüências referente à matriz binária construída na segunda etapa. Este grafo (V, A) será acíclico e direcionado (*directed acyclic graph – DAG*). V é um conjunto de estados onde, $V=\{v_1, v_2, v_3 \ldots v_n\}$, e A é a transição entre estes estados onde, $A=\{a_1, a_2, a_3 \ldots a_n\}$. Os estados $\mathbf{v}_i$ e as arestas a_i podem ser definidas como:

$$v_i = \{(x_1 x_2 ... x_n) \mid x_j \in \{0,1\}\}$$

$$a_i = \{(v_1, v_2) \in V x V \mid [v_2 - v_1 \geq \mathbf{0}] \wedge [|v_2| - |v_1| = 1]\}$$

Onde:

$$x_j = \begin{cases} 1 \text{ se operação } j \text{ foi realizada} \\ 0 \text{ se operação } j \text{ não foi realizada} \end{cases}$$

$$|v_i| = \sum_{j=1}^{n} x_j$$

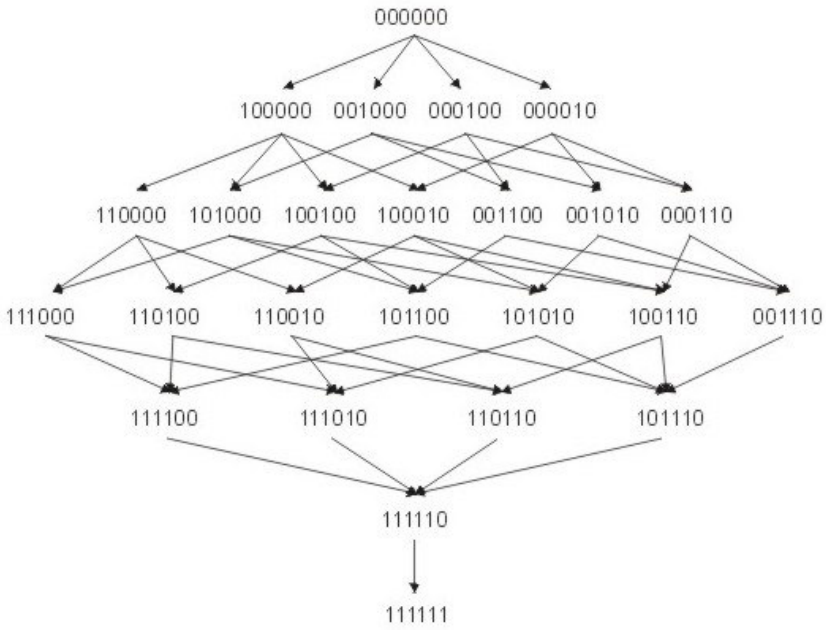

Figura 7 - Grafo de Seqüências de Manufatura.

O grafo de seqüências mostra todas as possibilidades de manufatura. Entretanto, como mencionado anteriormente, a geração de todas as possíveis seqüências de manufatura é um problema NP. Deste modo, esta representação torna-se inviável devido ao considerável esforço computacional necessário, no tempo e memória, para geração do espaço de estados. Obviamente, o aumento no número de operações aumenta exponencialmente o número de estados do grafo e, consequentemente, o tempo necessário para sua geração. Na seção 5.3 o tempo gasto na construção de cada representação é abordado com mais detalhes.

Para contornar o problema de memória computacional requerida pelo grafo de seqüências, é possível interpretar cada estado da manufatura como uma conjunção fundamental. Desta maneira, todo o grafo pode ser representado por uma equação booleana na FND. A representação na FND é abordada na próxima seção.

5.2. REPRESENTAÇÃO DO ESPAÇO DE ESTADOS USANDO FND

Esta seção mostra a transformação de todos os estados de manufatura, representados por conjunções fundamentais, em uma única equação na FND. A sentença booleana G e a sua FND (fnd_G), exemplificadas a seguir, representam todos os possíveis estados de manufatura (v_i) para a engrenagem da Figura 5.

$$G = (v_1) \vee (v_2) \vee (v_3) \vee ... \vee (v_n)$$

$$G = (000000) \vee (100000) \vee (001000) \vee (000100) \vee (000010) \vee (110000) \vee (101000) \vee (100100) \vee (100010) \vee (001100) \vee (001010) \vee (000110) \vee (111000) \vee (110100) \vee (110010) \vee (101100) \vee (101010) \vee (100110) \vee (001110) \vee (111100) \vee (111010) \vee (110110) \vee (101110) \vee (111110) \vee (111111)$$

$$fnd_G = (\neg B \neg F) \vee (A \neg F) \vee (ABCDEF)$$

As sentenças G e fnd_G também podem ser representadas por letras sentenciais, onde cada letra sentencial corresponde a uma operação na manufatura. Deste modo, a letra A corresponde à primeira operação, B à segunda, C à terceira, e assim sucessivamente.

A equação na FND foi obtida através do algoritmo de Quine-McCluskey, por se mostrar o mais apropriado dentre os três métodos citados na seção 4.4. Por ser um método gráfico, Mapas de Karnaugh é conveniente para obtenção de FNDs que envolvam até seis letras sentenciais. Sendo assim, seu uso é basicamente ilustrativo e didático, mostrando-se desapropriado para os objetivos desta dissertação.

Os dois métodos restantes, Quine-McCluskey e Consenso, trabalham com conjunções fundamentais distintas. Enquanto o método do consenso consegue encontrar a FND a partir de conjunções fundamentais incompletas, o de Quine-McCluskey necessita de conjunções fundamentais completas, ou seja, cada conjunção fundamental deve apresentar todas as letras sentenciais. Conforme pode ser observado na sentença G, os estados de manufatura são conjunções fundamentais completas. Deste modo, o método de Quine-McCluskey mostrou-se o mais adequado para transformar o grafo de seqüências em uma equação na FND. Uma vez encontrada a FND, é possível identificar um estado de manufatura onde:

$$v_i \in G \Leftrightarrow fnd_G (v_i) = 1$$

O operador $fnd_G(v_i)$ corresponde à aplicação de um estado v_i em uma sentença *G* na FND (fnd_G). Como a FND é capaz de representar todo o espaço de estados do sistema, e o

operador $\mathrm{fnd}_G(\mathrm{v}_i)$ verifica se um estado de manufatura é ou não válido, então elimina-se a necessidade de uma representação na forma de grafo. De maneira mais simples, podemos interpretar que as arestas do grafo de seqüências foram substituídas por um equação booleana. Assim, uma seqüência de manufatura, utilizando a representação na FND, pode ser definida por uma série de estados v_i aplicados no operador $\mathrm{fnd}_G(\mathrm{v}_i)$.

Apesar da representação na FND ser mais eficiente computacionalmente no que tange a memória requerida, julgou-se pertinente compará-la, e tecer alguns comentários, com a representação na forma de grafo. A Tabela 1 compara, em 80 exemplos, a quantidade de estados necessários para cada representação, levando em consideração o número de operações e precedências de diversas peças. A comparação é realizada em quantidade de estados, pois cada conjunção de uma FND pode assumir a forma vetorial, equivalendo em tamanho a um vértice da representação em grafo (Quadro 9).

Exemplos de conjunções como literais	**Exemplos de conjunções como vetores**	**Exemplos de vértices no Grafo**
¬B¬F	-0---0	100000
A¬F	1----0	110000
ABCDE	11111-	111110

Quadro 9 - Exemplos de conjunções e vértices com o mesmo tamanho.

Através da avaliação da Tabela 1 é possível perceber a redução 96,3% de memória utilizada pela FND em relação ao grafo. Cabe ressaltar que, quanto maior o número de operações em uma peça, maior será a redução de memória em relação ao grafo. Isso ocorre devido ao aumento exponencial na memória utilizada pelo grafo, o que não é verificado com a FND. Assim, ao contrário do grafo, quanto maior o número de possíveis estados assumidos por uma peça, menor será a necessidade de memória ocupada pela FND.

Particularmente, quando não existirem precedências em uma peça, a FND ocupará apenas a memória equivalente a um único estado no grafo. O crescente ganho de memória também pode ser percebido na Tabela 1, onde a representação na FND apresentou uma redução de aproximadamente 97,9% de memória, em relação ao grafo, para as peças com mais de nove operações.

Tabela 1 - Memória requerida em cada representação.

Nº de Op. e Prec.	Qtd Mem. Grafo	Qtd Mem. FND	Red. Mem. (%)	Nº de Op. e Prec.	Qtd Mem. Grafo	Qtd Mem. FND	Red. Mem. (%)	Nº de Op. e Prec.	Qtd Mem. Grafo	Qtd Mem. FND	Red. Mem. (%)
3-0	8	1	87,5	6-8	19	3	84,2	9-0	512	1	99,8
3-1	6	2	66,7	6-15	7	4	42,9	9-1	384	2	99,5
3-2	5	2	60,0	7-0	128	1	99,2	9-2	288	4	98,6
3-3	4	2	50,0	7-1	96	2	97,9	9-3	216	8	96,3
4-0	16	1	93,8	7-2	72	4	94,4	9-4	192	4	97,9
4-1	12	2	83,3	7-3	60	4	93,3	9-5	144	8	94,4
4-2	10	2	80,0	7-4	45	8	82,2	9-6	120	6	95,0
4-3	8	2	75,0	7-5	42	6	85,7	9-7	120	6	95,0
4-4	7	3	57,1	7-6	35	6	82,9	9-8	90	12	86,7
4-5	6	2	66,7	7-7	30	6	80,0	9-9	144	3	97,9
4-6	5	3	40,0	7-8	27	6	77,8	9-10	68	6	91,2
5-0	32	1	96,9	7-9	26	5	80,8	9-13	42	7	83,3
5-1	24	2	91,7	7-10	21	7	66,7	9-36	10	5	50,0
5-2	20	2	90,0	7-21	8	4	50,0	10-0	1024	1	99,9
5-3	15	4	73,3	8-0	256	1	99,6	10-1	768	2	99,7
5-4	12	4	66,7	8-1	192	2	99,0	10-2	576	4	99,3
5-5	12	3	75,0	8-2	160	2	98,8	10-3	432	8	98,1
5-6	10	3	70,0	8-3	120	4	96,7	10-4	384	4	99,0
5-10	6	3	50,0	8-4	96	4	95,8	10-5	288	8	97,2
6-0	64	1	98,4	8-5	80	4	95,0	10-6	240	6	97,5
6-1	48	2	95,8	8-6	60	8	86,7	10-7	240	6	97,5
6-2	40	2	95,0	8-7	55	8	85,5	10-8	180	12	93,3
6-3	30	4	86,7	8-8	54	6	88,9	10-9	288	3	99,0
6-4	25	4	84,0	8-9	54	5	90,7	10-10	136	6	95,6
6-5	24	3	87,5	8-10	45	8	82,2	10-13	84	7	91,7
6-6	19	5	73,7	8-11	38	5	86,8	10-45	11	6	45,5
6-7	18	3	83,3	8-28	9	5	44,4	Soma	9302	341	96,3

Nº de Op. e Prec.: número de operações e restrições em uma peça (op.-prec.).
Qtd. Mem. Grafo: quantidade de memória requerida pelo grafo de seqüências (em vetores).
Qtd. Mem. FND: quantidade de memória requerida pela FND (em vetores) .
Red. Mem.: redução da memória em relação ao grafo de seqüências (em percentual).

A eficiência computacional de uma representação, que pode ser medida em tempo e memória, está diretamente relacionada com a complexidade de seu algoritmo. Fica claro que a vantagem da representação na FND, em relação a outras representações, é a redução da complexidade de memória. Todavia, o mesmo não ocorre com a complexidade de tempo.

Se utilizarmos os algoritmos atualmente conhecidos para encontrar a FND, o tempo gasto nesta tarefa aumentaria exponencialmente com o aumento no número de variáveis booleanas. Assim, o problema só pode ser contornado com o desenvolvimento de um novo algoritmo mais eficiente ou de menor complexidade. Além disso, utilizando os mesmos algoritmos, é preciso enumerar todos os estados de uma manufatura para encontrar a FND. Sabemos que para enumerar todos os estados é necessário gerar o grafo de seqüências, o que inviabilizaria, na prática, a representação na FND. Deste modo foi preciso desenvolver um novo algoritmo que encontrasse a FND eficientemente e sem a necessidade de gerar o

grafo de seqüências. Este novo algoritmo será abordado com maiores detalhes na próxima seção (Novo Algoritmo para a FND).

5.3. IMPLEMENTAÇÃO COMPUTACIONAL DAS REPRESENTAÇÕES

Para execução desta etapa do trabalho foram implementados algoritmos capazes de encontrar a FND dos estados de uma manufatura, levando-se em consideração sua flexibilidade de seqüência. Os algoritmos foram implementados para o sistema operacional Windows 95/98/Me utilizando a linguagem DELPHI (Object Pascal). A linguagem DELPHI foi escolhida, pois proporciona o trabalho com ponteiros e *dynamic array*, facilitando a programação de grafos e matrizes. Um ponteiro é uma variável que contém um endereço da memória computacional, possibilitando armazenar diversos valores para uma única variável declarada no código (ALBUQUERQUE, 1991). O *dynamic array* pode ser compreendido como uma variável vetor que pode ter seu tamanho alterado em tempo de execução (REISDORPH, 1999).

Cada algoritmo foi desenvolvido e programado em módulos independentes que, posteriormente, foram agrupados em um único módulo (Anexos A e B). Assim, a partir do módulo principal (Figura 8) é possível acessar qualquer uma das etapas desta pesquisa. O computador utilizado na programação, testes e execuções dos algoritmos apresenta as seguintes características: processador AMD K6-2 com freqüência de 500 Mhz, memória RAM de 128 *megabytes*, sistema operacional Windows 98 SE.

Figura 8 - Interface do Módulo Principal.

Módulo Matriz de Precedência

O primeiro módulo foi desenvolvido para a obtenção da matriz de precedência, conforme visualizado na Figura 9. A matriz de precedência gerada neste módulo, posteriormente, será utilizada na construção do grafo de seqüências de manufatura. O código criado encontra-se no Anexo C e seu algoritmo é descrito a seguir.

Implementation

Procedure Botão (operação precedente);
Begin
escolha operação precedente;
escolha operação procedente;
matpreced[operação precedente,operação procedente]**:=true**;
mostrar matriz;
end;

Begin
escolha o número de operações na peça;
End.

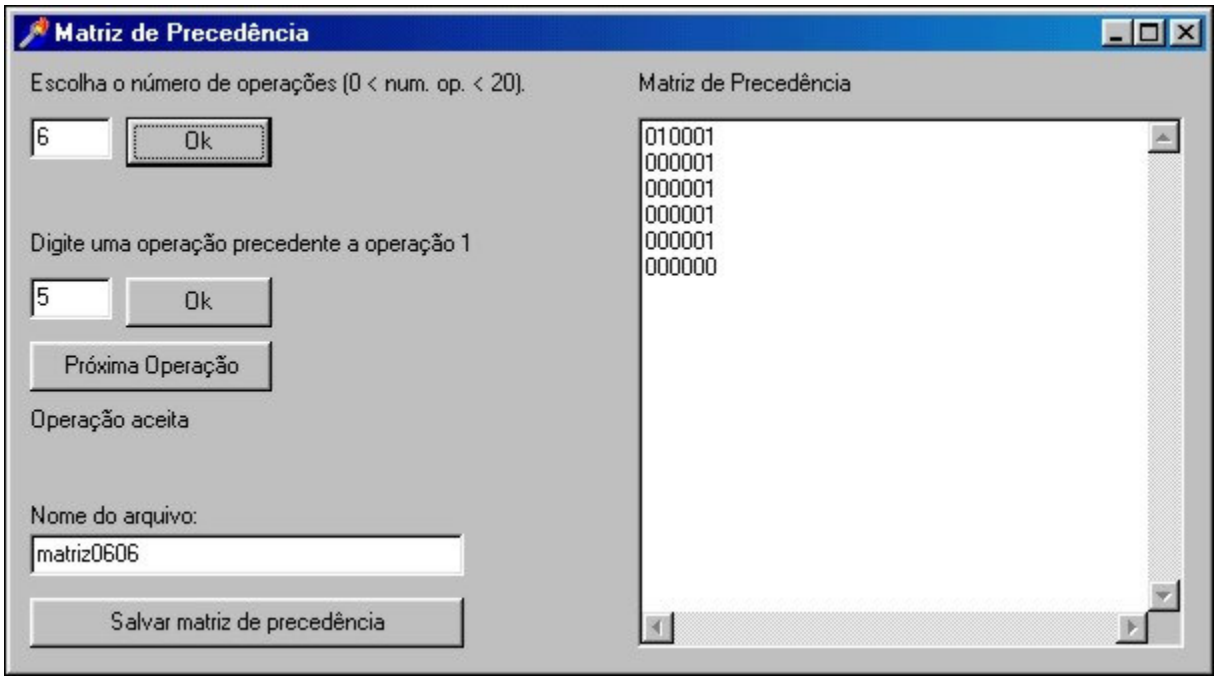

Figura 9 - Interface do Módulo Matriz de Precedência.

Módulo Grafo

O segundo módulo (Anexo E) é dedicado à construção do grafo de seqüências de manufatura. Este módulo lê o arquivo onde está armazenada a matriz de precedência e a partir dela gera todos os possíveis estados de manufatura, conforme pode ser observado na Figura 10.

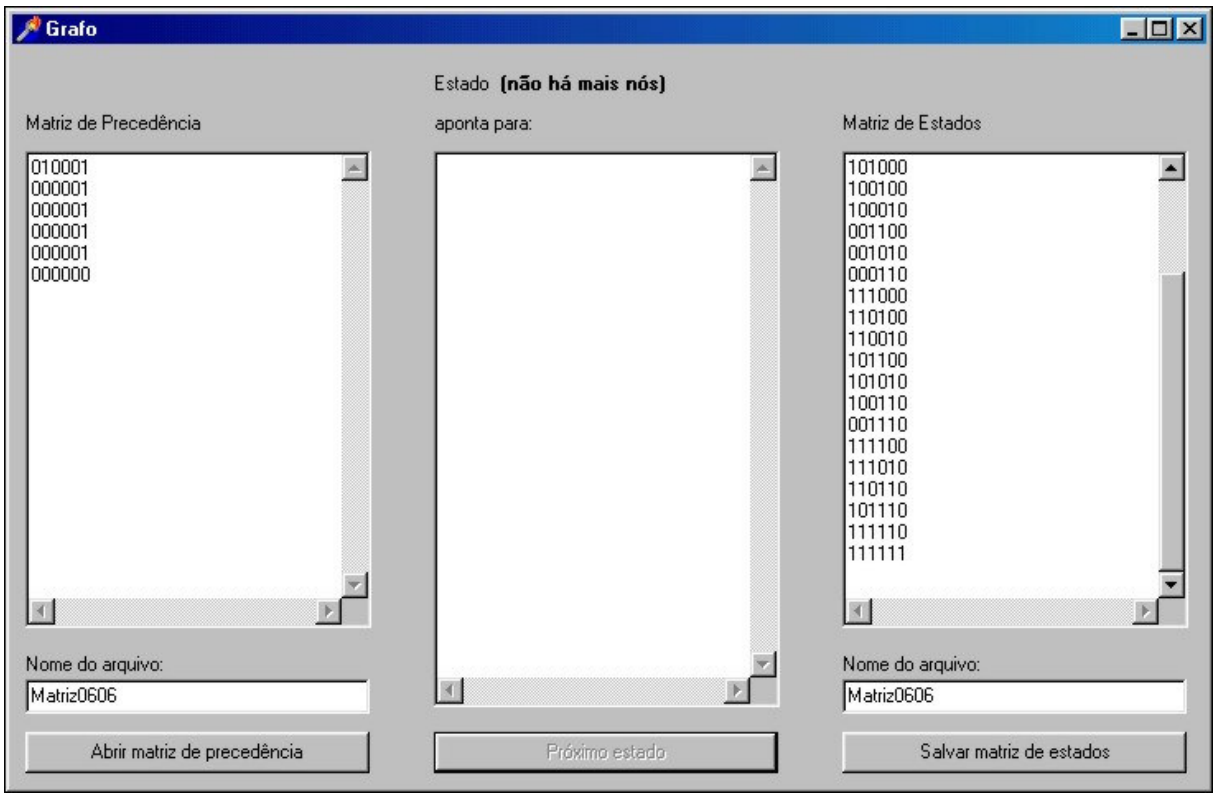

Figura 10 - Interface do Módulo Grafo.

Os estados de manufatura são gerados a partir de um estado inicial onde todos os valores são zeros, ou seja, nenhuma operação foi realizada. A partir daí, os novos estados são sucessivamente percorridos até que se obtenha todos os possíveis estados de manufatura. Os possíveis estados de manufatura a partir de outro estado, são obtidos através da equação e do procedimento descritos a seguir.

$$\mathbf{u} = A^T \times \neg\mathbf{v} + \mathbf{v}$$

Para i=1 até n
  Se u_i = 0 então
    Início
    j:=j+1;
    $\mathbf{w}^j$:=**v**;
    $w^j{}_i$:= 1;
    fim.

Onde:

$\mathbf{u} = \{(u_1 u_2 ... u_n) \mid u_i \in \{0,1\}\}$, é o vetor binário solução;

$\mathbf{v} = \{(x_1 x_2 ... x_n) \mid x_i \in \{0,1\}\}$, é o vetor binário do estado atual da manufatura;

$\mathbf{w}^j = \{(w^j{}_1 w^j{}_2 ... w^j{}_n) \mid w^j{}_i \in \{0,1\}\}$, é o j-ésimo vetor binário oriundo do vetor **v**;

A é a matriz de precedência.

Pelo procedimento apresentado, os possíveis estados de manufatura são obtidos através dos valores falsos (zero) do vetor **u**. Deste modo, a quantidade de novos estados de manufatura originários do vetor **v** é igual a quantidade de zeros encontrada no vetor **u**. A Figura 11 exemplifica a equação e o procedimento desenvolvido.

(A^T)						x	$(\neg \mathbf{v})$	+	$(\mathbf{v})$	=	$(\mathbf{u})$	$(\mathbf{w}')$
0	0	0	0	0	0		0		1		1	
1	0	0	0	0	0		1		0		0	$\rightarrow$ 111000
0	0	0	0	0	0	x	0	+	1	=	1	
0	0	0	0	0	0		1		0		0	$\rightarrow$ 101100
0	0	0	0	0	0		1		0		0	$\rightarrow$ 101010
1	1	1	1	1	0		1		0		1	

A é a matriz de precedência do Quadro 8; **v** = [101000]

Figura 11 - Exemplo de estados gerados a partir de outro estado.

Com um procedimento capaz de definir os estados oriundos de outro estado, foi elaborado um algoritmo capaz de gerar a representação na forma de grafo. Todavia, o comportamento exponencial deste algoritmo dificulta a utilização da representação em tempo real. A seguir é apresentado o algoritmo para a construção do grafo de seqüências, logo após, a Tabela 2 mostra, em alguns exemplos, o tempo gasto neste processo.

Implementation

```
    Procedure (Encontrar novos estados);
    Begin
    vetorresposta := (not(vetoratual)) x matpreced) + vetoratual;
    end;

    Procedure (Gerar novo estado);
    Begin
    se existem valores falsos no vetorresposta então gerar novos estados;
    verifica se cada novo estado já existe na matriz de estados;
    se novo estado não existe na matriz de estados então
        acrescentar novo estado na matriz de estados;
    end;

Begin
abrir arquivo contendo matpreced; {matpreced é a matriz de precedência};
vetoratual := [000...000];
primeiro vetor da matriz de estados := vetoratual;
    Repeat
    Encontrar novos estados;
    Gerar novo estado;
    vetoratual := próximo vetor da matriz de estados;
    until vetoratual = ultimo estado na matriz de estados;
salvar arquivo com a matriz de estados;
End.
```

Tabela 2 - Tempo de construção do grafo de seqüência.

Número de Operações	Quantidade de Precedências	Quantidade de Estados	Tempo Necessário
9	1	384	3 s
10	1	768	6 s
11	1	1536	12 s
12	1	3072	24 s
9	0	512	4 s
10	0	1024	8 s
11	0	2048	16 s
12	0	4096	32 s

Módulo FND

Este módulo lê o arquivo onde está armazenado o espaço de estados de um sistema de manufatura e encontra sua FND, conforme ilustrado na Figura 12. Conforme mencionado na seção anterior, o algoritmo implementado foi o de Quine-McCluskey. O código completo deste módulo encontra-se no Anexo F e o algoritmo de Quine-McCluskey é apresentado na seção 4.4.1.

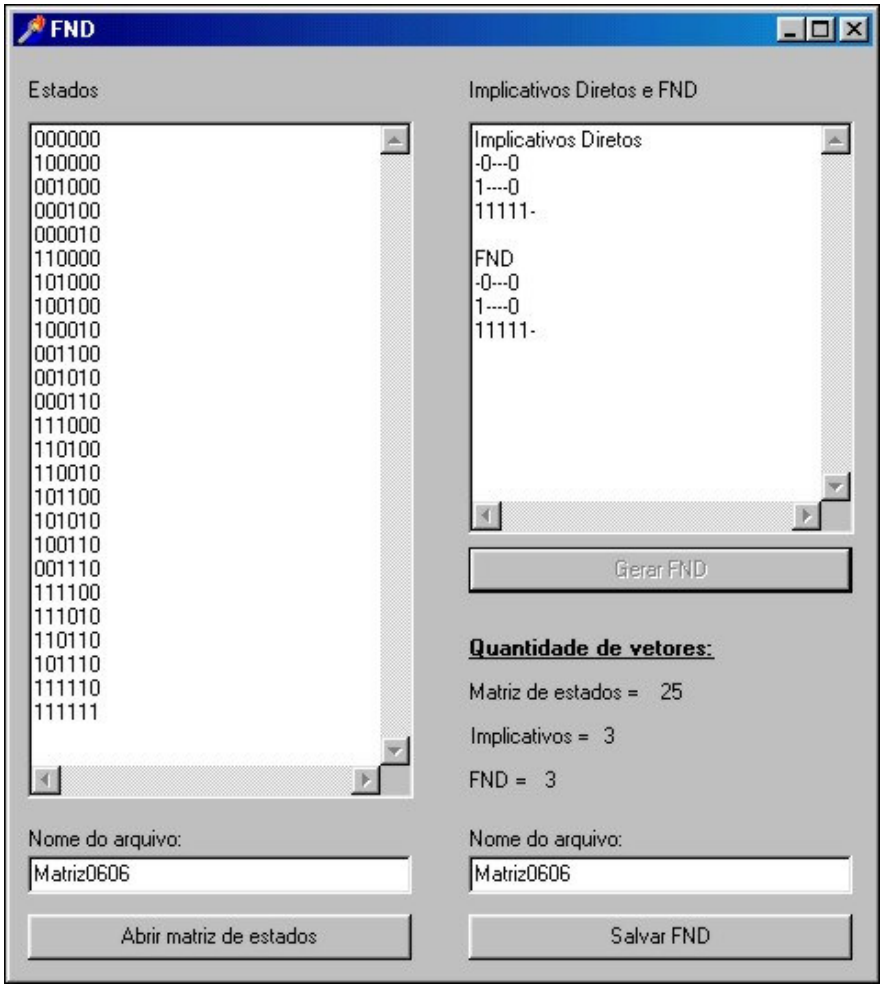

Figura 12 - Interface do Módulo FND.

Apesar dos algoritmos existentes (Quine-McCluskey, Karnaugh e Consenso) encontrarem corretamente a forma normal disjuntiva de uma equação, eles não são eficientes computacionalmente. A ineficiência deve-se ao aumento exponencial no tempo necessário para que estes algoritmos encontrem a FND de uma equação, conforme se aumenta o número de conjunções fundamentais (estados). O aumento exponencial pode ser verificado na Tabela 3. Deste modo, julgou-se oportuno o desenvolvimento de um novo algoritmo que encontrasse a FND sem a necessidade de gerar os estados de uma manufatura, ou seja, sem precisar gerar o grafo de seqüência.

Tabela 3 - Tempo de processamento do algoritmo de Quine-McCluskey.

Número de Operações	Quantidade de Precedências	Quantidade de Estados	Tempo Necessário
8	0	256	6,5 s
9	0	512	1 min e 7 s
10	0	1024	11 min e 9 s
8	1	192	2,9 s
9	1	384	20,8 s
10	1	768	3 min e 26 s
8	2	160	1,2 s
9	2	288	6,7 s
10	2	576	1 min e 4 s
9	3	216	2,5 s
10	3	432	21,3 s

Módulo Novo Algoritmo para a FND

O novo algoritmo (Figura 14) consegue encontrar a FND diretamente da matriz de precedência, sem a necessidade de combinar todos os estados de uma manufatura (conjunções fundamentais). Como o algoritmo não precisa combinar os estados de manufatura, não existe a necessidade de gerar o grafo de seqüências.

A FND é obtida de comparações entre as colunas da matriz de precedência, resultando em um algoritmo muito mais eficiente. Apesar do novo algoritmo também apresentar complexidade exponencial, a quantidade de colunas da matriz de precedência em relação ao número de estados do grafo de seqüências é muito pequena, o que aumenta a utilidade prática do algoritmo desenvolvido.

Estas características permitem que a FND seja encontrada em tempo real, tornando esta representação do espaço de estados apropriada para o comportamento dinâmico das manufaturas flexíveis. Nos exemplos das Tabelas 2 e 3, o novo algoritmo encontrou as FNDs num tempo considerado desprezível, ou seja, próximo de zero segundos. A Figura 13 compara o tempo gasto por cada algoritmo na construção da representação para o espaço de estados.

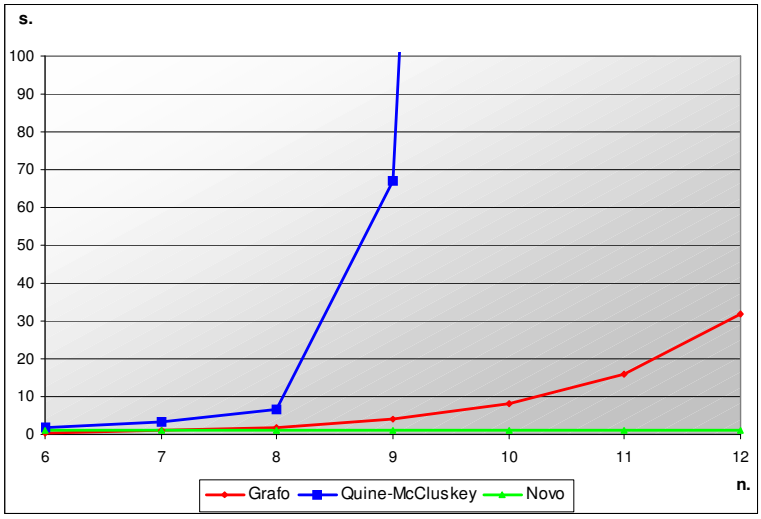

Figura 13 - Tempo gasto na construção de cada representação.

O novo algoritmo para encontrar a FND é descrito a seguir e o seu código encontra-se no Anexo G.

Implementation

```
Begin
abrir arquivo com matpreced; {matpreced é a matriz de precedência}
For i=1 to r do {r é o número de colunas na matriz de precedência}
      se existe alguma restrição na coluna[i] da matriz de precedência então
            c[i]:=’0’ caso contrário c[i]:=’-‘; {c é um vetor de dimensão n}
acrescentar vetor c na matriz de conjunções;
i:=1;
Repeat
For j:=i+1 to r do
      Begin
      novarestrição := coluna[i] + coluna[j] {novarestrição é a soma de duas colunas de matpreced};
      se não existe alguma coluna em matpreced igual a novarestrição então
            Begin
            inc(r);
            acrescentar novarestrição em matpreced como coluna[r];
            end;
      end;
inc(i);
until i = r;
For i:=1 to r do
      se existe alguma restrição na coluna[i] de matpreced então
            Begin
            novaconjunção := c;
            For j:=1 to n do {n é o número de operações em um peça}
                  se matrizpreced[j,i]=1 então novaconjunção[j] := ‘1’;
            For j:=1 to n do Se novaconjunção[j]=0 então
                  Begin
                  aux:=false; {variável auxiliar}
                  For k:=1 to n do
                        se (novaconjunção[k]<>’1’) e (matpreced[k,j]=true) então aux:=true;
                  se (not aux) então novaconjunção[j]:=’-‘;
                  end;
            end;
mostrar FND;
End.
```

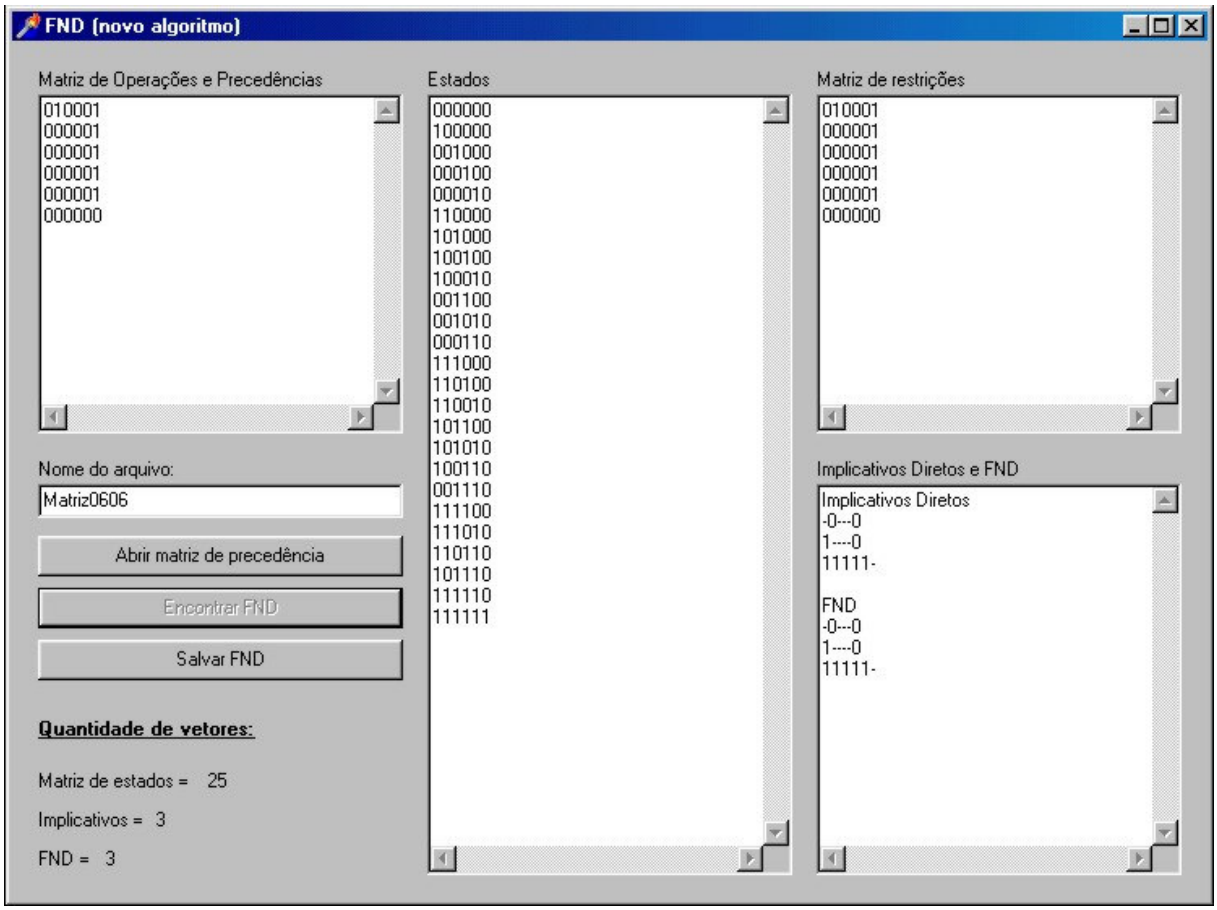

Figura 14 - Interface do Módulo FND (Novo Algoritmo).

5.4. VALIDAÇÃO DOS ALGORITMOS IMPLEMENTADOS

A validação dos algoritmos implementados ocorreu de três maneiras. A primeira forma de validação ocorreu através do acompanhamento de cada variável e procedimento em cada algoritmo. Este acompanhamento foi possível graças ao métodos de controle embutidos no aplicativo DELPHI da Borland. Inicialmente foi utilizado o método *step over*. Este método permite que os programas sejam executados e interrompidos a cada procedimento (ex. um botão, uma função, uma *procedure*, etc.), possibilitando constatar o funcionamento correto de cada procedimento dentro dos algoritmos. Outro método embutido no *software* é chamado de *trace into*. Este método permite averiguar e acompanhar o funcionamento de um algoritmo a cada linha de seu código. Deste modo, é possível acompanhar detalhadamente cada variável dentro do programa em tempo de execução. Este dois métodos foram utilizados na avaliação e acompanhamento dos algoritmos durante toda a sua execução, permitindo alterar o código em tempo de projeto.

A segunda forma de validação ocorreu através das saídas de cada algoritmo. O objetivo desta validação é verificar se os algoritmos implementados executavam corretamente as funções para que foram programados. Além disso, por trabalharem com representações distintas do mesmo espaço de estado, foi possível comparar empiricamente as respostas encontradas pelos algoritmos. Por exemplo, os algoritmos desenvolvidos para encontrar a FND (Quine-McCluskey e Novo Algoritmo) deveriam apresentar a mesma saída. Analogamente, foi possível comparar os resultados entre os estados de um grafo e sua respectiva FND, onde cada estado do grafo deveria ser representado pelo menos por uma conjunção, e, consequentemente, todos os estados representados pelas conjunções deveriam estar no grafo. Esta forma de verificação através das saídas dos algoritmos confirmou o bom funcionamento dos códigos implementados.

Por fim, a terceira forma de validação foi efetuada através da programação de algoritmos de busca que utilizassem representações distintas para o mesmo espaços de estados. Por se tratar do mesmo espaço de estados, estes algoritmos deveriam apresentar a mesma solução ótima para um seqüenciamento qualquer. Os algoritmo para busca de uma solução ótima não são apresentados nesta dissertação, uma vez que seu objetivo foi meramente testar as representações adotadas. Os algoritmos de busca também foram utilizados para validação do modelo linear apresentado no capítulo 6, visto que, este também objetiva o seqüenciamento ótimo da manufatura.

6. APLICAÇÃO DA REPRESENTAÇÃO NA FND EM PROGRAMAÇÃO LINEAR

O objetivo deste capítulo é mostrar uma possível aplicação da representação na FND, ressaltando a sua eficiência. Para verificar sua eficiência foi desenvolvido um modelo em programação linear que encontra a melhor seqüência de manufatura, de forma a evidenciar a potencialidade da representação (seção 6.1.1). O modelo linear desenvolvido permite a prévia análise das alternativas de produção, possibilitando a escolha da ação mais conveniente e eficiente num ambiente de contingências, antes da aplicação real de qualquer seqüência de manufatura.

Como o modelo desenvolvido trabalha com a busca da melhor seqüência de manufatura, foi necessária a criação de um módulo que registrasse os custos de *setup* de uma operação para outra. Este módulo é apresentado na Figura 15 e seu algoritmo é descrito a seguir.

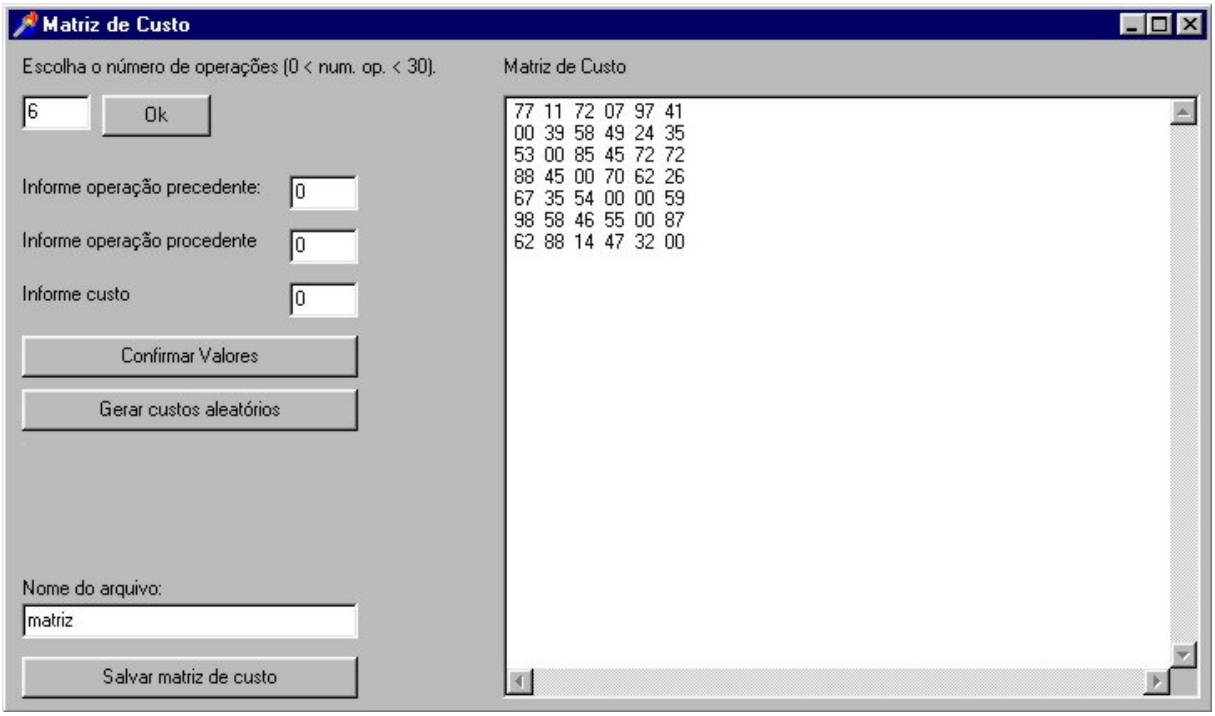

Figura 15 - Interface do Módulo Matriz de Custo.

```
Implementation

    Procedure Botão (determinar custo);
    Begin
    escolha operação precedente;
    escolha operação procedente;
    escolha custo;
    matcustos[operação precedente,operação procedente]:=custo;
    mostrar matriz de custos;
    End;

    Procedure Botão (gerar custos aleatórios);
    Begin
    For i:=0 to n do {n é número de operações}
        For j:=1 to n do matcustos[i,j]:=random;
    mostrar matriz de custos;
    End;

Begin
escolha o número de operações na peça;
End.
```

Na matriz de custo C, C_{ij} indica o custo da operação *i* para a operação *j*. Seguindo esta definição, a matriz C sempre terá uma linha a mais do que o número de colunas, pois existe um estado inicial onde nenhuma operação foi realizada. Arbitrariamente, os custos da operação inicial para a operação *j* são registrados na primeira linha da matriz (*i*=0). Além disso, foi acrescida ao módulo a função *gerar custos aleatórios* com o objetivo de simplificar e acelerar o processo de digitação dos custos. O módulo da matriz de custos também está vinculado ao módulo principal.

6.1. TRANSFORMAÇÃO DA REPRESENTAÇÃO NA FND EM RESTRIÇÕES LINEARES

Nesta seção a representação na FND é transformada em restrições, para resolver o problema de seqüenciamento com flexibilidade através de modelos de programação linear. A representação na FND permite que estes modelos sejam elaborados com uma quantidade menor de restrições e variáveis, comparados com os modelos desenvolvidos por outros autores, conforme observado na seção 2.3.

As restrições desenvolvidas utilizam variáveis inteiras (programação inteira), com sua modelagem realizada e testada no *software* LINDO. O LINDO foi escolhido pela aceitação e confiabilidade verificada em diversos artigos de pesquisa operacional. Cada uma das restrições e variáveis do modelo são explicadas e exemplificadas para melhor ilustrar suas funções. Estes exemplos também utilizam o padrão de modelagem do *software* LINDO. A forma canônica do modelo linear desenvolvido é apresentada a seguir.

Notação:

n : número de operações em uma peça.

K : indica o estado.

j : indica a operação.

id : indica a quantidade de conjunções (ou implicativos diretos) na FND.

X_{kj} : indica se a operação *j* é ou estava efetuada no estado *k*.

A_{ki} : indica se a conjunção *i* retorna valor verdadeiro para o estado k.

$$\sum_{j=1}^{n} X_{kj} = k \qquad \text{para} \quad k = 0...n \tag{1}$$

$$X_{kj} = \begin{cases} 1, \text{ se a } j\text{-ésima operação já estava ou é efetuada no estado } k \\ 0, \text{ no caso contrário} \end{cases}$$

$$X_{kj} \geq X_{k-1j} \qquad \text{para} \quad k = 1...n \quad \text{e} \quad j = 1...n \tag{2}$$

$$\sum_{j=1}^{n} Z_j \geq A_{ki} n \quad \text{para} \quad i = 1...id \quad \text{e} \quad k = 0...n \tag{3}$$

$$Z_j = \begin{cases} 1, \text{se não existe referência da } j\text{-ésima variável booleana na conjunção } i \\ X_{kj}, \text{ se a } j\text{-ésima variável booleana na conjunção } i \text{ for verdadeira} \\ 1 - X_{kj}, \text{ se a } j\text{-ésima variável booleana na conjunção } i \text{ for falsa} \end{cases}$$

$$\sum_{i=1}^{id} A_{ki} \geq 1 \qquad \text{para} \quad k = 0...n \tag{4}$$

Para compreender a primeira restrição (1), do modelo linear, é preciso relembrar a estrutura adotada para uma seqüência de manufatura. Nas seções 5.1 e 5.2, uma seqüência de manufatura foi definida como um conjunto de *n+1* (número de operações mais o estado inicial) estados, onde cada estado subseqüente deveria conter as operações realizadas no estado anterior, acrescido de uma operação. Deste modo, o número de operações já efetuadas em cada estado de manufatura será sempre igual a soma de suas variáveis. O Quadro 10 mostra a estrutura de uma seqüência de manufatura.

Estado da manufatura	Seqüência de manufatura	Qtd de operações já efetuadas
Primeiro (inicial)	000000	0
Segundo	100000	1
Terceiro	110000	2
Quarto	110100	3
Quinto	110110	4
Sexto	111110	5
Sétimo (final)	111111	6
Qtd de estados (n+1) = 7		

Quadro 10 - Seqüência de Manufatura.

Com a estrutura definida para uma seqüência de manufatura, pode-se compreender a restrição (1) do modelo linear. Inicialmente, a restrição define cada estado como o somatório de suas variáveis. Este somatório deve respeitar a quantidade de operações efetuadas, conforme estabelecido pela estrutura adotada. Desta forma, o somatório das variáveis que integram o primeiro estado deve ser zero, do segundo estado deve ser um, e assim, sucessivamente, até completar a quantidade de estados necessários em uma seqüência.

Podemos definir cada variável que integra um estado como X_{kj}, onde k indica o estado e j indica a operação. Por se tratar de uma programação inteira, as variáveis X_{kj} podem assumir apenas os valores um e zero. Sendo assim, X_{kj} assume o valor um quando a j-ésima operação estava ou é efetuada no estado k, e zero no caso contrário. A seguir é possível observar as restrições, modeladas no *software* LINDO, de uma peça que contenha seis operações a serem efetuadas.

X01 + X02 + X03 + X04 + X05 + X06 = 0
X11 + X12 + X13 + X14 + X15 + X16 = 1
X21 + X22 + X23 + X24 + X25 + X26 = 2
X31 + X32 + X33 + X34 + X35 + X36 = 3
X41 + X42 + X43 + X44 + X45 + X46 = 4
X51 + X52 + X53 + X54 + X55 + X56 = 5
X61 + X62 + X63 + X64 + X65 + X66 = 6

De acordo com o exemplo da seqüência de manufatura do Quadro 10, temos: X01, X02, X03, X04, X05, X06, X12, X13, X14, X15, X16, X23, X24, X25, X26, X33, X35, X36, X43, X46 e X56 iguais a zero; e X11, X21, X22, X31, X32, X34, X41, X42, X44, X45, X51, X52, X53, X54, X55, X61, X62, X63, X64, X65 e X66 iguais a um.

Como descrito anteriormente, cada estado subseqüente deve conter as operações realizadas no estado anterior, mais uma operação. Assim, a restrição (2) tem a função de manter organizada a estrutura da seqüência de manufatura, garantindo que cada variável do

estado subseqüente seja maior ou igual, a variável do estado anterior. O Quadro 11 ilustra duas seqüências de manufatura, uma correta outra não. Em seguida, são listados alguns exemplos para a restrição (2).

Seqüência de Manufatura	
Estrutura Correta	**Estrutura Incorreta**
000000	000000
100000	100000
110000	110000
110100	100101
110110	110110
111110	011111
111111	111111

Quadro 11 - Exemplo de uma seqüência incorreta.

$$
\begin{aligned}
&X11 >= X01\\
&X12 >= X02\\
&X13 >= X03\\
&\vdots \qquad\qquad \vdots\\
&X64 >= X54\\
&X65 >= X55\\
&X66 >= X56
\end{aligned}
$$

Uma vez que as restrições para formação dos estados estão definidas, é preciso criar restrições (3 e 4) que indiquem quais estados são válidos. Estas restrições são obtidas e definem a FND dentro do modelo linear. Assim, para que um estado seja considerado válido, as restrições 3 e 4 precisam ser satisfeitas.

Como revisado na seção 4.4, uma FND é formada por uma disjunção de conjunções. Sendo assim, cada conjunção de uma FND precisa ser transformada em uma restrição linear. A FND assumirá o valor verdadeiro se, e somente se, pelo menos uma destas restrições for satisfeita, validando o estado de manufatura.

Na seção 5.2 foi mostrado como uma conjunção pode assumir a forma vetorial. Utilizaremos esta representação para melhor compreender a nova restrição. Sabe-se que, para que um estado seja considerado válido é preciso que, após aplicado a uma FND, pelo menos em uma de suas conjunções, todas as literais retornem valores verdadeiros. Como uma conjunção na forma vetorial é composta por *n* literais, então, por analogia, as *n* variáveis da restrição precisam retornar valores verdadeiros (iguais a um). Deste modo, pode-se afirmar que o somatório das variáveis da restrição, que define uma conjunção, obrigatoriamente, necessita ser igual ou maior que *n*. Para se obter o somatório, é preciso transformar a forma vetorial da conjunção, em uma equação da seguinte maneira:

- para cada posição no vetor, com valor verdadeiro (um), deve-se atribuir a variável X_{kj} equivalente, onde *j* indica a posição;

- para cada posição no vetor, com valor falso (zero), deve-se atribuir 1-X_{kj}, onde *j* indica a posição;

- para cada posição no vetor, onde não existir referência de alguma literal, deve-se atribuir o valor um. O Quadro 12 auxilia na compreensão desta transformação.

Conjunção usando Literais	**Conjunção na formal vetorial**	**Conjunção usando variáveis X_{kj}**
¬B¬F	-0---0	$1 + (1-X_{k2}) + 1 + 1+ 1 + (1-X_{k6})$
A¬F	1----0	$X_{k1} + 1 + 1+ 1 + 1 + (1-X_{k6})$
ABCDE	11111-	$X_{k1} + X_{k2} + X_{k3} + X_{k4} + X_{k5} + 1$

Quadro 12 - Modos de representar conjunções.

Por exemplo, o estado [100000] aplicado as três conjunções do Quadro 12 retornará valor verdadeiro para as conjunções ¬B¬F e A¬F (A=1, B=0, ¬B=1, F=0 e ¬F=1). Do mesmo modo, utilizando o somatório das variáveis X_{kj}, e de uma constante (soma dos valores iguais a um), o estado será válido se, e somente se, o somatório for igual ou maior que *n* (Quadro 13). Para que o somatório das variáveis X_{kj} seja igual a *n*, é preciso que cada variável retorne um, o que equivale dizer que cada literal retornou um valor verdadeiro.

Conjunção	**Somatório variáveis X_{kj}**	**Somatório >= *n***
-0---0	1 + (1-X12) + 1 + 1+ 1 + (1-X16)	1 + 1 + 1 + 1+ 1 + 1 >= 6
1----0	X11 + 1 + 1+ 1 + 1 + (1-X16)	1 + 1 + 1 + 1+ 1 + 1 >= 6
11111-	X11 + X12 + X13 + X14 + X15 + 1	1 + 0 + 0 + 0 + 0 + 1 >= 6
Estado = [100000] X11=1, X12=0, X12=0, X13=0, X14=0, X15=0 e X16=0		

Quadro 13 - Exemplos de restrições satisfeitas, usando variáveis X_{kj}.

Como, raramente, um estado de manufatura irá satisfazer todas as conjunções, então, freqüentemente, o somatório das variáveis X_{kj} será menor do que *n*. É óbvio que para um modelo linear funcionar é preciso que todas as suas restrições sejam válidas. Por esta razão, na restrição (3), a constante *n* é multiplicada por uma variável binária (A_{ki}). Quando esta variável assumir o valor zero, *n* será anulado (*n*.0=0), tornando qualquer estado válido para aquele somatório. Por outro lado, se a variável A_{ki} assumir valores iguais a zero em todas as restrições, então estas restrições não terão utilidade, pois não ocorrerá a verificação e validação do estado. Para contornar este problema, uma nova restrição torna-se necessária. Esta nova restrição (4), soma todas as variáveis A_{ki}, e exige deste somatório

um valor maior ou igual a um. Deste modo, pelo menos, uma conjunção retornará um valor verdadeiro, atribuindo, concomitantemente, um valor verdadeiro para a FND.

Restrição no Modelo Linear	Referente à
- X12 - X16 + 6 >= 6 A11	Primeira conjunção
X11 - X16 + 5 >= 6 A12	Segunda conjunção
X11 + X12 + X13 + X14 + X15 + 1 >= 6 A13	Terceira conjunção
A11 + A12 + A13 >= 1	FND

Quadro 14 - Exemplo de restrições para a FND.

Com as restrições definidas é possível gerar todos os estados de uma seqüência de manufatura. O número de variáveis e restrições utilizadas na modelagem pode ser obtido pelas equações do Quadro 15.

Quantidade de Variáveis Utilizadas	
Quantidade	Definição
n(n-1)	Para as variáveis X_{kj}
id(n-1)	Para as variáveis A_{ki}
n^2 - n + id(n-1)	Total

Quantidade de Restrições Utilizadas	
Quantidade	Definição
n-1	Para restrição (1)
n(n-2)	Para restrição (2)
id(n-1)	Para restrição (3)
n-1	Para restrição (4)
n^2 + id(n-1) - 2	Total

Quadro 15 - Quantidade de variáveis e restrições.

Apesar das restrições apresentadas no início desta seção levarem em consideração *n+1* estados de manufatura, nas equações do Quadro 15 o número de estados foi considerado como *n-1*. A razão da subtração de dois estados deve-se aos valores que os estados, inicial e final, assumem independentemente da quantidade de operações ou seqüência escolhida. Os estados inicial e final sempre assumem um vetor onde todos os termos são zeros e uns, respectivamente (nenhuma e todas operações efetuadas). Assim estes estados são sempre constantes e, portanto não necessitam de variáveis e restrições para sua definição. Todavia, os dois estados foram mantidos na forma canônica do problema para melhor compreensão da programação linear.

Dando continuidade a esta seção, julgou-se apropriado a construção de um modelo linear, que encontre a melhor seqüência de manufatura, tendo como base as restrições desenvolvidas para a representação na FND. Além de evidenciar as vantagens da representação na FND, o modelo pode ser utilizado como interface do *software* LINDO. Esta

interface (Figura 16) está integrada ao módulo principal e utiliza as saídas dos módulos FND e Matriz de Custos, na construção do modelo linear.

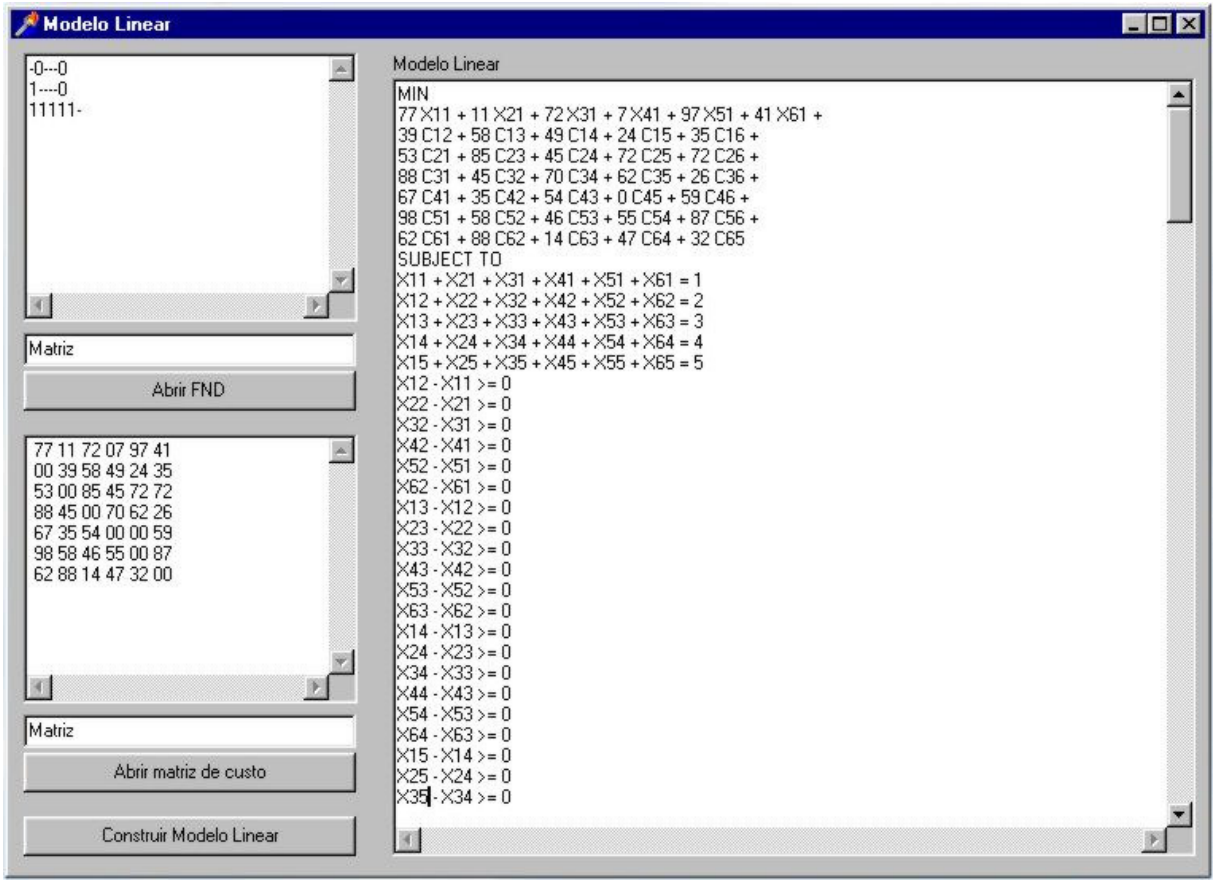

Figura 16 - Interface para o *software* LINDO.

6.1.1. Modelo Linear

Como o objetivo da modelagem é encontrar uma seqüência que minimize o custo de fabricação de uma peça e, considerando que, estes custos estão definidos em uma matriz $C_{n+1,\,n}$, temos como função objetivo.

$$MIN \sum_{i=0}^{n} \sum_{j=1}^{n} H_{ij} C_{ij}$$

Onde:

C_{ij} : é o custo da operação *i* para *j*.

H_{ij} : é a variável de decisão.

Uma vez definida a função objetivo, deve-se gerar as restrições que identifiquem os valores para variáveis de decisão (H_{ij}). Estas variáveis de decisão são binárias e assumem o valor um quando torna-se necessário contabilizar um custo, que por sua vez é determinado

através dos estados de uma seqüência de manufatura. Como vimos anteriormente, estes estados correspondem as variáveis X_{ij} da seção anterior. Sendo assim, para identificar qual operação foi realizada em cada estágio de manufatura, é preciso subtrair cada estado pelo seu antecessor, obtendo uma nova matriz.

Estados de Manufatura	Estado de manufatura menos estado anterior
000000	-
100000	100000
110000	010000
110100	000100
110110	000010
111110	001000
111111	000001

Quadro 16 - Nova matriz de estados.

Para melhor compreensão do processo, considere M_{kj} as variáveis que definem a nova matriz, onde *j* indica a operação realizada no estado *k*. Conforme o exemplo do Quadro 16, as variáveis M11, M22, M34, M45, M53 e M66 são iguais a um, e as demais são iguais a zero.

As novas variáveis (M_{kj}) auxiliam na definição das variáveis de decisão (H_{ij}). Para isso basta somar cada variável M_{kj} em um estado *k*, com cada variável M_{kj} do estado *k+1* (estado sucessor). Quando o resultado desta soma for igual a dois, então saberemos qual custo deve ser contabilizado. Repare que a soma será igual a dois se, e somente se, as duas variáveis M_{kj} apresentarem valores iguais a um, indicando a transição ocorrida entre dois estado. Com esta informação, é possível criar um conjunto de restrições que encontrem, corretamente, as variáveis de decisão. O exemplo abaixo e ilustra este procedimento.

M53 + M66 -1 <= H36

Nota-se, que os valores de *j* nas duas variáveis M_{kj}, são utilizados para identificar a variável H_{ij}. De acordo com o exemplo, o *j* da primeira variável M_{kj} tem valor 3, enquanto o da segunda variável possui valor 6. Isso significa, caso a soma das variáveis M_{kj} seja igual a dois, que a operação 6 foi realizada logo após a operação 3, devendo-se contabilizar este custo (H36=1).

Como as variáveis H_{ij} só podem assumir valores iguais a um e zero, então a restrição apresentada jamais será satisfeita, quando a soma das variáveis M_{kj} for igual a dois. Por esta razão, conforme pode ser observado no exemplo, a soma das variáveis M_{kj} é subtraída de uma unidade. Além disso, sem essa subtração, as variáveis de decisão seriam

contabilizadas para valores em que, a soma das variáveis M_{kj}, também assumissem resultados iguais a um, e como vimos anteriormente, os custos devem ser registrados apenas para resultados iguais a dois. Deste modo, a subtração de uma unidade viabiliza a restrição para quaisquer valores assumidos pelas variáveis M_{kj}.

O número de restrições e variáveis adicionadas ao modelo, é mostrado no Quadro 17. Como as variáveis M_{kj} são obtidas das subtrações de $X_{k,j}$ e $X_{k-1,j}$, então, para evitar o acréscimo de uma nova variável no modelo, atribuiu-se à restrição a seguinte forma canônica:

$$(X_{k,i} - X_{k-1,i}) + (X_{k+1,j} - X_{k,j}) - 1 \leq H_{ij} \quad \text{para} \quad k = 1...n \quad \text{e} \quad i, j = 1...n \quad \text{sendo} \quad i \neq j$$

Quantidade	Definição
$n(n+1) = n^2 + n$	Para as variáveis de decisão (H_{ij})
$(n-1)(n-1)(n) = n^3 - n^2 + n$	Para as restrições adicionais

Quadro 17 - Quantidade de restrições e variáveis do primeiro modelo linear.

O modelo linear desenvolvido minimiza os custos de uma seqüência de produção em uma manufatura flexível. Cabe ressaltar que outras programações lineares poderiam ser efetuadas utilizando a mesma representação na FND, permitindo, deste modo, o uso de diferentes restrições na identificação dos custos.

No modelo desenvolvido nesta seção, o acréscimo no número de restrições e variáveis não apresenta um comportamento exponencial, conforme pode ser verificado pelas equações nos Quadros 15 e 17. O exame de trabalhos anteriores (seção 2.3) mostra que o comportamento exponencial, encontrado em outros modelos lineares, acarreta em um severo esforço computacional, que impede a aplicação dos modelos na resolução de problemas relativamente grandes. Assim, ressalta-se a eficiência da representação na FND no tratamento da flexibilidade em programação linear. Todavia, não significa afirmar que a representação na FND seja a única a apresentar a vantagem de reduzir a quantidade de restrições e variáveis em modelos lineares, por outro lado, ela cumpre o objetivo deste capítulo quando ressalta sua eficiência e aplicabilidade.

Para finalizar a seção, julgou-se oportuno apresentar a melhor seqüência de manufatura, para peça motriz da Parker (Tabela 5). O resultado encontrado considera a matriz de custos da Tabela 4, e a programação linear utilizada pode ser visualizada no anexo H.

Tabela 4 - Matriz de custos.

Operação precedente	**Operação procedente**					
	1	2	3	4	5	6
0	77	11	72	7	97	41
1	0	39	58	49	24	35
2	53	0	85	45	72	72
3	88	45	0	70	62	26
4	67	35	54	0	0	59
5	98	58	46	55	0	87
6	62	88	14	47	32	0

Tabela 5 - Melhor seqüência de manufatura.

Estado	**Custo**	**Operação**	**Descrição**
000000	0	-	-
100000	77	1	Corte da engrenagem
110000	39	2	Aparar dentes
110100	45	4	Serrilhar f1
110110	0	5	Serrilhar superfícies A e B
111110	46	3	Serrilhar f2
111111	26	6	Tratamento de calor
Total	**233**		

7. CONCLUSÕES E RECOMENDAÇÕES

Esta pesquisa apresentou o desenvolvimento de uma representação por espaço de estados capaz de reproduzir, se necessário, todas as possíveis seqüências de uma manufatura flexível. Como citado durante a pesquisa, este é um problema de complexidade NP, exigindo excessivo esforço computacional. Deste modo, a representação desenvolvida deveria apresentar características que permitissem a tratabilidade deste tipo de problema, ou seja, necessidade de pouca memória computacional e a capacidade de construção do espaço de estados em tempo real.

Inicialmente, trabalhou-se com representações na forma matricial. Todavia esta representação não era hábil para enumerar todas as seqüências de manufatura, incapacitando a pesquisa de alternativas durante a produção. Para contornar esta limitação foi desenvolvida um representação por espaço de estados na forma de grafo. Estes grafo mostrava todas as seqüências de manufatura, onde seus nós indicavam os estados do sistema e seus arcos a transição entre estes estados. Contudo, o esforço computacional requerido nesta representação é por demais elevado, em relação as expectativas e necessidades dos sistemas de manufatura, exigindo o desenvolvimento de uma terceira, e mais eficiente, representação.

A modelagem de estados por vetores booleanos, associado a necessidade de uma representação mais enxuta, levou a transformação do espaço de estados em uma FND. Esta escolha fez com que o espaço de estados pudesse ser representado por algumas poucas conjunções fundamentais, tornando a memória requerida inexpressiva, em termos computacionais. A baixa necessidade de memória solucionou o problema da complexidade de espaço, todavia o tempo para encontrar a FND continuava excessivo, devido ao comportamento exponencial para cada operação adicionada ao sistema.

Como as características de uma manufatura flexível exigem uma resposta do sistema em tempo real, foi necessário desenvolver um novo algoritmo mais eficiente do que o de Quine-McCluskey, utilizado inicialmente para encontrar a FND. O novo algoritmo desenvolvido permitiu encontrar a FND em um tempo consideravelmente menor, apesar de também apresentar complexidade exponencial. A eficiência do novo algoritmo deve-se a

busca, das conjunções fundamentais, diretamente pela matriz de precedência, sem com isso, precisar construir o grafo de seqüências para enumerar todos os estados.

Cabe ressaltar que a maioria das representações não consegue contornar, simultaneamente, a complexidade de tempo e de memória. O novo algoritmo desenvolvido para encontrar a FND, diretamente da matriz de precedência, permitiu que uma única representação utilizasse pouca memória computacional e gerasse o espaço de estados completo de uma manufatura em tempo real.

Além das vantagens computacionais proporcionadas pela representação na FND, foi possível perceber as vantagens de sua aplicação em modelos lineares. Os modelo lineares, desenvolvidos até então, utilizavam uma grande quantidade de variáveis e restrições para poder tratar o problema de flexibilidade. Geralmente, nestes modelos cada variável ou restrição representava uma seqüência de produção, assim o aumento no número de inequações inviabilizava a programação linear. Deste modo, para demonstrar uma das possíveis aplicações da representação na FND, no final da pesquisa foi desenvolvido um modelo de programação linear que encontra uma seqüência de manufatura com o menor custo. É possível perceber, que o modelo linear desenvolvido não tem um aumento exponencial no número de variáveis e restrições, conforme aumenta-se o número de operações. Isso implica na capacidade de tratar a flexibilidade com programação linear de forma a representar explicitamente as questões de seqüenciamento.

Finalizando, acredita-se que a simplicidade da representação gerada nesta pesquisa permitirá que um computador central, responsável pelo planejamento da manufatura, prediga em menos tempo o comportamento do sistema sob diferentes configurações. A vantagem desta representação é a possibilidade de sua aplicação antes (*off-line*) ou durante (*on-line*) a produção de uma peça. Consequentemente, a representação poderá auxiliar na administração da produção sob diferentes condições de operação, permitindo a aplicação de diversos critérios na seleção da melhor alternativa, de acordo com os parâmetros adotados em cada manufatura (custo, tempo de transporte, tempo de processamento, prioridade de produção, etc.). Deste modo, espera-se que esta representação sirva para modelos de mensuração da performance de uma manufatura flexível e auxilie no uso mais eficiente dos recursos.

Estes benefícios são conseqüências de um modelo abordado de maneira simples e compacta, no qual fornece uma representação clara por espaço de estados da flexibilidade em manufatura e, ao mesmo tempo, permite uma eficiente implementação para análise computacional. Esta representação ainda poderá ser facilmente integrada com outros

modelos analíticos e de simulação direcionados para a análises e avaliações mais complexas de sistemas flexíveis de manufatura. Estes resultados indicam claramente a superioridade da representação desenvolvida sobre outras representações encontradas na literatura.

Para aperfeiçoamento e continuidade desta pesquisa, sugere-se que a representação desenvolvida também aborde a flexibilidade de máquinas. Estas duas flexibilidades são consideradas as mais importantes dentro de um sistema de manufatura, uma vez que englobam e explicam as demais (BROWNE, 1984). Dando continuidade a pesquisa, a representação na FND poderia ser utilizada na construção de simuladores ou ferramentas de decisão que auxiliassem na mensuração dos sistemas flexíveis de manufatura.

Apesar desta pesquisa focar a flexibilidade de manufatura, acredita-se que o modelo nela desenvolvido possa ser aplicado em diversas áreas do conhecimento. Assim, outros problemas, que também trabalhem com alguma forma de seqüenciamento, poderiam ser solucionados com o uso da representação desenvolvida. Do mesmo modo, o uso da programação linear na resolução destes problemas traria os benefícios de um modelo economicamente viável e acessível.

Referências Bibliográficas

ACKOFF, R. L. e SASIENI, M. W. **Pesquisa Operacional**. Rio de Janeiro: Livros Técnicos e Científicos, 1977.

ALBUQUERQUE, F. **Programando em Linguagem C, C++ e turbo C++**. Rio de Janeiro: Berkeley, 1991.

ANDRADE, E. L. de. **Introdução a Pesquisa Operacional – Métodos e modelos para a análise de decisão**. Rio de Janeiro: LTC, 1998.

BARAD M. e SIPPER, D. Flexibility in manufacturing sytems: definitions and Petri net modeling. **International Journal of Production Research**, vol 26, no 2, 1988, 237-248 p.

BARAD, M. e SIPPER, D. Flexibility and types of changes in FMSs: a timed petri-nets assessment of machine flexibility. **The International Journal of Advanced Manufacturing Technology**, no 5, 1990, 292-306 p.

BARAD, M. Impact of some flexibility factors in FMS – a performance evaluation approach. **International Journal of Production Research**, vol 30, no 11, 1992, 2587-2602 p.

BERNARDO, J. J. e MOHAMED, Z. The measurement and the use of operational flexibility in the loading of flexible manufacturing systems. **European Journal of Operational Research**, 1992, 144-155 p.

BERTALANFFY, L. V. **Teoria Geral dos Sistemas**. Editora: Vozes, Rio de Janeiro, 1975.

BORENSTEIN, D. **Modelo de Simulação Computacional de Sistemas Flexíveis de Manufatura**. Dissertação de Mestrado, PPGA, UFRGS, 1991.

BORENSTEIN, D. A Directed Acyclic Graph Representation of Routing Manufacturing Flexibility. **European Journal of Operational Research**, 1999, 1-17 p.

BOYER, K. Evolutionary patterns of flexibility automation and performance: a longitudinal study. **Management Science**, vol 45, no 6, 1999, 824-842 p.

BRILL, P. H. e MANDELBAUM, M. On measures of flexibility in manufacturing systems. **International Journal of Production Research**, vol 27, no 5, 1989, 747-756 p.

BROWNE, J. et al. Classification of Flexible Manufacturing Systems. **The FMS Magazine**, no 2, 1984, 114-117 p.

CAMPELLO, R. E. e MACULAN, N. **Algoritmos e Heurísticas - Desenvolvimento e Avaliação de Performance**. Niterói: EDUFF, 1994.

CHEN, I. J. e CHUNG, C. H. An examination of flexibility measurements and performance of flexibility manufacturing systems. **International Journal of Production Research**, vol 34, no 2, 1996, 379-394 p.

DAST, S. K. e GENDRA, P. N. Investigations into the impact of flexibility on manufacturing performance. **International Journal of Production Research**, vol 31, no 10, 1993, 2337-2354 p.

EVANS, G. W. e HADDOCK, J. Modeling tools for flexible manufacturing systems. **Production Planning and Control**, vol 3, no 2, 1992, 158-167 p.

FENSTERSEIFER, J. E. **Um modelo conceitual para avaliação de investimentos em tecnologias flexíveis de produção**. Documentos para Estudo, PPGA, UFRGS, no 12, 1991

FLOOD, R. e CARSON, E. **Dealing with Complexity**. New York: Plenum Press, 1993.

GERWIN, D. An agenda for research on the flexibility of manufacturing processes. **IJOPM**, 1986, 38-49 p.

GOLDBARG, M. C. e LUNA, H. P. L. **Otimização Combinatória e Programação Linear - Modelos e Algoritmos**. Rio de Janeiro: Campus, 2000.

GUPTA, Y. P. e GOYAL, S. Flexibility of manufacturing systems: concepts and measurements. **European Journal of Operational Research**, no 43, 1989, 119-135 p.

GUPTA, Y. P. e GOYAL, S. Flexibility trade-offs in a random flexible manufacturing system: a simulation study. **International Journal of Production Research**, vol 30, no 3, 1992, 527-557 p.

HANCOCK, T. M. Effects of alternate routeings under variable lot size conditions. **International Journal of Production Research**, vol 27, 1989, 247-259 p.

HWANG, C., GUO, T.Y. e SHIEH L. S. A canonical state-space representation for systems using Multipoint. **Journal of the Franklin Institute**, vol 328, no 2/3, 1991, 207-216 p.

ISIDORI A. e RUBERTI, A. State-space representation and realization of time-varying linear input-output functions. **Journal of the Franklin Institute**, vol 301, no 6, 1976, 573-592 p.

KIM, K. H. e EGBELU, P. J. Scheduling in a production environment with multiple process plans per job. **International Journal of Production Research**, vol 37, no 12, 1999, 2725-2753 p.

KOCHIKAR, V. P. e NARENDRAN, T. T. A framework for assessing the flexibility of manufacturing systems. **International Journal of Production Research**, vol 30, no 12, 1992, 2873-2895 p.

KOCHIKAR, V. P. e NARENDRAN, T. T. State Space approach to qualitative analysis of FMSs. **Computer Integrated Manufacturing Systems**, vol 6, 1993.

KUSIAK, A. The generalized group technology concept. **International Journal of Production Research**, vol 25, no 4, 1987, 561-569 p.

KUSIAK, A. e CHO, M. Similarity coefficient algorithms for solving the group technology problem. **International Journal of Production Research**, vol 30, no 11, 1992, 2633-2646 p.

LIN, Y. e SOLBERG, J. J. Effectiveness of flexible routing control. **The International Journal of Flexible Manufacturing Systems**, vol 3, 1991, 189-211 p.

MELLO, L. S. H. e SANDERSON, A. C. Representation of Mechanical Assembly Sequences. **IEEE Transactions on Robotics and Automation**, vol 7, no 2, 1991, 211-227 p.

MENDELSON, E. **Álgebra Booleana e Circuitos de Chaveamento**. McGraw-Hill: São Paulo, 1977, 283 p.

MORGAN, E. T. e RAZOUK, R. R. Interactive State-Space Analysis of Concurrent Systems. **IEEE Transactions on Software Engineering**, vol 13, no 10, 1987, 1080-1091 p.

MORIKAWA e FUJISAK. System Identification of the Speech Production Process Based on a State-Space Representation. **IEEE Transactions on Acoustic Speech**, vol 32, no 2, 1984, 252-262 p.

NEVIS, A. H. e MAJUNDAR, A. K. State space representation of nerve excitation. **International Journal of Quantum Chemistry**, vol 4, 1977, 155-160 p.

PIDD, Michael. **Modelagem Empresarial – ferramentas para tomada de decisão**. Porto Alegre: Bookman, 1998.

REISDORPH, K. **Aprenda em 21 dias Delphi 4**. Rio de Janeiro: Campus, 1999.

REVELIOTIS, S. A. Accomodating FMS Operational Contingencies Through Routing Flexibility. **IEEE Transactions on Robotics and Automation**, vol 15, no 1, 1999, 3-19 p.

ROTHENBERG, J. The Nature of Modeling. Em: L.. E. Wildman, K. A. Loparo e N. R. Nielsen (editores), **Artificial Intelligence, Simulation and Modeling**, Wiley, 1989, 75-92 p.

SARKER, B. R., KRISHNAMURTHY, S. e KUTHETHUR, S. G. A survey and critical review of flexibility measures in manufacturing systems. **Production Planning and Control**, vol 5, no 6, 1994, 512-523 p.

SHAFER, S. M. e CHARNES, J. M. Offsetting lower routeing flexibility in celular manufacturing due to machine dedication. **International Journal of Production Research**, vol 35, no 2, 1997, 551-567 p.

SHAMBLIN, J. E. e STEVENS JR., G. T. **Pesquisa Operacional – uma abordagem básica**. São Paulo: Atlas, 1979.

SIMON, H. A. Prediction and Prescription in Systems Modeling. **OR Forum**, 1989, 7-13 p.

TINCKNELL, D. J. e RADCLIFFE, D. F. A generic model of manufacturing flexibility based on system control hierarchies. **International Journal of Production Research**, vol 34, no 1, 1996, 19-32 p.

TOSCANI, L. V. e VELOSO, P. A. S. **Complexidade de algoritmos**. Porto Alegre: Instituto de Informática da UFRGS: Sagra Luzzatto, 2001.

UPTON, D. M. What Really Makes Factories Flexible?. **Harvard Business Review**, Julho-Agosto, 1995, 74-84 p.

YAO, D. D. e PEI, F. F. Flexible parts routing in manufacturing systems. **IEEE Transactions**, vol 22, 1987, 48-55 p.

Anexo A – Código do Projeto do Módulo Principal

```
program Projmenuprincipal;

uses
  Forms,
  menuprincipal in 'menuprincipal.pas' {FormMenuPrincipal},
  matrizprecedencia in 'matrizprecedencia.pas' {FormMatrizPrecedencia},
  matrizcusto in 'matrizcusto.pas' {FormMatrizCusto},
  buscagrafo in 'buscagrafo.pas' {FormBuscaGrafo},
  grafo in 'grafo.pas' {FormGrafo},
  fnd in 'fnd.pas' {FormFND},
  buscafnd in 'buscafnd.pas' {FormBuscaFND},
  fnddireto in 'fnddireto.pas' {FormFNDDireto},
  LinearModel in 'LinearModel.pas' {FormLinearModel};
  LinearModel2 in 'LinearModel2.pas' {FormLinearModel2};


{$R *.RES}

Begin
  Application.Initialize;
  Application.CreateForm(TFormMenuPrincipal, FormMenuPrincipal);
  Application.CreateForm(TFormMatrizPrecedencia, FormMatrizPrecedencia);
  Application.CreateForm(TFormMatrizCusto, FormMatrizCusto);
  Application.CreateForm(TFormBuscaGrafo, FormBuscaGrafo);
  Application.CreateForm(TFormGrafo, FormGrafo);
  Application.CreateForm(TFormFND, FormFND);
  Application.CreateForm(TFormBuscaFND, FormBuscaFND);
  Application.CreateForm(TFormFNDDireto, FormFNDDireto);
  Application.CreateForm(TFormLinearModel, FormLinearModel);
  Application.CreateForm(TFormLinearModel2, FormLinearModel2);
  Application.Run;
End.
```

Anexo B – Código do Módulo Principal

```
unit menuprincipal;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls;

type
  TFormMenuPrincipal = class(TForm)
    Button1: TButton;
    Button2: TButton;
    Button3: TButton;
    Button4: TButton;
    Button5: TButton;
    Button6: TButton;
    Button7: TButton;
    Button8: TButton;
    Button9: TButton;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure Button5Click(Sender: TObject);
    procedure Button6Click(Sender: TObject);
    procedure Button7Click(Sender: TObject);
    procedure Button8Click(Sender: TObject);
    procedure Button9Click(Sender: TObject);
    procedure Button10Click(Sender: TObject);

  private
    { Private declarations }
  public
    { Public declarations }
  End;

var
  FormMenuPrincipal: TFormMenuPrincipal;

Implementation

uses matrizprecedencia, matrizcusto, buscagrafo, fnd, grafo, buscafnd, fnddireto, LinearModel,
LinearModel2;

{$R *.DFM}

procedure TFormMenuPrincipal.Button1Click(Sender: TObject);
Begin
formmatrizprecedencia.showmodal;
End;

procedure TFormMenuPrincipal.Button2Click(Sender: TObject);
Begin
formmatrizcusto.showmodal;
```

```
End;

procedure TFormMenuPrincipal.Button3Click(Sender: TObject);
Begin
formBuscaFND.showmodal;
End;

procedure TFormMenuPrincipal.Button4Click(Sender: TObject);
Begin
formbuscagrafo.showmodal;
End;

procedure TFormMenuPrincipal.Button5Click(Sender: TObject);
Begin
formfnd.showmodal;
End;

procedure TFormMenuPrincipal.Button6Click(Sender: TObject);
Begin
formgrafo.showmodal;
End;

procedure TFormMenuPrincipal.Button8Click(Sender: TObject);
Begin
formfnddireto.showmodal;
End;

procedure TFormMenuPrincipal.Button9Click(Sender: TObject);
Begin
formLinearModel.showmodal;
End;

procedure TFormMenuPrincipal.Button10Click(Sender: TObject);
Begin
formLinearModel2.showmodal;
End;

End.
```

Anexo C – Código do Módulo Matriz de Precedência

```
unit matrizprecedencia;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls;

type
  TFormMatrizPrecedencia = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label7: TLabel;
    Edit1: TEdit;
    Edit2: TEdit;
    Button1: TButton;
    Button2: TButton;
    Button3: TButton;
    Button4: TButton;
    Memo1: TMemo;
    Edit3: TEdit;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure Button4Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  End;

var
  FormMatrizPrecedencia: TFormMatrizPrecedencia;
  a,k,numprocess: integer;
  matop: array of array of boolean;
  s:string;
  arqmatriz: file of integer;

Implementation

{$R *.DFM}

procedure TFormMatrizPrecedencia.Button3Click(Sender: TObject);
var
i,j:integer;

Begin
numprocess:=strtoint(edit2.text);
  if (numprocess>0) and (numprocess<21) then
    Begin
    {define o tamanhao da matop e atribue false para todos seus valores}
```

```
    memo1.lines.clear;
    label4.caption:='Digite uma operação precedente a operação 1';
    label2.caption:='Nome do arquivo:';
    button1.Enabled:=true;
    button2.Enabled:=true;
    button4.Enabled:=true;
    setlength(matop,numprocess,numprocess);
    k:=0;
    For i:=0 to numprocess-1 do
      Begin
      s:='';
      For j:=0 to numprocess-1 do if matop[i,j] then s:=s+'1' else s:=s+'0';
      memo1.lines.add(s);
      End;
    End;
End;

procedure TFormMatrizPrecedencia.Button1Click(Sender: TObject);
var
x:integer;

   procedure mostrarmatop;
   var
   i,j:integer;
   Begin
   memo1.lines.clear;
   For i:=0 to numprocess-1 do
     Begin
     s:='';
     For j:=0 to numprocess-1 do if matop[i,j] then s:=s+'1' else s:=s+'0';
     memo1.lines.add(s);
     End;
   End;

Begin
x:=strtoint(edit1.text);
if (x>0) and (x<=numprocess) and (x<>k+1) then
  Begin
  matop[x-1,k]:=true;
  label5.caption:='Operação aceita';
  mostrarmatop;
  End
  else label5.caption:='Valor deve ser: 0<x<num.op. e x<>op.';
End;

procedure TFormMatrizPrecedencia.Button2Click(Sender: TObject);
Begin
if k>=numprocess-1 then label5.caption:='Não há mais operações.' else inc(k);
label4.caption:='Digite uma operação precedente a operação '+ inttostr(k+1);
End;

procedure TFormMatrizPrecedencia.Button4Click(Sender: TObject);
var
j,i:integer;

Begin
s:=edit3.text+'.mtr';
AssignFile(arqmatriz,s);
rewrite(arqmatriz);
reset(arqmatriz);
```

```
write(arqmatriz,numprocess);
For i:=0 to numprocess-1 do
   For j:=0 to numprocess-1 do
      if matop[i,j] then
         Begin
         a:=1;
         write(arqmatriz,a);
         End
      else
         Begin
         a:=0;
         write(arqmatriz,a);
         End;
closefile(arqmatriz);
label2.caption:='Matriz salva';
button1.Enabled:=false;
button2.Enabled:=false;
button4.Enabled:=false;
End;

End.
```

Anexo D – Código do Módulo Matriz de Custo

```
unit matrizcusto;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls;

type
  TFormMatrizCusto = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label7: TLabel;
    Label9: TLabel;
    Edit1: TEdit;
    Edit2: TEdit;
    Edit3: TEdit;
    Edit4: TEdit;
    Edit5: TEdit;
    Button1: TButton;
    Button2: TButton;
    Button3: TButton;
    Button4: TButton;
    Button5: TButton;
    Memo1: TMemo;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure Button5Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  End;

var
  FormMatrizCusto: TFormMatrizCusto;
  a,k,numprocess: integer;
  matcust: array of array of integer;
  s:string;
  arqmatriz: file of integer;

Implementation
{$R *.DFM}

procedure TFormMatrizCusto.Button3Click(Sender: TObject);
var
i,j:integer;

Begin
```

```
numprocess:=strtoint(edit2.text);
  if (numprocess>0) and (numprocess<31) then
    Begin
    memo1.lines.clear;
    label2.caption:='Nome do arquivo:';
    button1.Enabled:=true;
    button2.Enabled:=true;
    button4.Enabled:=true;
    button5.Enabled:=true;
    setlength(matcust,numprocess+1,numprocess);
    For i:=0 to numprocess do
      Begin
      s:='';
      For j:=0 to numprocess-1 do s:=s+' 00';
      memo1.lines.add(s);
      End;
    End;
End;

procedure TFormMatrizCusto.Button1Click(Sender: TObject);
var
i,j:integer;

procedure mostrarmatcust;
   var
   i,j:integer;
   Begin
   memo1.lines.clear;
   For i:=0 to numprocess do
     Begin
     s:='';
     For j:=0 to numprocess-1 do
        if matcust[i,j]<10 then s:=s+' 0'+inttostr(matcust[i,j]) else s:=s+' '+inttostr(matcust[i,j]);
     memo1.lines.add(s);
     End;
   End;

Begin
k:=0;
i:=strtoint(edit4.text);
if (i<1) or (i>numprocess) then k:=1;
j:=strtoint(edit5.text);
if (j<1) or (j>numprocess) then k:=1;
a:=strtoint(edit1.text);
if (a<0) or (a>99) then k:=1;
if k=0 then
   Begin
   matcust[i,j-1]:=a;
   label5.caption:='Custo atribuído.';
   label2.caption:='Nome do arquivo:';
   mostrarmatcust;
   End
      else label5.caption:='Custo deve ser menor que 99, ou reinforme as operações.';
End;

procedure TFormMatrizCusto.Button2Click(Sender: TObject);
var
i,j: integer;

procedure mostrarmatcust;
```

```
  var
  i,j:integer;
  Begin
  memo1.lines.clear;
  For i:=0 to numprocess do
    Begin
    s:='';
    For j:=0 to numprocess-1 do
      if matcust[i,j]<10 then s:=s+' 0'+inttostr(matcust[i,j]) else s:=s+' '+inttostr(matcust[i,j]);
    memo1.lines.add(s);
    End;
  End;

Begin
randomize;
For i:=0 to numprocess do
  For j:=0 to numprocess-1 do
    if (i-1=j) then matcust[i,j]:=0 else matcust[i,j]:=random(100);
mostrarmatcust;
label5.caption:='Matriz aleatória gerada.';
label2.caption:='Nome do arquivo:';
End;

procedure TFormMatrizCusto.Button5Click(Sender: TObject);
var
i,j: integer;

procedure mostrarmatcust;
  var
  i,j:integer;
  Begin
  memo1.lines.clear;
  For i:=0 to numprocess do
    Begin
    s:='';
    For j:=0 to numprocess-1 do
      if matcust[i,j]<10 then s:=s+' 0'+inttostr(matcust[i,j]) else s:=s+' '+inttostr(matcust[i,j]);
    memo1.lines.add(s);
    End;
  End;

Begin
For i:=0 to numprocess do
  For j:=0 to numprocess-1 do
    if (j>=i) and (i<>0) then matcust[i,j]:=matcust[j+1,i-1];
mostrarmatcust;
label5.caption:='Matriz simétrica gerada.';
label2.caption:='Nome do arquivo:';
End;

procedure TFormMatrizCusto.Button4Click(Sender: TObject);
var
j,i:integer;

Begin
s:=edit3.text+'.cst';
AssignFile(arqmatriz,s);
rewrite(arqmatriz);
reset(arqmatriz);
write(arqmatriz,numprocess);
```

```
For i:=0 to numprocess do
   For j:=0 to numprocess-1 do
      Begin
      a:=matcust[i,j];
      write(arqmatriz,a);
      End;
closefile(arqmatriz);
label2.caption:='Matriz salva';
End;

End.
```

Anexo E – Código do Módulo Grafo

```
unit grafo;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls;

type
  TFormGrafo = class(TForm)
    Memo1: TMemo;
    Memo2: TMemo;
    Memo3: TMemo;
    Label1: TLabel;
    Label2: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    Button1: TButton;
    Button2: TButton;
    Next: TButton;
    Edit1: TEdit;
    Edit2: TEdit;
    Label3: TLabel;
    Label4: TLabel;
    procedure NextClick(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  End;

var
  FormGrafo: TFormGrafo;
  numprocess,a,ultimono,noatual: integer;
  matop,matest: array of array of boolean;
  nomostrado,noresp: array of boolean;
  s:string;
  arqmatriz: file of integer;
  x: real;

Implementation

{$R *.DFM}

procedure TFormGrafo.Button1Click(Sender: TObject);
var
i,j,y:integer;

Begin
noatual:=-1;
ultimono:=0;
```

```
memo1.lines.clear;
memo2.lines.clear;
memo3.lines.clear;
label6.Caption:='';
button2.Enabled:=false;
next.Enabled:=true;
next.Caption:='Iniciar';
label1.caption:='Nome do arquivo:';
edit2.text:=edit1.text;
s:=edit1.text+'.mtr';
AssignFile(arqmatriz,s);
reset(arqmatriz);
read(arqmatriz,a);
numprocess:=a;
setlength(matop,numprocess,numprocess);
x:=exp(ln(2)*numprocess);
Y:=trunc(x);
setlength(matest,y,numprocess);
setlength(noresp,numprocess);
setlength(nomostrado,numprocess);
For i:=0 to numprocess-1 do
  Begin
  s:='';
  For j:=0 to numprocess-1 do
   Begin
   read(arqmatriz,a);
   if a=1 then
     Begin
     matop[i,j]:=true;
     s:=s+'1'
     End
        else
     Begin
     matop[i,j]:=false;
     s:=s+'0';
     End;
   End;
  memo1.lines.add(s);
  End;
closefile(arqmatriz);
End;

procedure TFormGrafo.NextClick(Sender: TObject);
var
i:integer;

   procedure multmatr;
   var
   i,j:integer;
   Begin
   {calcula novas possibilidades de nós}
   For i:=0 to numprocess-1 do noresp[i]:=false;
   For i:=0 to numprocess-1 do
      For j:=0 to numprocess-1 do
     if (not matest[noatual,j]) and (matop[j,i]) then noresp[i]:=true;
   For i:=0 to numprocess-1 do
     if (noresp[i]) or (matest[noatual,i]) then noresp[i]:=true;
   End;

   procedure mostrarnos;
```

```
  var
  i,j:integer;

    procedure gerarnovono;
    var
    z,x,i,j:integer;
    Begin
    z:=0;
    For i:=noatual to ultimono do
      Begin
      x:=0;
      For j:=0 to numprocess-1 do
        if nomostrado[j]=matest[i,j] then inc(x);
      if x=numprocess then z:=1;
      End;
    if z=0 then
      Begin
      s:='';
      inc(ultimono);
      For i:=0 to numprocess-1 do matest[ultimono,i]:=nomostrado[i];
      For j:=0 to numprocess-1 do if nomostrado[j] then s:=s+'1' else s:=s+'0';
      memo2.lines.add(s);
      End;
    End;

  Begin
  s:='';
  For i:=0 to numprocess-1 do if matest[noatual,i] then s:=s+'1' else s:=s+'0';
  label6.caption:=s;
  memo3.lines.clear;
  For i:=0 to numprocess-1 do
    if noresp[i]=false then
      Begin
      s:='';
      For j:=0 to numprocess-1 do nomostrado[j]:=matest[noatual,j];
      nomostrado[i]:=true;
      For j:=0 to numprocess-1 do if nomostrado[j] then s:=s+'1' else s:=s+'0';
      memo3.lines.add(s);
      gerarnovono;
      End;
  End;

Begin
{verificando se já é o nó 111...111}
button1.Enabled:=false;
next.Caption:='Próximo estado';
s:='';
For i:=0 to numprocess-1 do
  if matest[ultimono,i] then s:=s+'1' else s:=s+'0';
memo2.lines.add(s);
Repeat
  Begin
  inc(noatual);
  multmatr;
  mostrarnos;
  End;
until noatual=ultimono;
label6.caption:='(não há mais nós)';
next.Enabled:=false;
button1.Enabled:=true;
```

```
button2.Enabled:=true;
End;

procedure TFormGrafo.Button2Click(Sender: TObject);
var
j,i:integer;

Begin
s:=edit2.text+'.est';
AssignFile(arqmatriz,s);
rewrite(arqmatriz);
reset(arqmatriz);
write(arqmatriz,numprocess);
write(arqmatriz,ultimono);
For i:=0 to ultimono do
   For j:=0 to numprocess-1 do
      if matest[i,j] then
         Begin
         a:=1;
         write(arqmatriz,a);
         End
      else
         Begin
         a:=0;
         write(arqmatriz,a);
         End;
closefile(arqmatriz);
label1.caption:='Matriz salva';
button2.Enabled:=false;
End;

End.
```

Anexo F – Código do Módulo FND

```
unit fnd;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls;

type
  TFormFND = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Memo2: TMemo;
    Memo3: TMemo;
    Edit1: TEdit;
    Edit2: TEdit;
    Button1: TButton;
    Button4: TButton;
    Button2: TButton;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    Label9: TLabel;
    Label11: TLabel;
    Label12: TLabel;
    procedure Button4Click(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  End;

var
  FormFND: TFormFND;
  p,jaexiste,ultimono,ultimonoquine2,ultimonoquine3,numprocess,g: integer;
  marcado: array of boolean;
  novono: array of char;
  mataux,matquine1,matquine2,matquine3: array of array of char;
  arqmatriz: file of integer;
  s:string;
  tempo: TDateTime;
  hora,min,seg,mseg: word;

Implementation
{$R *.DFM}

procedure TFormFND.Button1Click(Sender: TObject);
var
y,i,j,a:integer;
```

```
x:real;

Begin
memo2.lines.clear;
memo3.lines.clear;
button2.Enabled:=false;
label3.caption:='Nome do arquivo:';
edit2.text:=edit1.text;
s:=edit1.text+'.est';
AssignFile(arqmatriz,s);
reset(arqmatriz);
read(arqmatriz,a);
numprocess:=a;
read(arqmatriz,a);
ultimono:=a;
ultimonoquine3:=a;
x:=exp(ln(2)*numprocess);
Y:=trunc(x);
setlength(novono,numprocess);
setlength(marcado,y*y);
setlength(matquine1,y*y,numprocess);
setlength(matquine2,y*y,numprocess);
setlength(matquine3,y,numprocess);
For i:=0 to ultimono do
  Begin
  s:='';
  For j:=0 to numprocess-1 do
    Begin
    read(arqmatriz,a);
    if a=1 then
      Begin
      matquine1[i,j]:='1';
      matquine3[i,j]:='1';
      s:=s+'1';
      End
    else
      Begin
      matquine1[i,j]:='0';
      matquine3[i,j]:='0';
      s:=s+'0';
      End;
    End;
  memo2.lines.add(s);
  End;
{memo2.lines.add('Arquivo aberto');}
closefile(arqmatriz);
button4.Enabled:=true;
label8.caption:=inttostr(ultimonoquine3+1);
label9.caption:='-';
label12.caption:='-';
End;

procedure TFormFND.Button4Click(Sender: TObject);

var
i,j:integer;

  procedure quine;
  var
  noatualquine,x,i,j:integer;
```

```
  procedure criar_novono;
  var
  i,j,m:integer;
  Begin
  For i:=0 to numprocess-1 do novono[i]:=matquine1[noatualquine,i];
  novono[g]:='-';
  m:=0;
  For i:=0 to ultimonoquine2-1 do
    Begin
    jaexiste:=0;
    For j:=0 to numprocess-1 do if novono[j]=matquine2[i,j] then inc(jaexiste);
    if jaexiste=numprocess then m:=1;
    End;
  if m=0 then
    Begin
    For j:=0 to numprocess-1 do matquine2[ultimonoquine2,j]:=novono[j];
    inc(ultimonoquine2);
    End;
  inc(p);
  End;

Begin
p:=0;
For noatualquine:=0 to ultimono-1 do
  For i:=noatualquine+1 to ultimono do
    Begin
    x:=0;g:=0;
    For j:=0 to numprocess-1 do
      if matquine1[noatualquine,j]<>matquine1[i,j] then
        Begin
        inc(x);
        g:=j;
        End;
    if x=1 then
      Begin
      marcado[noatualquine]:=true;
      marcado[i]:=true;
      criar_novono;
      End;
    End;
End;

procedure copiar_nos_nao_usados_para_novamatriz;
var
i,j:integer;
Begin
For i:=0 to ultimono do
  if not marcado[i] then
    Begin
    For j:=0 to numprocess-1 do matquine2[ultimonoquine2,j]:=matquine1[i,j];
    inc(ultimonoquine2);
    End;
End;

procedure trocamatriz_e_zera_variaveis;
var
i:integer;
Begin
mataux:=matquine1;
```

```
matquine1:=matquine2;
matquine2:=mataux;
For i:=0 to ultimonoquine2-1 do marcado[i]:=false;
ultimono:=ultimonoquine2-1;
ultimonoquine2:=0;
End;

procedure escolher;
var
i,j,z,cont,rep,quantrep,process:integer;

Begin
setlength(marcado,ultimono+1);
For i:=0 to ultimono do marcado[i]:=false;
z:=1;
Repeat
For i:=0 to ultimonoquine3 do
  Begin
  quantrep:=0;
  rep:=0;
  For j:=0 to ultimono do
    Begin
    cont:=0;
    For process:=0 to numprocess-1 do
      if (matquine1[j,process]='-') or (matquine1[j,process]=matquine3[i,process]) then inc(cont);
    if (cont=numprocess) then
      if marcado[j] then
        matquine3[i,0]:='-'
      else
        Begin
        inc(quantrep);
        rep:=j;
        End;
    End;
  if (quantrep=z) and (matquine3[i,0]<>'-') then
    Begin
    marcado[rep]:=true;
    matquine3[i,0]:='-';
    End;
  End;
{reconstruir estados}
cont:=-1;
For i:=0 to ultimonoquine3 do
  if matquine3[i,0]<>'-' then
    Begin
    inc(cont);
    For j:=0 to numprocess-1 do matquine3[cont,j]:=matquine3[i,j];
    End;
if cont=ultimonoquine3 then inc(z) else ultimonoquine3:=cont;
until ultimonoquine3=-1;
z:=0;
memo3.lines.add('FND');
For i:=0 to ultimono do
  Begin
  s:='';
  For j:=0 to numprocess-1 do s:=s+matquine1[i,j];
  if marcado[i]=true then
    Begin
    memo3.lines.add(s);
    inc(z);
```

```
        End;
      End;
   label9.caption:=inttostr(z);
   label12.caption:=inttostr(ultimono+1);
   End;

Begin
Button4.Enabled:=false;
tempo:=now;
Repeat
quine;
copiar_nos_nao_usados_para_novamatriz;
trocamatriz_e_zera_variaveis;
memo3.lines.clear;
until p=0;
memo3.lines.add('Implicativos Diretos');
For i:=0 to ultimono do
   Begin
   s:='';
   For j:=0 to numprocess-1 do s:=s+matquine1[i,j];
   memo3.lines.add(s);
   End;
memo3.lines.add('');
button2.Enabled:=true;
escolher;
tempo:=now-tempo;
decodetime(tempo,hora,min,seg,mseg);
memo3.lines.add('');
memo3.lines.add('Tempo de processamento');
memo3.lines.add(inttostr(hora)+':'+inttostr(min)+'` '+inttostr(seg)+'``  '+inttostr(mseg));
End;

procedure TFormFND.Button2Click(Sender: TObject);
var
a,i,j:integer;

Begin
a:=0;
s:=edit2.text+'.dfn';
AssignFile(arqmatriz,s);
rewrite(arqmatriz);
reset(arqmatriz);
For i:=0 to ultimono do
   if marcado[i]=true then inc(a);
write(arqmatriz,numprocess);
write(arqmatriz,a);
For i:=0 to ultimono do
   if marcado[i]=true then
      For j:=0 to numprocess-1 do
         if matquine1[i,j]='1' then
            Begin
            a:=1;
            write(arqmatriz,a);
            End
         else
            if matquine1[i,j]='0' then
               Begin
               a:=0;
               write(arqmatriz,a);
               End
```

```
          else
            Begin
            a:=2;
            write(arqmatriz,a);
            End;
closefile(arqmatriz);
label3.caption:='FND salva';
button2.Enabled:=false;
End;

End.
```

Anexo G – Código do Módulo Novo Algoritmo

```
unit fnddireto;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls;

type
  TFormFNDDireto = class(TForm)
    Memo1: TMemo;
    Memo2: TMemo;
    Memo3: TMemo;
    Label2: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    Button1: TButton;
    Calcular: TButton;
    Edit1: TEdit;
    Label5: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label9: TLabel;
    Label10: TLabel;
    Label1: TLabel;
    Memo4: TMemo;
    Label6: TLabel;
    Label11: TLabel;
    Label12: TLabel;
    Button2: TButton;
    procedure CalcularClick(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  End;

var
  FormFNDDireto: TFormFNDDireto;
  numprocess,a,restratual,ultimarestr,n,fndatual,ultimonoquine3: integer;
  matop: array of array of boolean;
  novarestr,marcado: array of boolean;
  s:string;
  temprecedencia,jaexiste: boolean;
  arqmatriz: file of integer;
  matfnd,matquine3: array of array of char;
  tempo: TDateTime;
  hora,min,seg,mseg: word;

Implementation

{$R *.DFM}
```

```
procedure TFormFNDDireto.Button1Click(Sender: TObject);
var
i,j:integer;

Begin
memo1.lines.clear;
memo2.lines.clear;
memo3.lines.clear;
memo4.lines.clear;
label10.caption:='-';
label12.caption:='-';
button2.Enabled:=false;
s:=edit1.text+'.mtr';
AssignFile(arqmatriz,s);
reset(arqmatriz);
read(arqmatriz,a);
numprocess:=a;
setlength(matop,numprocess,numprocess);
setlength(matop,numprocess,numprocess);
setlength(novarestr,numprocess);
setlength(matfnd,1,numprocess);
For i:=0 to numprocess-1 do
  Begin
  s:='';
  For j:=0 to numprocess-1 do
   Begin
   read(arqmatriz,a);
   if a=1 then
     Begin
     matop[i,j]:=true;
     matop[i,j]:=true;
     s:=s+'1'
     End
       else
     Begin
     matop[i,j]:=false;
     matop[i,j]:=false;
     s:=s+'0';
     End;
   End;
  memo1.lines.add(s);
  End;
closefile(arqmatriz);
s:=edit1.text+'.est';
AssignFile(arqmatriz,s);
reset(arqmatriz);
read(arqmatriz,a);
numprocess:=a;
read(arqmatriz,a);
ultimonoquine3:=a;
setlength(matquine3,ultimonoquine3+1,numprocess);
For i:=0 to ultimonoquine3 do
   Begin
   s:='';
   For j:=0 to numprocess-1 do
      Begin
      read(arqmatriz,a);
      if a=1 then
        Begin
```

```
        matquine3[i,j]:='1';
        s:=s+'1';
        End
      else
        Begin
        matquine3[i,j]:='0';
        s:=s+'0';
        End;
      End;
   memo4.lines.add(s);
   End;
{memo2.lines.add('Arquivo aberto');}
closefile(arqmatriz);
label9.caption:=inttostr(ultimonoquine3+1);
Calcular.Enabled:=true;
End;

procedure TFormFNDDireto.CalcularClick(Sender: TObject);
var
i,j,k:integer;

   procedure escolher;
   var
   i,j,z,cont,rep,quantrep,process:integer;

   Begin
   setlength(marcado,fndatual+1);
   For i:=0 to fndatual do marcado[i]:=false;
   z:=1;
   Repeat
   For i:=0 to ultimonoquine3 do
      Begin
      quantrep:=0;
      rep:=0;
      For j:=0 to fndatual do
         Begin
         cont:=0;
         For process:=0 to numprocess-1 do
            if (matfnd[j,process]='-') or (matfnd[j,process]=matquine3[i,process]) then inc(cont);
         if (cont=numprocess) then
            if marcado[j] then
               matquine3[i,0]:='-'
            else
               Begin
               inc(quantrep);
               rep:=j;
               End;
         End;
      if (quantrep=z) and (matquine3[i,0]<>'-') then
         Begin
         marcado[rep]:=true;
         matquine3[i,0]:='-';
         End;
      End;
   {reconstruir estados}
   cont:=-1;
   For i:=0 to ultimonoquine3 do
      if matquine3[i,0]<>'-' then
         Begin
         inc(cont);
```

```
        For j:=0 to numprocess-1 do matquine3[cont,j]:=matquine3[i,j];
        End;
    if cont=ultimonoquine3 then inc(z) else ultimonoquine3:=cont;
    until ultimonoquine3=-1;
    z:=0;
    memo2.lines.add('FND');
    For i:=0 to fndatual do
      Begin
      s:='';
      For j:=0 to numprocess-1 do s:=s+matfnd[i,j];
      if marcado[i]=true then
        Begin
        memo2.lines.add(s);
        inc(z);
        End;
      End;
    label10.caption:=inttostr(z);
    End;

Begin
memo2.lines.clear;
memo3.lines.clear;
Calcular.Enabled:=false;
tempo:=now;
{encontrando a primeira fnd}
For i:=0 to numprocess-1 do
    Begin
    temprecedencia:=false;
    For j:=0 to numprocess-1 do
      if matop[j,i] then temprecedencia:=true;
    if temprecedencia then matfnd[0,i]:='0'
      else matfnd[0,i]:='-';
    End;
{comparando precedencias}
restratual:=0;
ultimarestr:=numprocess;
Repeat
if (restratual>=numprocess) or (matfnd[0,restratual]='0') then
    Begin
    For i:=restratual+1 to ultimarestr-1 do
      Begin
      For j:=0 to numprocess-1 do
        if (matop[j,restratual]) or (matop[j,i]) then novarestr[j]:=true
          else novarestr[j]:=false;
      {verificando se nova restrição existe}
      jaexiste:=false;
      For k:=restratual to ultimarestr-1 do
        Begin
        n:=0;
        For j:=0 to numprocess-1 do
          if novarestr[j]=matop[j,k] then inc(n);
        if n=numprocess then jaexiste:=true;
        End;
      if not jaexiste then
        Begin
        inc(ultimarestr);
        setlength(matop,numprocess,ultimarestr);
        For j:=0 to numprocess-1 do
          matop[j,ultimarestr-1]:=novarestr[j];
        End;
```

```
      End;
    End;
inc(restratual);
until restratual=ultimarestr;
{mostrando nova matriz de restrições}
For i:=0 to numprocess-1 do
  Begin
  s:='';
  For j:=0 to ultimarestr-1 do
    if matop[i,j] then s:=s+'1' else s:=s+'0';
  memo3.lines.add(s);
  End;
{gerando fnd}
fndatual:=0;
For i:=0 to ultimarestr-1 do
  if (i>=numprocess) or (matfnd[0,i]='0') then
    Begin
    inc(fndatual);
    setlength(matfnd,fndatual+1,numprocess);
    For j:=0 to numprocess-1 do matfnd[fndatual,j]:=matfnd[0,j];
    For j:=0 to numprocess-1 do if matop[j,i] then matfnd[fndatual,j]:='1';
    For j:=0 to numprocess-1 do
      if matfnd[fndatual,j]='0' then
        Begin
        temprecedencia:=false;
        For k:=0 to numprocess-1 do
          if (matfnd[fndatual,k]<>'1') and (matop[k,j]) then
            temprecedencia:=true;
        if (not temprecedencia) then matfnd[fndatual,j]:='-';
        End;
    {verificando se fnd ja existe}
    jaexiste:=false;
    For j:=0 to fndatual-1 do
      Begin
      n:=0;
      For k:=0 to numprocess-1 do
        if matfnd[fndatual,k]=matfnd[j,k] then inc(n);
      if n=numprocess then jaexiste:=true;
      End;
    if jaexiste then dec(fndatual);
    End;
{mostrando implicativos}
memo2.lines.add('Implicativos Diretos');
For i:=0 to fndatual do
  Begin
  s:='';
  For j:=0 to numprocess-1 do s:=s+matfnd[i,j];
  memo2.lines.add(s);
  End;
memo2.lines.add('');
label12.caption:=inttostr(fndatual+1);
escolher;
tempo:=now-tempo;
decodetime(tempo,hora,min,seg,mseg);
memo2.lines.add('');
memo2.lines.add('Tempo de processamento');
memo2.lines.add(inttostr(hora)+':'+inttostr(min)+'` '+inttostr(seg)+'``  '+inttostr(mseg));
button2.Enabled:=true;
End;
```

```
procedure TFormFNDDireto.Button2Click(Sender: TObject);
var
i,j:integer;

Begin
a:=0;
s:=edit1.text+'.dfn';
AssignFile(arqmatriz,s);
rewrite(arqmatriz);
reset(arqmatriz);
For i:=0 to fndatual do
  if marcado[i]=true then inc(a);
write(arqmatriz,numprocess);
write(arqmatriz,a);
For i:=0 to fndatual do
  if marcado[i]=true then
    For j:=0 to numprocess-1 do
      if matfnd[i,j]='1' then
        Begin
        a:=1;
        write(arqmatriz,a);
        End
      else
        if matfnd[i,j]='0' then
          Begin
          a:=0;
          write(arqmatriz,a);
          End
        else
          Begin
          a:=2;
          write(arqmatriz,a);
          End;
closefile(arqmatriz);
button2.Enabled:=false;
End;

End.
```

Anexo H – Programação Linear para a Peça Motriz

A programação linear deste anexo foi feita utilizando o modelo padrão do software Lindo.

```
MIN
 77 X11 + 11 X21 + 72 X31 + 7 X41 + 97 X51 + 41 X61 +
 39 C12 + 58 C13 + 49 C14 + 24 C15 + 35 C16 +
 53 C21 + 85 C23 + 45 C24 + 72 C25 + 72 C26 +
 88 C31 + 45 C32 + 70 C34 + 62 C35 + 26 C36 +
 67 C41 + 35 C42 + 54 C43 + 0 C45 + 59 C46 +
 98 C51 + 58 C52 + 46 C53 + 55 C54 + 87 C56 +
 62 C61 + 88 C62 + 14 C63 + 47 C64 + 32 C65
SUBJECT TO
 X11 + X21 + X31 + X41 + X51 + X61 = 1
 X12 + X22 + X32 + X42 + X52 + X62 = 2
 X13 + X23 + X33 + X43 + X53 + X63 = 3
 X14 + X24 + X34 + X44 + X54 + X64 = 4
 X15 + X25 + X35 + X45 + X55 + X65 = 5
X12 - X11 >= 0
X22 - X21 >= 0
X32 - X31 >= 0
X42 - X41 >= 0
X52 - X51 >= 0
X62 - X61 >= 0
X13 - X12 >= 0
X23 - X22 >= 0
X33 - X32 >= 0
X43 - X42 >= 0
X53 - X52 >= 0
X63 - X62 >= 0
X14 - X13 >= 0
X24 - X23 >= 0
X34 - X33 >= 0
X44 - X43 >= 0
X54 - X53 >= 0
X64 - X63 >= 0
X15 - X14 >= 0
X25 - X24 >= 0
X35 - X34 >= 0
X45 - X44 >= 0
X55 - X54 >= 0
X65 - X64 >= 0
X11 + X22 - X21 - C12 <= 1
X11 + X32 - X31 - C13 <= 1
X11 + X42 - X41 - C14 <= 1
X11 + X52 - X51 - C15 <= 1
X11 + X62 - X61 - C16 <= 1
X21 + X12 - X11 - C21 <= 1
X21 + X32 - X31 - C23 <= 1
X21 + X42 - X41 - C24 <= 1
X21 + X52 - X51 - C25 <= 1
X21 + X62 - X61 - C26 <= 1
```

X31 + X12 - X11 - C31 <= 1
X31 + X22 - X21 - C32 <= 1
X31 + X42 - X41 - C34 <= 1
X31 + X52 - X51 - C35 <= 1
X31 + X62 - X61 - C36 <= 1
X41 + X12 - X11 - C41 <= 1
X41 + X22 - X21 - C42 <= 1
X41 + X32 - X31 - C43 <= 1
X41 + X52 - X51 - C45 <= 1
X41 + X62 - X61 - C46 <= 1
X51 + X12 - X11 - C51 <= 1
X51 + X22 - X21 - C52 <= 1
X51 + X32 - X31 - C53 <= 1
X51 + X42 - X41 - C54 <= 1
X51 + X62 - X61 - C56 <= 1
X61 + X12 - X11 - C61 <= 1
X61 + X22 - X21 - C62 <= 1
X61 + X32 - X31 - C63 <= 1
X61 + X42 - X41 - C64 <= 1
X61 + X52 - X51 - C65 <= 1
X12 - X11 + X23 - X22 - C12 <= 1
X12 - X11 + X33 - X32 - C13 <= 1
X12 - X11 + X43 - X42 - C14 <= 1
X12 - X11 + X53 - X52 - C15 <= 1
X12 - X11 + X63 - X62 - C16 <= 1
X22 - X21 + X13 - X12 - C21 <= 1
X22 - X21 + X33 - X32 - C23 <= 1
X22 - X21 + X43 - X42 - C24 <= 1
X22 - X21 + X53 - X52 - C25 <= 1
X22 - X21 + X63 - X62 - C26 <= 1
X32 - X31 + X13 - X12 - C31 <= 1
X32 - X31 + X23 - X22 - C32 <= 1
X32 - X31 + X43 - X42 - C34 <= 1
X32 - X31 + X53 - X52 - C35 <= 1
X32 - X31 + X63 - X62 - C36 <= 1
X42 - X41 + X13 - X12 - C41 <= 1
X42 - X41 + X23 - X22 - C42 <= 1
X42 - X41 + X33 - X32 - C43 <= 1
X42 - X41 + X53 - X52 - C45 <= 1
X42 - X41 + X63 - X62 - C46 <= 1
X52 - X51 + X13 - X12 - C51 <= 1
X52 - X51 + X23 - X22 - C52 <= 1
X52 - X51 + X33 - X32 - C53 <= 1
X52 - X51 + X43 - X42 - C54 <= 1
X52 - X51 + X63 - X62 - C56 <= 1
X62 - X61 + X13 - X12 - C61 <= 1
X62 - X61 + X23 - X22 - C62 <= 1
X62 - X61 + X33 - X32 - C63 <= 1
X62 - X61 + X43 - X42 - C64 <= 1
X62 - X61 + X53 - X52 - C65 <= 1
X13 - X12 + X24 - X23 - C12 <= 1
X13 - X12 + X34 - X33 - C13 <= 1
X13 - X12 + X44 - X43 - C14 <= 1
X13 - X12 + X54 - X53 - C15 <= 1
X13 - X12 + X64 - X63 - C16 <= 1
X23 - X22 + X14 - X13 - C21 <= 1
X23 - X22 + X34 - X33 - C23 <= 1
X23 - X22 + X44 - X43 - C24 <= 1
X23 - X22 + X54 - X53 - C25 <= 1
X23 - X22 + X64 - X63 - C26 <= 1

X33 - X32 + X14 - X13 - C31 <= 1
X33 - X32 + X24 - X23 - C32 <= 1
X33 - X32 + X44 - X43 - C34 <= 1
X33 - X32 + X54 - X53 - C35 <= 1
X33 - X32 + X64 - X63 - C36 <= 1
X43 - X42 + X14 - X13 - C41 <= 1
X43 - X42 + X24 - X23 - C42 <= 1
X43 - X42 + X34 - X33 - C43 <= 1
X43 - X42 + X54 - X53 - C45 <= 1
X43 - X42 + X64 - X63 - C46 <= 1
X53 - X52 + X14 - X13 - C51 <= 1
X53 - X52 + X24 - X23 - C52 <= 1
X53 - X52 + X34 - X33 - C53 <= 1
X53 - X52 + X44 - X43 - C54 <= 1
X53 - X52 + X64 - X63 - C56 <= 1
X63 - X62 + X14 - X13 - C61 <= 1
X63 - X62 + X24 - X23 - C62 <= 1
X63 - X62 + X34 - X33 - C63 <= 1
X63 - X62 + X44 - X43 - C64 <= 1
X63 - X62 + X54 - X53 - C65 <= 1
X14 - X13 + X25 - X24 - C12 <= 1
X14 - X13 + X35 - X34 - C13 <= 1
X14 - X13 + X45 - X44 - C14 <= 1
X14 - X13 + X55 - X54 - C15 <= 1
X14 - X13 + X65 - X64 - C16 <= 1
X24 - X23 + X15 - X14 - C21 <= 1
X24 - X23 + X35 - X34 - C23 <= 1
X24 - X23 + X45 - X44 - C24 <= 1
X24 - X23 + X55 - X54 - C25 <= 1
X24 - X23 + X65 - X64 - C26 <= 1
X34 - X33 + X15 - X14 - C31 <= 1
X34 - X33 + X25 - X24 - C32 <= 1
X34 - X33 + X45 - X44 - C34 <= 1
X34 - X33 + X55 - X54 - C35 <= 1
X34 - X33 + X65 - X64 - C36 <= 1
X44 - X43 + X15 - X14 - C41 <= 1
X44 - X43 + X25 - X24 - C42 <= 1
X44 - X43 + X35 - X34 - C43 <= 1
X44 - X43 + X55 - X54 - C45 <= 1
X44 - X43 + X65 - X64 - C46 <= 1
X54 - X53 + X15 - X14 - C51 <= 1
X54 - X53 + X25 - X24 - C52 <= 1
X54 - X53 + X35 - X34 - C53 <= 1
X54 - X53 + X45 - X44 - C54 <= 1
X54 - X53 + X65 - X64 - C56 <= 1
X64 - X63 + X15 - X14 - C61 <= 1
X64 - X63 + X25 - X24 - C62 <= 1
X64 - X63 + X35 - X34 - C63 <= 1
X64 - X63 + X45 - X44 - C64 <= 1
X64 - X63 + X55 - X54 - C65 <= 1
X15 - X14 - X25 - C12 <= 0
X15 - X14 - X35 - C13 <= 0
X15 - X14 - X45 - C14 <= 0
X15 - X14 - X55 - C15 <= 0
X15 - X14 - X65 - C16 <= 0
X25 - X24 - X15 - C21 <= 0
X25 - X24 - X35 - C23 <= 0
X25 - X24 - X45 - C24 <= 0
X25 - X24 - X55 - C25 <= 0
X25 - X24 - X65 - C26 <= 0

```
X35 - X34 - X15 - C31 <= 0
X35 - X34 - X25 - C32 <= 0
X35 - X34 - X45 - C34 <= 0
X35 - X34 - X55 - C35 <= 0
X35 - X34 - X65 - C36 <= 0
X45 - X44 - X15 - C41 <= 0
X45 - X44 - X25 - C42 <= 0
X45 - X44 - X35 - C43 <= 0
X45 - X44 - X55 - C45 <= 0
X45 - X44 - X65 - C46 <= 0
X55 - X54 - X15 - C51 <= 0
X55 - X54 - X25 - C52 <= 0
X55 - X54 - X35 - C53 <= 0
X55 - X54 - X45 - C54 <= 0
X55 - X54 - X65 - C56 <= 0
X65 - X64 - X15 - C61 <= 0
X65 - X64 - X25 - C62 <= 0
X65 - X64 - X35 - C63 <= 0
X65 - X64 - X45 - C64 <= 0
X65 - X64 - X55 - C65 <= 0
- X21 - X61 - 6I11 >= - 6
X11 - X61 - 6I21 >= - 5
X11 + X21 + X31 + X41 + X51 - 6I31 >= - 1
- X22 - X62 - 6I12 >= - 6
X12 - X62 - 6I22 >= - 5
X12 + X22 + X32 + X42 + X52 - 6I32 >= - 1
- X23 - X63 - 6I13 >= - 6
X13 - X63 - 6I23 >= - 5
X13 + X23 + X33 + X43 + X53 - 6I33 >= - 1
- X24 - X64 - 6I14 >= - 6
X14 - X64 - 6I24 >= - 5
X14 + X24 + X34 + X44 + X54 - 6I34 >= - 1
- X25 - X65 - 6I15 >= - 6
X15 - X65 - 6I25 >= - 5
X15 + X25 + X35 + X45 + X55 - 6I35 >= - 1
I11 + I21 + I31 >= 1
I12 + I22 + I32 >= 1
I13 + I23 + I33 >= 1
I14 + I24 + I34 >= 1
I15 + I25 + I35 >= 1
END
INT 105
```

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ADMINISTRAÇÃO
Programa de Pós-Graduação em Administração
Grupo de Estudos em Sistemas de Informação e de Apoio à Decisão

REPRESENTAÇÃO NA FORMA NORMAL DISJUNTIVA PARA A FLEXIBILIDADE DE SEQÜÊNCIA NA MANUFATURA

por

Leonardo Rosa Rohde

Porto Alegre, Agosto de 2002.