UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

PROGRAMAÇÃO DA PRODUÇÃO: UMA ABORDAGEM UTILIZANDO ALGORITMOS GENÉTICOS

Tese submetida à Universidade Federal de Santa Catarina para a obtenção do título de Doutor em Engenharia de Produção

FERNANDO ÁLVARO OSTUNI GAUTHIER

Florianópolis, outubro de 1993

PROGRAMAÇÃO DA PRODUÇÃO:UMA ABORDAGEM UTILIZANDO ALGORITMOS GENÉTICOS

FERNANDO ÁLVARO OSTUNI GAUTHIER

Esta tese foi julgada adequada para a obtenção do título de DOUTOR EM ENGENHARIA DE PRODUÇÃO e aprovada em sua forma final pelo Programa de Pós-Graduação em Engenharia de Produção.

PROF. NERI DOS SANTOS, Dr. Ing.
Coordenador do Curso

BANCA EXAMINADORA:

PROF. RICARDO MIRANDA BARCIA, Ph.D.
Orientador

PROF. OSAMA EYADA, Ph.D.
Examinador Externo

PROF. SURESH KHATOR , Ph.D.
Examinador Externo

PROF. ANTÔNIO GALVÃO NOVAES, Dr.

PROF. EDGAR AUGUSTO LANZER, Ph.D.

PROF. HERMANN ADOLF H. LÜCKE, Dr. Ing.
Moderador

À memória de minha avó
Maria Jorgelina Picart de Gauthier

AGRADECIMENTOS

Este trabalho foi desenvolvido com o constante incentivo e apoio decisivo de meu orientador Prof. Ricardo Miranda Barcia. Ao Prof. Barcia agradeço não apenas os ensinamentos acadêmicos, mas, sobretudo, a transmissão de uma filosofia de ação e de conduta.

Sou grato também aos amigos Alceu Ribeiro Alves e Édis Mafra Lapolli, com quem tive o prazer de trabalhar durante todo o decorrer do curso. Com eles dividi angústias e preocupações. Com eles compartilho a alegria pela conclusão dessa etapa de formação.

Os amigos Flávio Rubens Lapolli, Lia Caetano Bastos e Rogério Cid Bastos complementam o meu grupo de trabalho. A eles agradeço o apoio, o incentivo, a troca de idéias e, sobremaneira, a constante amizade.

Aos bolsistas Adriano Coser, Rodrigo Beck Cabral, César Silvestre, José Ricardo Alves e Fernando Schramin agradeço a dedicação com que trabalharam comigo durante o desenvolvimento deste trabalho.

Aos Professores do Departamento de Engenharia de Produção e Sistemas agradeço o convívio e o estímulo para o desenvolvimento de trabalhos científicos.

Agradeço a Zelita Chaves de Souza, Secretária do Programa de Pós-Graduação e demais funcionários do Departamento de Engenharia de Produção pelo apoio continuado que recebi durante todo o curso.

Aos Professores Osama Eyada - Virginia Polytechnic Institute and State University, Virginia, USA - e Suresh Kathor - University of South Florida, Florida, USA - examinadores externos, agradeço pelas sugestões recebidas para melhoria deste trabalho.

Aos Professores Antônio Galvão Novaes e Edgar Augusto Lanzer, membros da Banca, agradeço pela inestimável contribuição e amizade recebida durante todo o tempo que desenvolvi o trabalho de tese.

Agradeço ao Prof. Hermann Adolf Harry Lucke, moderador, por sua atuação e condução dos trabalhos de apresentação da tese.

RESUMO

O controle e o gerenciamento da produção estão entre os principais fatores que influenciam a produtividade industrial. As empresas devem se adaptar às condições de mercado, que mudam constantemente, afetando o tempo disponível para a tomada de decisões.

Um algoritmo para solucionar problemas de programação da produção em diferentes ambientes "flow shop", podendo levar em consideração diferentes critérios, apresenta-se como uma contribuição para essa área.

Neste trabalho é apresentada uma abordagem baseada em algoritmos genéticos que dadas as suas características de paralelismo implícito e robustez, mostra resultados promissores.

O algoritmo proposto separa a função objetivo da sua rotina principal, o que proporciona vantagens sobre as soluções heurísticas que são desenvolvidas para funções objetivo específicas. Essa separação permite rápida adaptação do algoritmo a qualquer ambiente do tipo "flow shop".

O sistema computacional desenvolvido foi utilizado para resolver problemas gerados aleatoriamente, através de três diferentes critérios de programação. Dadas as suas características híbridas, o algoritmo obtém soluções iguais ou melhores que as das regras de programação heurísticas. Os resultados dos testes mostraram a viabilidade do algoritmo.

ABSTRACT

There has been a lot of work done on production scheduling problems during the last 40 years. A literature survey will reveal the existence of exact methods, including complete enumeration, branch and bound techniques or integer programming, and heuristic methods.

This work proposes an approach based on genetic algorithms that shows promising results for solving the general Flow Shop Problem owing mainly to this technique implicit parallelism and robustness.

A computational package was developed and used to solve randomly generated problems with three different criteria to optimiza the schedule. Due to the hybrid characteristics the algorithm obtains solutions better or equal to those of heuristic rules.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

1. INTRODUÇÃO

1.1 Origem do Trabalho

O controle e o gerenciamento da produção estão entre os principais fatores que influenciam a produtividade industrial. As empresas devem se adaptar às condições de mercado, que mudam constantemente, afetando o tempo disponível para a tomada de decisões.

Para SMITH, FOX e OW (1986), o obstáculo mais significativo para aumentar o desempenho da fábrica está relacionado com a complexidade associada com a construção e manutenção de boas programações da produção. Os problemas relacionados com a programação da produção implicam em determinar a ordem na qual um conjunto de produtos,ou lotes de produtos, devem ser processados por uma máquina ou por um conjunto de máquinas.

O problema da programação da produção vem sendo estudado desde a década de cinquenta, quando JOHNSON (SEN e GUPTA, 1984) propôs uma técnica para encontrar a solução ótima no problema de minimização do tempo máximo de processamento ("makespan") de um conjunto de produtos ("jobs"), em ambientes do tipo flow shop com duas ou três máquinas.

A literatura apresenta inúmeros trabalhos e revisões bibliográficas relacionados com o tema. O número desses trabalhos "tem alcançado tal extensão que é impraticável, senão impossível, revisar a literatura num único trabalho"(SEN e GUPTA, 1984).

McKAY, SAFAYENI e BUZACOTT (1988) concluem que os avanços na teoria e nas técnicas de solução não têm gerado os resultados práticos esperados. Segundo esses autores, a formulação teórica do problema parece não captar a essência das dificuldades enfrentadas pelos programadores da produção.

O problema de programação da produção em ambientes "flow shop", tem merecido a atenção de pesquisadores da área e diversos métodos de solução vêm sendo apresentados. Além das técnicas tradicionais de pesquisa

operacional, como, por exemplo, programação inteira, "branch and bound", vários procedimentos heurísticos têm sido propostos. Esses procedimentos têm-se constituído como uma alternativa à programação matemática, dado o elevado grau de complexidade computacional verificado nesta área. As heurísticas apresentadas, em sua grande maioria, abordam o problema da programação da produção sob uma única ótica.

A alta complexidade dos problemas de programação da produção tem concentrado as pesquisas na otimização de medidas individuais de desempenho (DANIELS e CHAMBERS, 1990). As técnicas de solução estão direcionadas para ambientes específicos. Nos casos em que devem ser utilizadas para dois ou mais objetivos, seu desempenho deixa a desejar.

Em um ambiente "flow shop" e discutindo a minimização do "makespan", HAN e DEJAX (1991) afirmam, por exemplo, que a hipótese de demoras associadas a trocas de produtos numa determinada máquina ("changeover") independentes da seqüência analisada, em muitas abordagens, não se verifica em algumas aplicações práticas.

Nesse contexto, novas abordagens para o problema de programação da produção, tais como o uso de redes neuronais, de sistemas inteligentes, etc. são objetos de constantes pesquisas na área.

Os algorithmos genéticos, propostos por HOLLAND, são uma técnica alternativa, baseada na genética e seleção natural, que, dadas suas características e propriedades teoricamente provadas, apresenta-se como promissora com relação à solução de problemas de progamação da produção. Além disso, esses algoritmos permitem a adaptação de diversas funções objetivo, uma vez que, claramente, separam os critérios de otimização do algoritmo principal.

1.2 Objetivos do Trabalho

Essa pesquisa tem como objetivo geral:

• Analisar a aplicação de algoritmos genéticos para solução de problemas de programação da produção em ambientes "flow shop", objetivando a otimização de diferentes critérios de avaliação.

Concomitantemente, propõe-se:

• Desenvolver um algoritmo genético adaptado ao problema de programação da produção em ambiente "flow shop";

• Implementar computacionalmente o algoritmo desenvolvido;

• Propor e implementar uma metodologia para calibração do algoritmo genético desenvolvido.

1.3 Justificativa do Trabalho

Os problemas da programação da produção apresentam uma forte complexidade. Para programar "n" ordens em um recurso, existem n! alternativas.

A natureza combinatorial do problema faz com que os tempos envolvidos na procura de soluções sejam extremamente altos, muitas vezes tornando as abordagens exatas de pouca utilidade, a nível prático. Dessa forma, tornam-se importantes abordagens que, apesar de não garantir a melhor solução, forneçam soluções próximas a essa, em tempos compatíveis com o ambiente fabril.

Os algoritmos genéticos, desenvolvidos a partir dos trabalhos de HOLLAND, têm mostrado eficiência em resolver problemas de análises combinatoriais, nos quais o espaço de solução cresce exponencialmente. Na sua forma tradicional, essa eficiência é demonstrada teoricamente.

CLEVELAND e SMITH (1989) têm considerado a utilização de algoritmos genéticos como um modo de programar liberação de tarefas em uma unidade de manufatura denominada setor. WITLEY, STARKWEATHER e FUQUAY (1989) utilizam um operador recombinado e avaliam o problema da

programação da produção a partir de soluções para o problema do caixeiro viajante.

Na mesma área, mas visando descobrir heurísticas para programação da produção, HILLIARD et al (1987) propõem um sistema classificador para inferir heurísticas adequadas para programação da produção baseado em algoritmos genéticos. DAVIS (1985) estudou o problema de programação da produção, em ambinete "job shop", também com essa abordagem.

A aplicação de algoritmos genéticos para problemas de programação da produção tem conduzido a resultados que podem ser considerados satisfatórios e promissores. Contudo, tratam de problemas específicos, como, por exemplo, liberação de ordens de produção para um setor, dessa forma, verificando-se a necessidade de uma abordagem mais genérica e robusta.

As potencialidades da técnica de algoritmos genéticos evidenciam a possibilidade de desenvolver aplicações robustas que se comportam de maneira eficiente em um conjunto de problemas.

Um algoritmo genético para solucionar problemas de programação da produção em diferentes ambientes "flow shop", podendo levar-se em consideração diferentes critérios, apresenta-se como uma contribuição para essa área.

Os algoritmos genéticos utilizam uma série de parâmetros que devem ser calibrados de acordo com o problema. Ao se propor um algoritmo genético para vários problemas, atenção deve ser dada a essa calibração, que requer a intervenção de um especialista no uso dessa técnica. Assim, a utilização por vários usuários, com diferentes níveis de conhecimento, requer a inclusão de um método de auto-calibração. Nesse contexto, é apresentado um módulo baseado em raciocínio difuso para a calibração do algoritmo, de acordo com as características do problema a ser solucionado.

1.4 Estrutura do Trabalho

O presente trabalho encontra-se estruturado em dez capítulos.

Neste capítulo, a motivação e os objetivos da pesquisa são descritos. Seu caráter é introdutório, sendo também ressaltada a sua justificativa e apresentada a estrutura.

No segundo capítulo, apresenta-se uma abordagem teórica sobre programação da produção.

O terceiro capítulo é dedicado à fundamentação teórica dos algoritmos genéticos.

No quarto capítulo é apresentado o algoritmo genético proposto para programação da produção, onde inicialmente o problema é definido e posteriormente o algoritmo é desenvolvido.

A calibração do algoritmo genético proposto, utilizando raciocínio difuso, é apresentada no sétimo capítulo.

O sexto capítulo é dedicado ao desenvolvimento computacional do algoritmo proposto, bem como, do método de calibração.

Os resultados obtidos e sua avaliação são discutidos no sétimo capítulo.

O capítulo seguinte, o oitavo, apresenta as conclusões do referido trabalho e as recomendações para futuras pesquisas.

Finalmente, é apresentada a bibliografia utilizada, bem como, a citada neste trabalho.

2. PROGRAMAÇÃO DA PRODUÇÃO

2.1 Introdução

No presente capítulo, o problema de programação da produção é definido, procurando-se levantar as suas peculiaridades no que tange às informações disponíveis, às variáveis envolvidas, às abordagens para solução (incluindo modelos, estratégias, algoritmos) e aos resultados alcançados. Com essa finalidade, são analisadas as abordagens ótimas, abordagens através de regras de sequenciamento, de inteligência artificial e de algoritmos genéticos.

2.2 Conceito de Programação da Produção

A programação da produção se situa dentro de um contexto mais geral de gerenciamento da produção. Existem duas grandes filosofias de administração da produção e dos materiais: filosofia "just in time" e filosofia "just in case". A primeira, desenvolvida inicialmente pelas empresas japonesas, constitui uma estratégia que tenta reduzir custos, pela rápida adequação à demanda, diminuindo dessa forma as quantidades de matérias primas, produtos em processo e produtos acabados. A segunda, mais desenvolvida nos Estados Unidos e na Europa Ocidental, tem como idéia geral minimizar a ociosidade dos meios de produção, baseando-se, principalmente, na otimização da produção, pela fabricação em grandes lotes. Dessa maneira, existem grandes quantidades de produtos em processo e produtos acabados (ANTUNES JÚNIOR, KLIEMANN NETO e LIMA, 1990).

A arquitetura clássica de um sistema de planejamento da produção envolve três níveis de decisão (GELDERS e VAN WASSENHOVE, 1982) (BENSANA, 1987):

- Primeiro nível: decisões agregadas sobre produção e capacidade levam à determinação do programa de produção;

• Segundo nível: uma vez fixado o programa mestre, é necessário determinar as quantidades a serem produzidas (ou compradas) dos diferentes componentes;

• Terceiro nível: dadas as quantidades e datas de entrega dos diferentes componentes, devem-se elaborar os seus programas de produção e alocar os recursos necessários.

O problema de programação da produção, objeto desta pesquisa, enquadra-se no terceiro nível, e pode ser definido como:

" Dado um conjunto de equipamentos e restrições tecnológicas, e dadas as necessidades de produção em termos de quantidade e qualidade do produto, e restrições de tempo, encontrar uma sequência viável de operações nos vários equipamentos, que satisfaça as necessidades de produção" (BENSANA, BELL e DUBOIS, 1988).

2.3 Classificação dos Problemas de Programação da Produção

Várias formas de classificação têm sido propostas para os problemas de programação da produção. GRAVES (1981) propõe três dimensões para classificar os problemas de programação da produção:

- geração de pedidos;
- complexidade do processo produtivo;
- critério de programação.

A primeira dimensão, também enfatizada por HENDRY e KINGSMAN (1989) com relação ao planejamento da produção, refere-se à origem dos pedidos que chegam para serem programados. Esses pedidos podem ser gerados diretamente, por ordens de compra de clientes, ou indiretamente, por decisões de substituição de estoques vendidos. Essa distinção é referida como "open shop" e "closed shop", respectivamente. O problema de programação da produção é bastante diferente segundo a origem dos pedidos de produção. No caso de "open shop", na sua forma mais simples, constitui-se na busca de uma

seqüência de produção. Já no caso de "closed shop", devem ser incluídos o dimensionamento de lotes e as políticas de substituição de estoques.

Várias características de empresas que produzem para estoques e para ordens são mostradas na TABELA 2.1.

Característica	Para Estoque	Por Ordens
produtos	muitos produtos	poucos produtos
recursos	mão-de-obra e maquinaria especializada	máquinas multitarefas e mão-de-obra flexível
demanda	demanda pode ser prevista	demanda raramente pode ser prevista
planejamento da capacidade	baseada na demanda prevista. Planejada com antecedência.	baseada no recebimento de ordens. Não pode ser planejada com antecedência

TABELA 2.1 - Comparação entre as Empresas que Produzem para Substituir Estoques e Empresas que Produzem por Ordens dos Clientes
Fonte: Adaptada de HENDRY e KINGMAN, 1989

A segunda dimensão, complexidade do processamento, relaciona-se com o número de etapas de processamento associadas a cada item ou tarefa. Podem-se caracterizar os seguintes tipos de problemas de programação da produção:

- um estágio, um processador;
- um estágio, processadores paralelos;
- múltiplos estágios, "flow shop";
- múltiplos estágios, "job shop".

O problema de um estágio com um processador (máquina ou operário) é o mais simples de todos, pois as tarefas devem ser realizadas num único processador e requerem somente uma única operação para serem completadas.

O caso de um estágio com processadores paralelos é similar ao anterior. Cada item envolve uma única operação, que pode ser executada em qualquer um dos equipamentos disponíveis.

Nas situações de múltiplos estágios, cada produto ou lote devem ser processados numa série de equipamentos, onde existe uma ordem de precedência das etapas de processamento. As soluções viáveis são aquelas que satisfazem as relações de precedência.

O ambiente de produção é definido como "flow shop" quando todos os produtos devem ser processados na mesma sequência em um conjunto de equipamentos. Nesse tipo de estrutura de produção existe um único sentido no fluxo, desde as matérias primas até os produtos acabados. Alguns produtos podem começar seu processamento por máquinas diferentes da primeira, não sendo necessário, também, que exista uma máquina comum para o último processamento.

No problema caracterizado como "job shop", não existem restrições nas etapas de processamento de um produto. Os produtos também não apresentam um fluxo único podendo ser permitidas rotas alternativas.

A terceira dimensão proposta por GRAVES (1981), ou seja, o critério de programação, refere-se ao modo como a programação da produção será avaliada, distinguindo-se como critérios o seu custo e seu desempenho. O custo associa a um determinado programa de produção, custos de "setup", estoques, falta de estoques, etc. O desempenho do programa pode ser avaliado de várias formas, como nível de utilização dos recursos produtivos, percentagem de tarefas concluídas com atraso, tempo médio de processamento de tarefas ou itens, tempo médio ou máximo de atraso, etc. Na maioria dos ambientes de produção, a programação da produção se baseia numa mistura de ambos os critérios.

Ainda segundo GRAVES (1981), poderiam ser incluídas mais duas dimensões: segundo a natureza da especificação dos pedidos e do ambiente de produção. A determinação dos pedidos pode ser feita de forma determinística ou estocástica (por exemplo, os tempos de processamento podem ser conhecidos ou randômicos ou, no caso de "closed shop", a demanda pode ser aleatória e, em consequência, as necessidades de substituição dos estoques também). No que

se refere ao ambiente, a programação pode ser realizada em ambientes estáticos ou dinâmicos. No primeiro caso, os pedidos de produção são perfeitamente conhecidos e não sofrem nenhuma mudança. No segundo caso, novos pedidos podem ser adicionados e a programação deve ser realizada levando em consideração esses possíveis pedidos adicionais, que podem ser realizados em períodos futuros.

2.4 Complexidade do Problema de Programação da Produção

Os problemas de programação da produção apresentam uma forte complexidade. Para programar n ordens em um recurso, existem n! alternativas. No caso de "job shop", com ***n*** ordens e ***m*** máquinas, existem $(n!)^m$ possibilidades (BENSANA, 1986).

A complexidade de um algoritmo é definida como sendo o limite superior do número de operações necessárias. A complexidade de um problema é aquela do melhor algoritmo conhecido para resolvê-lo.

"Diz-se que a função f(n) tem uma função de complexidade de tempo O(g(n)) sempre que existe uma constante **c** tal que:

$$| f(n) | < c |g(n)|$$

para todos os valores de n>0. Um algoritmo com tempo polinomial é definido como aquele cuja função de complexidade de tempo é O(p(n)), para alguma função polinomial p, onde n é usado para denotar o comprimento da entrada. Qualquer algoritmo cuja função de complexidade de tempo não possa ser limitada dessa maneira é chamado como algoritmo de tempo exponencial" (GAREY e JOHNSON, 1979).

A TABELA 2.2, a seguir, apresenta uma comparação de várias funções de complexidade de tempos polinomiais e exponenciais.

Função de complexidade de tempo	Tamanho n				
	10	20	30	50	60
n	0,00001 segundos	0,00002 segundos	0,00003 segundos	0,00005 segundos	0,00006 segundos
n^2	0,0001 segundos	0,0004 segundos	0,0009 segundos	0,0025 segundos	0,0036 segundos
n^3	0,001 segundos	0,008 segundos	0,027 segundos	0,125 segundos	0,216 segundos
n^5	0,1 segundos	3,2 segundos	24,3 segundos	5,2 segundos	13,0 segundos
2^n	0,001 segundos	1 segundos	17,9 minutos	35,7 anos	366 séculos
3^n	0,59 segundos	58 minutos	6,5 anos	2×10^8 séculos	$1,3 \times 10^{13}$ séculos

TABELA 2.2 - Comparação de várias Funções de Complexidade de Tempo Polinomiais e Exponenciais
Fonte: Adaptada de GAREY e JOHNSON, 1979

A classe dos problemas não polinomiais mais "duros" ou difíceis é denominada classe dos problemas NP-completos, e a ela pertencem muitos problemas combinatórios, o problema do caixeiro viajante, etc. Segundo PAPADIMITRIOU e STEIGLITZ (1982), para provar que um problema e NP-completo deve-se mostrar que:

- o problema é NP;
- todos os outros problemas NP polinomiais se transformam no problema.

O problema de programação da produção em ambientes do tipo "flow shop" tem sido descrito como sendo NP-completo (GAREY e JOHNSON, 1979).

A verificação de que um problema pertence à classe dos NP-completos é usualmente o começo do trabalho. O conhecimento de que o mesmo é NP-completo proporciona informação valiosa sobre quais as linhas de

abordagem que têm potencial de serem mais produtivas. Assim, a procura de algoritmos eficientes e exatos, certamente, deve ter baixa prioridade. Torna-se mais apropriado concentrar-se em outras abordagens menos ambiciosas (GAREY e JOHNSON, 1979), como relaxar o problema de alguma maneira, de forma que a maioria das especificações dos componentes sejam alcançadas. "Quanto mais geral um problema, mais difícil é resolvê-lo" (PAPADIMITRIOU e STEIGLITZ, 1982).

2.5 Abordagens Dadas ao Problema

Muitas são as abordagens propostas para a solução do problema de programação da produção. KANET e ADELSBERGER (1987) distinguem dois tipos de abordagens:

- abordagens algorítmicas;
- abordagens de reformulação.

No primeiro grupo, são classificados os métodos que incluem "formulação do problema de programação no contexto clássico de programação matemática " (KANET e ADELSBERGER, 1987). Essa abordagem concentra-se na metodologia de solução e considera a formulação do problema como dada. A segunda abordagem difere da anterior em dois sentidos. Em primeiro lugar, o problema pode ser formulado de forma que o objetivo principal seja encontrar uma solução satisfatória. Em segundo lugar, o problema é reformulado, quando não existe solução viável ou seja muito difícil encontrá-la.

HENDRY e KINGSMAN (1989) classificam as abordagens para a solução do problema em três categorias:

- abordagens ótimas;
- regras heurísticas de atendimento;
- aplicações de inteligência artificial.

As abordagens ótimas, desenvolvidas em Pesquisa Operacional, constroem modelos matemáticos do problema, procurando encontrar a melhor

solução. Estas abordagens desenvolvem algoritmos bastante rígidos e, dada a natureza combinatorial do problema, torna-se difícil considerar todas as variáveis envolvidas e, para problemas práticos, o tempo de processamento necessário as faz impraticáveis.

As regras heurísticas de atendimento não fornecem uma programação das tarefas, mas permitem escolher, dentre as que estão na fila de espera, qual a próxima a ser processada no posto de trabalho.

Dentro da abordagem de inteligência artificial, a metodologia mais utilizada é a de sistemas especialistas. Os sistemas especialistas são sistemas computacionais que permitem resolver problemas não estruturados, utilizando para tal fim o conhecimento de especialistas na área do problema, e o conhecimento sobre o ambiente.

Essa classificação será utilizada, com o objetivo de estudar detalhadamente as abordagens dadas ao problema.

2.5.1 Abordagens Ótimas

Existem várias formas de representar o problema. Entre elas, a formulação matemática e a representação através de um grafo disjuntivo são as mais utilizadas (BENSANA, 1987).

Na programação matemática, a forma mais utilizada é programação linear inteira. Nas abordagens ótimas, mesmo nos casos de um estágio, é necessário realizar simplificações adicionais. Nas soluções, são utilizados busca em árvores com "branch and bound", programação dinâmica e programação inteira. "Poucos desses métodos poderiam ser aplicados a uma unidade real de produção" (HENDRY e KINGSMAN, 1989).

No caso de um grafo disjuntivo, o problema é representado através de um grafo orientado e um conjunto de arcos não orientados, que representam tarefas não ordenadas. Essa representação equivale a um problema de programação inteira, onde os arcos representam as variáveis de decisão.

2.5.2 Regras Heurísticas de Atendimento

Um levantamento do estado da arte em regras heurísticas para problemas do tipo "job shop" é apresentado por BLACKSTONE, PHILLIPS e YOGG (1982), onde as regras são classificadas em quatro grupos:

- regras envolvendo tempo de processamento (por exemplo, a tarefa com menor tempo de processamento deve ser executada primeiro) ;

- regras envolvendo data de entrega (por exemplo, a tarefa que apresentar a menor folga entre o tempo de processamento faltante e a data de entrega deve ser processada primeiro) ;

- regras que não envolvem nem data de entrega nem tempo de processamento (por exemplo, a primeira tarefa a chegar ao centro de trabalho deve ser processada primeiro) ;

- regras envolvendo duas ou mais das regras anteriores (por exemplo, soma ponderada da folga por operação faltante e tempo de processamento faltante).

As regras heurísticas de atendimento têm sido testadas, utilizando diferentes critérios de avaliação. "Existe a necessidade de se pesquisar muito mais, para identificar claramente a regra que deve ser utilizada para cada tipo de ambiente produtivo "(HENDRY e KINGSMAN, 1989).

A utilização de regras heurísticas de atendimento pode parecer simplista, se comparada às regras utilizadas por um programador da produção experiente. Como consequência, é preferível utilizar as heurísticas desenvolvidas pelo programador, a concluir que os modelos produzem melhores resultados (GRANT, 1986). Esse enfoque leva à terceira abordagem, de inteligência artificial, para os problemas de programação da produção.

2.5.3 Aplicações de Inteligência Artificial

Um sistema especialista é um programa computacional inteligente, que utiliza regras heurísticas desenvolvidas por especialistas para resolver problemas do mundo real. Definições alternativas podem ser encontradas em BARR e FEIGENBAUM (1981), WEISS e KUBBOWSKI (1984), e WATERMANN (1986).

A estrutura de sistemas especialistas possui vários componentes, conforme a FIGURA 2.1.

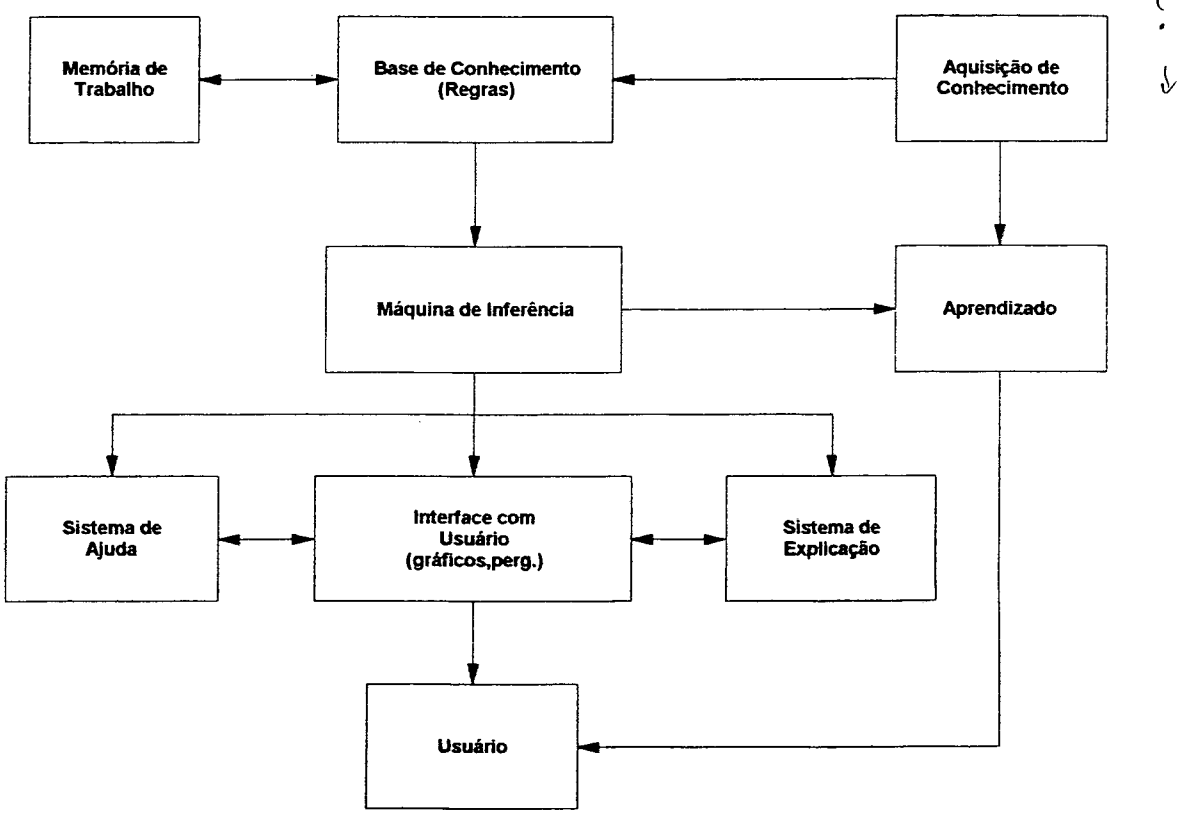


FIGURA 2.1 - Componentes de um Sistema Especialista
Fonte: ADELI (1988)

Os principais componentes a serem considerados na construção de um sistema especialista são: a base de conhecimento, o motor de inferência e a aquisição do conhecimento.

Para representação do conhecimento, existem numerosos formalismos, dentre os quais os mais utilizados são: regras de produção, "frames" e redes semânticas.

Regras de produção consistem em um conjunto de condições e um conjunto de ações. Se todas as condições são verdadeiras, então as ações são executadas. Dado que os especialistas humanos nem sempre têm certeza sobre as regras, medidas ou fatores de certeza podem ser utlizados. "Frames" representam objetos ou conceitos através do conjunto de seus atributos e operadores de manipulação. As redes semânticas enfatizam as relações entre os diferentes objetos e consistem em nós, que representam esses objetos ou conceitos, e arcos, que representam as relações.

O motor de inferência pode ser comparado aos algoritmos dos programas convencionais, exercendo a estratégia de controle na busca de soluções. Segundo KUSIAK e CHEN (1988) três tipos de estratégias de controle são encontradas em sistemas especialistas para planejamento e programação da produção: meta-regras, operadores de busca no espaço de estados e união de padrões para ativar algoritmos ou módulos.

Dadas as incertezas dos ambientes, algumas vezes é necessário prever aquisição dinâmica de conhecimento. Esse conhecimento pode ser adquirido de especialistas, de modelos matemáticos, de programas de simulação e de bases de dados globais.

Vários sistemas especialistas têm sido desenvolvidos para programação da produção. Entre eles apresentam-se os considerados mais importantes dadas suas características inovadoras.

Na Carnegie Mellon University, foi desenvolvido o sistema ISIS (SIMITH, FOX e OW, 1983), para programação da produção em ambientes "job shop" de grande escala. O sistema se baseia no raciocino dirigido por restrições e tem uma estrutura hierárquica em três níveis. Três versões do sistema foram construídas, mas, como obstáculo principal, manteve-se o tempo necessário para a determinação de uma solução.

Após uma análise crítica do sistema ISIS, foi iniciado o desenvolvimento do sistema OPIS. A arquitetura adotada para o sistema OPIS é

o princípio do "quadro-negro", onde diferentes fontes de conhecimento se comunicam através de mensagens.

O ISA (Intelligent Scheduling Assistant) (Orciuch e Frost, 1984) é um sistema especialista que se encontra em nível operacional na Digital Equipment Corporation, resultado de acordo entre a empresa e a Carnegie Mellon University. Esse programa é utilizado, sobretudo, para o planejamento, não entrando em nível de detalhes da programação da produção.

ERSCHLER e ESQUIROL (1986) relatam o desenvolvimento de um sistema denominado MASCOT, que utiliza análise baseada em restrições para programar a produção em ambientes tipo "job shop".

O OPAL, sistema desenvolvido por BENSANA (1986), é similar ao anterior, onde foram incluídas a possibilidade de superposição de lotes e um módulo decisão, para o caso onde a análise de restrições temporais não permite solucionar conflitos. Esse módulo de decisão é composto de regras e, na sua estratégia de controle, é utilizada uma técnica oriunda da Teoria dos Conjuntos Difusos. A cada regra é atribuído um índice de relevância com o objetivo a ser alcançado. Esse índice representa o grau de pertinência com que a regra contribui com o objetivo. Todas as regras relevantes são utilizadas e, através de um processo difuso de votação, é resolvido o conflito.

2.5.4 Aplicações de Algoritmos Genéticos

Os algoritmos genéticos propostos por HOLLAND têm sido muito utilizados para otimização de problemas do tipo combinatorial. Em programação da produção, não são muitas as aplicações existentes desta técnica.

Propondo um operador que combina arcos no problema do caixeiro viajante, WHITLEY, STARKWEATHER e FUQUAY (1989) apresentam uma aplicação para programação da produção em ambientes "job shop". O algoritmo genético é utilizado para programar a primeira máquina como um problema de seqüenciamento similar ao do caixeiro viajante. As máquinas restantes são seqüenciadas utilizando regras simples. O programa de produção é então

avaliado e o valor utilizado como medida de desempenho da seqüência da primeira máquina. Os autores reportam uma aplicação em uma linha de montagem de placas da Hewlett Packard em Fort Collins(USA). Um limite teórico para um problema é calculado, no qual se assume que não existem tempos de espera em nenhuma máquina e considera-se somente um "setup" por máquina. Num experimento com 50000 recombinações, o que implica no mesmo número de avaliações dos programas resultantes, o algoritmo forneceu uma solução a 16,8 % do limite teórico.

A programação de liberação de ordens em um ambiente de produção do tipo "flow shop" utilizando algoritmos genéticos é apresentada por CLEVELAND e SMITH (1989). O ambiente é constituído por uma linha de montagem e pode ser visto como um problema de uma única máquina. É formulado como um problema de seqüenciamento e é utilizada uma codificação semelhante à do caixeiro viajante. Uma pesquisa dos operadores mais utilizados na sua solução e adaptados ao problema de seqüenciamento é realizada.

Para comparar os resultados obtidos, são utilizadas regras de despacho heurísticas para obter soluções. As regras utilizadas são : EDD (Earliest Dues Date first), SPT (Shortest Processing Time first) e LST (Least Slack Time first). Essas regras utilizam respectivamente, a menor data de entrega, a menor duração e a menor folga, para seqüenciar as tarefas.

Um conjunto de quatro problemas foi construído com a finalidade de testar os diferentes operadores. Para cada um desses problemas, foi simulado um setor composto de cinco estações de trabalho, incluíndo três máquinas em cada uma. Em todos os problemas, dez produtos deveriam ser processados. Nos três primeiros não foi admitida qualquer incerteza. O quarto problema foi elaborado da seguinte forma, duas estações realizavam testes que possuíam 15% de probabilidade de fracassar o que implicava em realizar novamente a operação anterior. As demais estações apresentavam comportamento similar aos problemas um, dois e três. Além disso, em cada um dos problemas-teste as condições de carga das máquinas variavam. Os resultados alcançados pelos três melhores algoritmos genéticos (cada um deles, com um operador de recombinação diferente, são apresentados na TABELA 2.3.

Problema	EDD (horas)	SPT (horas)	LST (horas)	Algoritmos Genéticos (horas)
Problema 1	10704	21660	11658	7000-7200
Problema 2	3981	16837	3981	3600-3900
Problema 3	12434	15516	4068	2100-2150

TABELA 2.3 - Comparação das Avaliações das Soluções com as Regras Heurísticas
Fonte: CLEVELAND e SMITH, 1989.

HUSBANDS, MILL e WARRINGTON (1992) utilizam algoritmos genéticos que incluem espécies co-evolutivas para modelar problemas de otimização do planejamento e programação da produção. No modelo proposto pelos autores, várias espécies (planos de produção e programas de produção) evoluem com a pressão da seleção para encontrar um plano ótimo de produção para cada um dos componentes de um produto. Sendo que os resultados alcançados, para dois produtos, foram obtidos num computador com processamento paralelo.

2.6 Programação da Produção em Ambientes "Flow Shop "

Um ambiente "flow shop" constitue-se num "job shop" no qual severas restrições são colocadas (PARK, PEGDEN e ENSCORE, 1984). O seqüenciamento da produção em "flow shop" geral é um problema de programação da produção no qual n produtos ou lotes (jobs) devem ser processados em m máquinas. Além disso, as máquinas podem ser numeradas de forma tal que, para qualquer produto, se o processamento na máquina i precede o processamento na máquina j, então $i < j$.

Os programas são geralmente avaliados de acordo com um único critério que envolve informações sobre todos os produtos. A alta complexidade e as características combinatoriais do problema "têm concentrado as pesquisas na otimização de medidas de desempenho individuais"(DANIELS e CHAMBERS,

1990) e muitas simplificações são feitas nas suposições básicas do problema. As técnicas se concentram em ambientes e critérios específicos, sendo inapropriadas para aplicações práticas, onde algumas das condições não se verificam. Estudando ambientes "flow shop" e preocupados com a minimização do tempo total de processamento, HAN e DEJAX (1991) afirmam que as hipóteses de tempos de "changeover" independentes da seqüência das tarefas nem sempre se verificam. Como exemplo, considerando o problema de programação num ambiente de tecnologia de grupo, no caso de começar o processamento de um conjunto de partes, um longo tempo de preparação é necessário. Por outro lado, pequenos tempos de troca são necessários durante o processamento de uma família de peças. (GUPTA e DARROW, 1986).

As principais hipóteses feitas para o problema são:

- todos os produtos estão disponíveis para processamento no tempo zero;

- a descrição dos produtos é conhecida com antecedência;

- todos os tempos de processamento são conhecidos e algum deles podem ser iguais a zero;

- uma vez que o processamento começa numa máquina, deve ser completado antes que outro produto possa ser processado na mesma máquina;

- existe capacidade suficiente de armazenagem para estocar qualquer número de produtos entre duas máquinas;

- os tempos de preparação das operações são independentes da seqüência e estão incluídos nos tempos de processamento.

Uma das medidas mais utilizadas na literatura como forma de avaliação é o tempo total de processamento (makespan). Outras medidas utilizadas são: tempo de fluxo médio, atraso dos produtos, atraso médio dos produtos, desvio da data de entrega dos produtos etc. (CONWAY 1967).

Existem certos casos onde a consideração de que a ordem de processamento dentro de cada máquina é igual para todos os produtos pode ser

suficiente (permutation "flow shop") (CONWAY, 1967). Mesmo esse problema, onde existem n! soluções uma vez que basta considerar permutações dos produtos, é NP-completo (GAVEY e JOHNSON, 1979). Vários algoritmos heurísticos têm sido propostos para esse problema (DANNENBRING, 1977; PARK, 1984; e TAILLARD, 1990).

TAILLARD (1990) estudou o problema de programação de flow shop considerando somente seqüências de permutações e tempo total de processamento como critério de otimização. Nesse problema, apesar de "fácil se comparado com o problema geral, pode ser solucionado de maneira exata somente para pequenos problemas"(TAILLARD, 1990). O autor obtém todos os possíveis tempos de processamento, por enumeração completa, para 500 problemas com 9 e 10 máquinas, com tempos de processamento aleatoriamente gerados (inteiro entre 1 e 99). O tempo total de processamento relativo à solução ótima parece quase simétrico e seus valores estão contidos em torno de 20% da média. Assim falar do tempo total de processamento médio fornecido por um algoritmo heurístico parece ser uma medida significativa (TAILLARD, 1990). Como critério de comparação da qualidade dos diferentes métodos heurísticos para esses problemas, é utilizado percentual acima do ótimo.

2.7 Conclusões

Verifica-se que os problemas de programação da produção são de difícil solução, devido às suas características combinatoriais. Esse fato explica a inexistência de algoritmos polinomiais para encontrar soluções ótimas, o que torna necessário a busca de outras alternativas.

Entre as abordagens citadas na literatura, para a solução dos problemas de programação da produção, constata-se que as mais utilizadas são aquelas que empregam regras heurísticas. Um dos motivos de tal utilização está relacionado com a facilidade computacional verificada no emprego dessas regras.

Um dos métodos considerados adequados para resolver problemas desse tipo, é aquele no qual são usados algoritmos genéticos, pois esses

apresentam flexibilidade de adaptação a diversos tipos de aplicações, tendo em vista que o conhecimento do domínio encontra-se separado do algoritmo principal.

3. ALGORITMOS GENÉTICOS

3.1 Introdução

Em Biologia, adaptação designa qualquer processo através do qual uma estrutura é progressivamente modificada para obter melhor desempenho no seu ambiente (HOLLAND, 1992). Basicamente, os processos adaptativos são processos de otimização. Algoritmos Genéticos são métodos de busca que se baseiam nos mecanismos da genética natural e da seleção natural. Eles manipulam cadeias de caracteres, combinando a sobrevivência das mais promissoras com a geração aleatória, para formar um algoritmo de busca com características similares ao raciocínio do ser humano.

De acordo com GOLDBERG (1989), em cada geração um novo conjunto de cadeias de caracteres é criado, usando trechos dos melhores da geração anterior, e uma outra parcela é gerada aleatoriamente. Embora gerem alguns elementos de forma aleatória, os algoritmos genéticos não realizam pesquisa aleatória: eles buscam explorar eficientemente a informação histórica, procurando novos pontos para melhorar seu desempenho.

Na natureza, a adequação ao meio ambiente direciona o processo evolutivo. Os indivíduos considerados inadequados e, portanto, fracos, tendem a morrer antes de se reproduzirem, enquanto que os mais fortes, ou seja, bem adaptados, vivem mais, gerando muitos descendentes que, quase sempre, herdam as qualidades dos seus pais.

Inspirado nos processos observados na natureza, John Holland, nos anos setenta, desenvolveu os algoritmos genéticos. Os cromossomas constituem um dos elementos da evolução natural, sendo que os seus processos de codificação não são completamente conhecidos; entretanto, algumas das características da teoria da evolução são completamente aceitas, tais como:

- Evolução é um processo que ocorre nos cromossomas e não nos seres vivos que foram por eles codificados.

- Seleção natural é a ligação entre os cromossomas e o desempenho de suas estruturas decodificadas. O processo de seleção natural propicia a reprodução das melhores estruturas e desestimula a reprodução das outras.

- A evolução ocorre no processo de reprodução. Processos de mutações podem causar diferenças entre os cromossomas dos descendentes e dos pais, e processos de recombinação podem criar cromossomas ligeiramente diferentes, pela combinação dos cromossomas dos pais.

- Evolução biológica não tem memória. Toda evolução parte dos elementos codificados nos cromossomas.

Diferentemente dos algoritmos convencionais que empregam técnicas exaustivas para encontrar o "ótimo local", os algoritmos genéticos utilizam processos aleatórios. Apresentam uma grande flexibilidade de adaptação de suas técnicas a uma extensa variedade de aplicações, uma vez que o conhecimento do domínio encontra-se separado do algoritmo principal. São ainda considerados como um método adequado para solução eficiente de problemas de programação do tipo combinatorial.

3.2 Descrição de Algoritmo Genético

Algoritmo genético relaciona-se com o problema a ser solucionado, principalmente, através de duas estruturas: a codificação do problema e a função de avaliação.

A estrutura de codificação permite representar soluções do problema como cromossomas e a função de avaliação retorna uma medida de adequação de um cromossoma (solução) no contexto do problema.

Muitas são as formas de codificar soluções, variando de problema para problema, e de um algoritmo genético para outro. Historicamente, cadeias de bits têm sido utilizadas a partir dos trabalhos de Holland e grande parte da fundamentação teórica foi desenvolvida sob essa ótica.

Genericamente, um algoritmo genético pode ser descrito pelos seguintes passos (DAVIS, 1991):

1. inicializar uma população de cromossomas;

2. avaliar cada cromossoma na população;

3. criar novos cromossomas pela união dos atuais, aplicando mutações e recombinações a partir do cruzamento dos pais;

4. deletar membros da população, de forma a permitir a inclusão dos novos cromossomas;

5. avaliar os novos cromossomas e inseri-los na população;

6. se alcançado o limite de iterações, parar e retornar o melhor cromossoma; caso contrário, voltar ao passo 3.

No primeiro passo, é inicializada a população de cromossomas através de um processo aleatório.

No passo seguinte, verifica-se o desempenho de cada cromossoma da população inicial, através da função de avaliação.

A seguir, no terceiro passo, novos cromossomas são gerados. Para essa geração, utilizam-se operadores simples, que permitem uma melhoria da população ao longo do tempo. Os operadores mais usuais são: reprodução, cruzamento e mutação. O processo de reprodução permite escolher quais cromossomas irão permanecer na geração de novos cromossomas. Esse operador é uma versão artificial da seleção natural, segundo a Teoria Darwinista, onde sobrevivem os mais aptos. Entre os mais aptos (par a par), é feito o cruzamento, que consiste na troca de partes dos cromossomas entre si. A mutação é executada após o cruzamento. Consiste em uma alteração aleatória de uma parte do cromossoma. As mutações em populações naturais apresentam uma baixa probabilidade de ocorrência.

Nos dois passos seguintes, para que a população permaneça com o mesmo número de cromossomas estabelecidos inicialmente, procede-se à eliminação de alguns membros e à inclusão dos novos, após a sua avaliação.

Finalmente, no sexto e último passo, verifica-se o limite de iterações. Se foi atingido, pára-se o processo e retoma-se o melhor cromossoma; caso contrário volta-se ao passo 3, em busca de melhores cromossomas.

3.3 Definições Básicas

3.3.1 Cromossomas

Cromossomas são extensas cadeias de compostos químicos, consideradas de grande importância nos seres vivos, pois descrevem a completa composição genética dos indivíduos. Em algoritmos genéticos, os cromossomas representam as variáveis de decisão ou de controle de um determinado problema. Eles são codificados sob a forma de cadeia de caracteres, sendo que a alteração de qualquer um desses caracteres muda a informação. Pode-se exemplificar um cromossoma como uma cadeia de bits, onde a presença de "um" ou "zero" numa determinada posição significa a existência ou não de uma determinada característica.

3.3.2 Codificação Binária

Neste tipo de codificação, a cadeia de caracteres é composta somente de caracteres "0" e "1". Um exemplo de Codificação na otimização de uma função com duas variáveis (x e y) (DAVIS, 1991) é:

Cromossoma:

000010 100001100000000110001010100011101111011

é particionado em:

00001010000110000000001 e 10001010100011101111011

Essas cadeias de "bits" são convertidas da base 2 para a base 10 no campo x_1 e y_1:

165377 e 2270139

3.3.3 Codificação Baseada em Ordem

Para alguns problemas reais, não existe uma maneira prática de codificação através de cadeias de caracteres binários, de modo a representar uma relação de ordem. Entre esses, cita-se o problema do caixeiro viajante e o de coloração de grafos.

No problema do caixeiro viajante, deseja-se ordenar cidades de forma a minimizar o percurso, sem que haja repetições das cidades já visitadas. Uma codificação natural para os cromossomas, consiste em representar as soluções como uma lista de cidades, onde a ordem indica a seqüência em que as mesmas serão visitadas. Como WHITLEY et al (1989) observam, os próprios nomes das cidades são manipulados pelo algoritmo, ao invés de cadeias de caracteres. Por exemplo: dadas as cidades A, B, C, D e E, soluções viáveis do problema do caixeiro viajante podem ser representadas pelos seguintes cromossomas:

Cromossoma 1: B A C D E
Cromossoma 2: A C B E D

A codificação baseada em ordem, que surge de forma natural, cria a necessidade de definir operadores específicos de cruzamento e mutação.

3.3.4 População

População é um conjunto de cromossomas, geralmente inicializada de forma aleatória. As populações subseqüentes são obtidas pela aplicação de operadores.

O tamanho da população é um fator importante no desempenho dos algoritmos. Como os cromossomas pertencentes às novas populações são gerados pela combinação dos cromossomas pertencentes às populações anteriores, e sua quantidade não é alterada, o estabelecimento de um tamanho muito pequeno para a população inicial pode ocasionar ciclos. Por outro lado, um tamanho muito grande irá exigir um maior número de avaliações, comprometendo a eficiência do método.

3.3.5 Avaliação

A avaliação permite quantificar a qualidade dos cromossomas ou soluções do problema. Para tal fim, é utilizada uma função de avaliação, estabelecida de acordo com o problema a ser resolvido. Essa função mapeia cada cadeia de caracteres em um número real, sendo que os valores obtidos serão utilizados para a escolha das cadeias que permanecerão na população e serão responsáveis pelos descendentes dessa população. Assim, aqueles que tiverem valores altos serão os responsáveis pela formação da população futura, e aqueles com valores baixos tendem a desaparecer e não contribuem com a nova geração. Entretanto, em muitas aplicações práticas, verifica-se a necessidade de uma transformação da função de avaliação, através de uma função de aptidão, de forma a permitir que mesmo os elementos com valores baixos possam, também, contribuir na reprodução, evitando com isso a dominância dos elementos com valores altos. Para um melhor controle do número de descendentes, várias técnicas de aptidão podem ser utilizadas. Entre elas, pode-se citar: a transformação de avaliação de custos para aptidão, mudança de escala na aptidão e normalização linear.

3.3.5.1 Transformação de Avaliação de Custos para Aptidão

Transformação de avaliação de custos para aptidão é uma técnica utilizada em algoritmos genéticos, quando a função de avaliação é, naturalmente, estabelecida como a minimização de alguma função de custo, ao invés da maximização de alguma função de lucro. A dualidade da minimização de custo (g(x)) e a maximização de lucro é bem conhecida, e em algoritmo genético essa transformação é da seguinte forma:

$$f(x) = \begin{cases} C_{máx} - g(x), \text{ onde } g(x) < C_{máx} \\ 0, \text{ caso contrário} \end{cases} \qquad (3.1)$$

O coeficiente $C_{máx}$ pode ser escolhido de diversas maneiras. Pode-se tomar como um coeficiente de entrada; como o maior valor observado "g"; como o maior valor "g" da população atual; ou ainda, como o maior valor "g" das últimas "k" gerações.

3.3.5.2 Normalização Linear

Normalização linear é uma técnica utilizada para encontrar o mínimo valor de avaliação na população, e designar a cada cromossoma um valor de adaptação igual à diferença entre esse valor e sua avaliação.

3.3.5.3 Mudança de Escala na Aptidão

Mudança de escala na aptidão é um processo utilizado devido à extrema importância do controle do número de descendentes em algoritmos genéticos com populações pequenas. Deve-se evitar que cromossomas altamente aptos se reproduzam excessivamente, tornando-se dominantes e conduzindo a uma convergência prematura. Por outro lado, após muitas

iterações, pode ocorrer que a aptidão de todos os cromossomas, apesar da sua diversidade, seja muito próxima da aptidão do melhor cromossoma. Nesse caso, a escolha dos sobreviventes não segue o princípio da seleção natural, tornando-se, praticamente, uma escolha aleatória. Uma mudança linear de escala na aptidão minimiza essas tendências. A aptidão transformada é obtida da seguinte forma:

$$f' = af + b \qquad (3.2)$$

onde: f' = aptidão transformada
f = aptidão
a e b = constantes

As constantes a e b podem ser escolhidas de diferentes formas, mas, em todos os casos, a transformação deve manter a aptidão média constante. Para controlar o número de descendentes do melhor cromossoma, escolhem-se as constantes de forma que $f'_{máx} = k\, f_{médio}$, onde "k" é o número esperado de descendentes do melhor cromossoma.

3.3.6 Seleção de Pais

A seleção de pais tem por objetivo proporcionar maiores chances de reprodução aos indivíduos ou cromossomas que têm uma função de adaptação melhor.

Uma técnica comumente utilizada é a seleção através de roleta, na qual cada indivíduo na população tem um número de posições proporcionais à sua adaptação.

3.3.7 Esquema

Um esquema é um padrão de similaridade, constituído por um pequeno subconjunto de posições em um cromossoma, que, tomando valores específicos, atuam como uma unidade, influenciando na avaliação.

Para exemplificar, considere uma codificação binária, à qual é adicionado um caracter "*". Também, considere cadeias com quatro caracteres e o esquema *.010, onde o símbolo "*" significa "qualquer valor". Esse esquema combina com as cadeias 1010 e 0010, uma vez que, em cada posição, o "zero" casa com o "zero" e o "um" com o "um", e o "*" com qualquer um deles.

A noção de esquema permite referir-se às similaridades de cromossomas, de forma compacta.

Os esquemas e sua contribuição para a função objetivo propagam-se através das gerações, proporcionando amostras das melhores contribuições que crescem exponencialmente.

3.3.8 Reprodução

Reprodução é a substituição dos cromossomas de uma população velha, gerando uma nova população. Entre as técnicas de reprodução, pode-se citar: reprodução tradicional, reprodução com elitismo, reprodução em estado estacionário e reprodução em estado estacionário sem duplicação.

3.3.8.1 Reprodução Tradicional

Reprodução tradicional é o processo de substituição de todos os indivíduos da população velha pelos seus descendentes, para formar a nova população.

3.3.8.2 Reprodução com Elitismo

Reprodução com elitismo é uma técnica que privilegia os melhores indivíduos. Na escolha dos indivíduos da população que gerarão descendentes, os mais adaptados podem não conseguir se reproduzir. Quando se utiliza a técnica da reprodução por elitismo, os melhores indivíduos são diretamente mantidos na nova população.

3.3.8.3 Reprodução em Estado Estacionário

Reprodução em estado estacionário é uma técnica através da qual, um ou dois indivíduos, a cada iteração, são substituídos, avaliando os descendentes e inserindo-os na nova população. Segundo DAVIS (1989), essa técnica foi desenvolvida para solucionar os problemas de outras técnicas, como, por exemplo, a reprodução tradicional ou reprodução com elitismo, que podem ocasionar a perda de muitos dos bons indivíduos que não se reproduzem. Além disso, os operadores de cruzamento e mutação podem destruir e/ou alterar os esquemas dos melhores indivíduos, fazendo com que as características boas dos mesmos sejam destruídas. O mesmo autor salienta que alguns pesquisadores experimentaram esta técnica e acharam-na inferior à reprodução tradicional.

3.3.8.4 Reprodução em Estado Estacionário sem Duplicação

Reprodução em estado estacionário sem duplicação é uma técnica similar à reprodução em estado estacionário, porém a substituição somente é realizada se não existir um indivíduo igual ao descendente na população.

3.3.9 Cruzamento

No cruzamento, inicialmente, são escolhidos os pais. Uma vez selecionados os pais, o par de cromossomas passa por um processo de cruzamento, dessa forma gerando dois novos cromossomas.

Vários são os tipos de cruzamento possíveis. Entre os mais relevantes, pode-se citar: cruzamento com um ponto, cruzamento com dois pontos, cruzamento uniforme, cruzamento uniforme para representações baseadas em ordem, operador de Goidberg, operador de Grefentestte e substituição de sub-rotas.

3.3.9.1 Cruzamento com Um Ponto

Neste tipo de cruzamento, uma posição "k" no cromossoma é selecionada aleatoriamente, com probabilidade uniforme. No intervalo [1,c-1], onde "c" é o comprimento do cromossoma, os dois descendentes são criados pelo intercâmbio das cadeias de caracteres da posição "k" até a posicão "c". Uma característica importante desse operador é que, se os dois pais possuem os mesmos caracteres, em determinadas posições, o cruzamento não altera essas posições.

O cruzamento com um ponto foi inspirado nos processos biológicos naturais; entretanto, esse operador é incapaz de combinar certas estruturas codificadas nos cromossomas.

3.3.9.2 Cruzamento com Dois Pontos

Este tipo de cruzamento é utilizado para resolver algumas deficiências do cruzamento com um ponto. Por exemplo, dados os cromossomas:

Cromossoma 1: **1 0 1**|1 0 1|**11 0 0**
Cromossoma 2: 0 1 0|**1 1 0**|11 1 0

As informações codificadas nos caracteres em negrito não podem ser combinadas ou repassadas para um único cromossoma, com o cruzamento com um ponto. Entretanto, no cruzamento utilizando dois pontos, isso é possível. Nesse operador, a seleção dos pontos para cruzamento é feita de maneira idêntica à realizada no cruzamento com um ponto, e as cadeias de caracteres entre os dois pontos são trocadas entre si.

Utilizando as posições marcadas como ponto de cruzamento, obtêm-se os seguintes cromossomas:

Cromossoma 1: 1 0 1|1 1 0|11 0 0
Cromossoma 2: 0 1 0|1 0 1|11 1 0

3.3.9.3 Cruzamento Uniforme

Para cada posição, o operador de cruzamento uniforme decide aleatoriamente que pai contribui com seu valor para qual descendente. O outro descendente recebe o valor do outro pai.

3.3.9.4 Cruzamento Uniforme para Representações Baseadas em Ordem

DAVIS (1991) relata a obtenção de bons resultados com este operador. Da mesma forma que o operador de cruzamento uniforme, este operador preserva parte de um dos pais, enquanto incorpora informações do outro. A informação codificada, entretanto, não é um valor fixo associado com uma posição no cromossoma, e sim, uma ordenação relativa de elementos no cromossoma. Ou seja, a informação não é da forma "nó 3 está na quinta posição", mas da forma "nó 3 está depois do nó 2 e antes do nó 7".

3.3.9.5 Operador de Goldberg - PMX

Este operador é diretamente aplicável em problemas de formulação seqüencial. O operador PMX de Goldberg implementa um operador de recombinação, que garante a legalidade das soluções produzidas, escolhendo um intervalo em cada solução, trocando os intervalos e, então, mapeando o intervalo selecionado dentro de cada solução.

3.3.9.6 Operador de Grefentestte

Este operador não tem conhecimento do domínio por si, mas faz bom uso da estrutura de representação. É motivado pelo desejo de manter a localização relativa de partes que são mudadas durante a recombinação.

O operador trabalha selecionando parte dos pais e tentando colocar essas partes no descendente nas mesmas posições que são ocupadas nos pais. Os conflitos que aparecem são resolvidos pela condensação das partes e pelo deslocamento à direita ou à esquerda, de tal forma que ocorra um ajuste.

3.3.9.7 Substituição de Sub-Rotas

Este operador examina as duas soluções para cruzar por similaridade as subseqüências localizadas que possuem os mesmos elementos. Quando duas subseqüências são encontradas nas respectivas soluções, elas são substituídas uma pela outra. Infelizmente, é baixa a frequência de encontrar tais subseqüências e deveria se esperar a fase de seleção do algoritmo genético para escapar da fase de cruzamento, se este é o único operador de recombinação utilizado.

O operador de substituição de sub-rotas não causa distúrbio para o restante da solução.

3.3.10 Mutação

A mutação permite incluir diversidade na população. O processo de mutação consiste na alteração das informações codificadas através da mudança aleatória de caracteres do cromossoma.

Na codificação binária dos cromossomas, percorre-se a cadeia de caracteres, determinando a probabilidade de mudança. Se essa probabilidade atingir o valor desejado, o "bit" é substituído por um "bit" selecionado aleatoriamente.

3.4 Métodos Tradicionais e Algoritmos Genéticos

Na literatura, identificam-se três tipos principais de métodos tradicionais de busca, quais sejam: baseados em cálculo, enumerativos e aleatórios.

Os métodos baseados em cálculo podem ser encontrados em duas categorias: diretos e indiretos. Os métodos diretos procuram os máximos locais, pulando de ponto em ponto e movendo-se na direção de maior gradiente da função, enquanto os métodos indiretos utilizam sistemas de equações não lineares, através da derivação das funções e igualando-as a zero. Esses métodos não são considerados suficientemente robustos e, por isso, são métodos locais, só podendo ser usados na vizinhança dos pontos de máximo.

Os métodos enumerativos determinam a função em cada ponto, comparando-a com o valor precedente e mantendo-se sempre o maior valor, quando se deseja determinar o seu máximo. Dependendo da dimensão do problema a ser solucionado, tornam-se impraticáveis devido à grande quantidade de pontos em que a função deve ser determinada, não sendo, então, considerados robustos.

Os métodos de busca aleatória procuram randomicamente o ponto ótimo da função, dentro do espaço de soluções. Entretanto, a busca aleatória

exaustiva pode não conduzir ao ponto ótimo. Se comparados com o mesmo número de iterações necessárias nos métodos enumerativos, os métodos aleatórios são considerados robustos.

Deve-se ter o cuidado de distinguir os métodos de busca aleatória dos métodos que utilizam técnicas aleatórias. Por exemplo, os algoritmos genéticos são considerados procedimentos de busca que usam escolhas aleatórias como uma ferramenta para direcionar uma busca altamente explorativa, através de uma codificação de um espaço de parâmetros. Nesses algoritmos, novas gerações são criadas usando partes das antigas mais aptas, mantendo um compromisso entre eficiência e eficácia, para dar apoio ao princípio da sobrevivência dos descendentes mais aptos, apresentando, como ponto central, a robustez.

Para GOLDBERG (1989), os estudos e trabalhos desenvolvidos com sistemas biológicos comprovam sua alta eficiência, flexibilidade e robustez, apresentando qualidades como as de auto-corrigir-se, auto-orientar-se e auto-reproduzir-se.

Para um melhor entendimento da consideração de robustez dos algoritmos genéticos, deve-se verificar os aspectos diferenciais destes em relação aos métodos tradicionais de otimização. Esses aspectos, de uma forma geral, são:

- Algoritmos Genéticos trabalham com uma codificação do conjunto de parâmetros, não com os próprios parâmetros.

- Algoritmos Genéticos buscam solução a partir de uma população de pontos, não com um único ponto;

- Algoritmos Genéticos usam informações obtidas pela avaliação da função objetivo, não com derivadas ou outros conhecimentos auxiliares;

- Algoritmos Genéticos usam regras probabilísticas, não regras determinísticas.

3.5 Fundamentos Matemáticos

Um conceito fundamental em algoritmo genético é o relativo a um esquema ou padrão de similaridade.

A avaliação quantitativa do número de esquemas ou de padrões de similaridade pode ser efetuada, respeitando-se as seguintes etapas:

- Contagem de esquemas representados dentro de uma população de cadeia de carecteres;

- Considerações sobre as taxas de crescimento relativamente ao número de esquemas na população considerada;

- Estabelecimento de uma formulação matemática para os crescimentos verificados.

Uma questão fundamental é a de estabelecer um relacionamento entre a formulação matemática que descreve o processo de enumeração dos esquemas, suas taxas de crescimento e a obtenção de respostas a problemas arbitrários. Em outras palavras, como se pode avaliar as respostas obtidas pela combinação "arbitrária" de blocos de similaridade.

O teorema fundamental de algoritmos genéticos é: a operação em um algoritmo genético é, em termos simples, uma combinação direta entre cadeias de caracteres que melhor representem as propriedades de uma população. Através da troca de blocos de esquemas, objetiva-se a construção de novas populações, nas quais, por intermédio de mecanismos de cruzamento, reprodução e mutação, as características desejadas vão sendo sucessivamente melhor representadas.

Para analisar os mecanismos relativos ao processo de reprodução de uma população, considerem-se as seguintes definições:

Cadeia de Caracteres - é uma coleção de objetos binários do tipo (0,1), denotando-se um indivíduo através de letras latinas maiúsculas, e o i-ésimo elemento de uma cadeia de carecteres por letras minúsculas, subscritas por sua posição.

$$A = 0111000 \text{ ou } A = a_1a_2a_3a_4a_5a_6a_7$$

Gens - são formados pelos elementos a_i das cadeias de carcteres. Representam a presença ou não de determinada característica.

Além dessas definições, deve-se considerar o seguinte:

- O número de esquemas para um alfabeto de cardinalidade k é dado por $(k+1)^l$, onde l é o comprimento da cadeia de caracteres;

- O número de esquemas representativos para uma cadeia de caracteres de comprimento l é 2^l;

- O número máximo de esquemas representativos em uma população de tamanho n é $n \cdot 2^l - (x-1)$.

As propriedades de um esquema consideradas importantes são:

- Ordem (0(H)): número de posições fixadas em um gabarito.

$$1***1** \rightarrow 0(H) = 2$$
$$1**1*** \rightarrow 0(H) = 2$$

- Comprimento de definição δ (H): distância entre a primeira e a última posição especificadas para uma cadeia de carcteres.

$$1***1** \rightarrow \delta(H) = 5 - 1 = 4$$
$$1**1*** \rightarrow \delta(H) = 4 - 1 = 3$$

Essas propriedades são úteis no processo de discussão e de classificação de similaridades na análise de conjuntos de cadeias de carcteres. Através delas, é possível avaliar os efeitos da reprodução, do cruzamento, da mutação e do uso de operadores genéticos em blocos pré-construídos, dentro de uma população.

Seja m = m(H,t), a representação de um exemplo de um esquema particular H, contido em uma população A(t), no tempo t. Durante o

processo reprodutivo, uma cadeia de caracteres é copiada de acordo com sua aptidão, ou mais especificamente, uma cadeia de caracteres A_i é selecionada com probabilidade:

$$p_i = \frac{f_i}{\sum_j f_j} \qquad (3.3)$$

onde: f_i = valor da função de avaliação para a cadeia de caracteres i

Após um processo de geração de uma nova população, o número esperado de indivíduos representativos de um esquema H, no tempo (t+1), é dado por:

$$m(H,t+1) = \frac{f(H)}{\hat{f}} \cdot m(H,t) \qquad (3.4)$$

onde: f(H) = aptidão média das cadeiaa de caracteres representativas do esquema H, no tempo t

$\hat{f}$ = aptidão média das cadeias de caracteres existentes na população, no tempo t

Assumindo que um esquema apresenta uma aptidão acima da média em $c\hat{f}$ unidades, então, a equação (3.4) pode ser reescrita:

$$m(H,t+1) = m(H,t) \cdot \frac{(\hat{f}+c\hat{f})}{\hat{f}} \qquad (3.5)$$

$$m(H,t+1) = (1+c)m(H,t) \qquad (3.6)$$

Partindo de t = 0, e supondo c um valor constante, tem-se:

$$m(H,1) = (1+c)m(H,0) \qquad (3.7)$$

$$m(H,2) = (1+c)m(H,1) = (1+c)^2 m(H,0) \qquad (3.8)$$

$$\vdots$$

$$m(H,t) = (1+c)^t m(H,0) \qquad (3.9)$$

Assim, um esquema acima da média deverá crescer exponencialmente com o tempo (em contrapartida, aqueles que se situam abaixo

da média tenderão a desaparecer). O valor de c é a taxa de crescimento para o esquema considerado.

O cruzamento representa, na realidade, uma troca de informações aleatorizada entre cadeias de caracteres. O efeito do cruzamento é o de, fundamentalmente, proporcionar a criação de novas estruturas, sem que ocorra uma ruptura na estratégia de alocação fornecida pela reprodução. Paralelamente, o cruzamento tem o efeito de aumentar exponencialmente a proporção de alguns esquemas verificados em uma dada população. Como exemplo, considere o seguinte:

$$A = 0111000$$
$$H_1 = *1****0**$$
$$H_2 = ***10**$$

H_1 e H_2 estão representados na cadeia de caracteres A. Considerando um cruzamento efetuado entre a terceira e a quarta posição, o esquema H_2 vai sobreviver. Entretanto, nada se pode afirmar sobre o esquema H_1. Para uma cadeia de caracteres de tamanho L, existem L -1 pontos de cortes possíveis. A probabilidade (p_d) de um esquema H_i ser destruído é:

$$p_d = \frac{\delta(H_i)}{L-1} \qquad (3.10)$$

$$\overline{p}_d = 1 - p_d = 1 - \frac{\delta(H_i)}{L-1} \qquad (3.11)$$

Considere-se, agora, a probabilidade de um esquema sobreviver. Essa probabilidade poderá ser obtida a partir de análises realizadas sobre um processo de destruição de um esquema: Se o cruzamento é realizado através de processos aleatórios, então, p_c é a probabilidade de um esquema ser escolhido para um dado cruzamento. A probabilidade de um esquema escolhido ser destruído é conhecida e menor ou igual a $\frac{\delta(H_i)}{L-1}$. Logo, a probabilidade de o esquema ser escolhido e destruído será menor ou igual a $p_c \cdot \frac{\delta(H_i)}{L-1}$, donde, obtem-se a probabilidade de sobrevivência para um esquema:

$$p_s \geq 1 - p_c \cdot \frac{\delta(H_i)}{L-1} \qquad (3.12)$$

O número esperado de elementos de um esquema particular na próxima geração, considerando o efeito combinado de reprodução e de cruzamento, assumindo independência entre as operações de reprodução e cruzamento, é dado por:

$$m(H,t+1) \geq m(H,t) \cdot \frac{f(H)}{\hat{f}} \cdot \left[1 - p_c \frac{\delta(H_i)}{L-1}\right] \qquad (3.13)$$

Considera-se mutação como sendo uma alteração aleatória de uma única posição e admite-se que a sua ocorrência se dá com probabilidade p_m. Assim, para uma única posição, a probabilidade de sobrevivência é igual a $1-p_m$. Se um dado esquema sobrevive, isso implica em que cada uma de suas posições fixas (O(H)), também sobrevive. Então, para um esquema com O(H) = n, obtém-se a seguinte probabilidade de sobrevivência:

$$p_s' = (1-p_m) \cdots (1-p_m) = (1-p_m)^n = (1-p_m)^{O(H)} \qquad (3.14)$$

Expandindo-se o termo $(1 - p_m)^{O(H)}$ através de coeficientes binomiais para valores de p_m pequenos, pode-se desprezar os termos de grau superior a 1, e obtém-se a seguinte aproximação:

$$p_s' = 1 - O(H)p_m \qquad (3.15)$$

Assim, combinando-se os efeitos da reprodução, cruzamento e mutação, o valor esperado de cópias que um dado esquema H irá receber na próxima geração é dado por:

$$m(H,t+1) \geq m(H,t) \cdot \frac{f(H)}{\hat{f}} \cdot \left[1 - p_c \frac{\delta(H)}{L-1}\right] \cdot \left[1 - O(H)p_m\right] \qquad (3.16)$$

$$m(H,t+1) \geq m(H,t) \cdot \frac{f(H)}{\hat{f}} \cdot \left[1 - O(H)p_m - p_c \frac{\delta(H)}{L-1} + p_c \frac{\delta(H)}{L-1} \cdot O(H)p_m\right] \qquad (3.17)$$

Ignorando os produtos cruzados (probabilidade desprezível de ocorrência) vem:

$$m(H,t+1) \geq m(H,t) \cdot \frac{f(H)}{\hat{f}} \cdot \left[1 - p_c \frac{\delta(H)}{L-1} - O(H)p_m\right] \qquad (3.18)$$

Se O(H), a ordem do esquema for pequena, então o produto $O(H)\cdot p_m$ também o será. Adicionalmente, considerando para um esquema uma probabilidade pequena de ser escolhido e destruído, o efeito conjugado referente à sua presença em geração futura depende apenas de $\frac{f(H)}{\hat{f}}$. Ou seja, para esquemas de alto grau de ajuste, comprimento pequeno e baixa ordem, o seu número esperado para gerações subseqüentes cresce exponencialmente. Tal resultado é conhecido como sendo o Teorema Fundamental de Algoritmos Genéticos.

A alocação exponencial de experimentos é bastante análoga ao problema bem conhecido da Teoria Estatística de Decisão - problema da máquina com duas alavancas. Para esse problema, HOLLAND (1992) apresenta a seguinte estratégia de alocação máxima:

$$n^* \approx \frac{1}{2r} \ln\left[\left(r^3 c^2 N^2\right) / \pi \ln\left(r^2 c^2 N^2 / 2\pi\right)\right] \qquad (3.19)$$

onde: $c = \inf_t(m(t))$, com m(t) = diferença entre as funções geradoras de momentos correspondentes a cada uma das alavancas

$r = |\ln(c)|$

N = número total de experimentos

De acordo com essa estratégia, a alocação dos experimentos subseqüentes, para a alavanca com melhor média observada, cresce exponencialmente com o aumento do número de experimentos. Para a generalização do problema da máquina com k alavancas, HOLLAND (1973) observou que a alocação crescia exponencialmente para o grupo de alavancas com melhor desempenho médio observado. Um raciocínio similar pode ser efetuado para um grupo de esquemas que competem entre si.

Para avaliar a aplicabilidade do raciocínio discutido anteriormente, considerem-se dois esquemas A e B, com posições individuais a_i, b_i. Esses esquemas competem entre si se, em todas as posições i, i=1,...,l, ou $a_i=b_i=*$ ou $a_i \neq *$, $b_i \neq *$, $a_i \neq b_i$ para pelo menos um i.

Para uma população de cadeia de caracteres de comprimento l e tamanho n, pode-se obter entre 2^l e $n2^l$ esquemas para processamento. Todavia,

nem todos os esquemas são processados. Os cruzamentos realizados acabam por destruir os esquemas de maior tamanho.

A estimativa mais amplamente utilizada para o número de esquemas processados é $O(n^3)$, ou seja, é proporcional ao cubo do tamanho da população. Esse resultado é denominado por Holland (GOLDBERG, 1989) como paralelismo implícito e mostra que, apesar da destruição de esquemas de mais alta ordem, através de cruzamento e mutação, os algoritmos genéticos processam uma grande quantidade de esquemas, embora trabalhem com uma quantidade relativamente pequena de cadeias de caracteres.

O quadro do desempenho do algoritmo genético fica muito mais claro com a perspectiva fornecida pelos esquemas. Esquemas curtos, de baixa ordem e de alta aptidão são amostrados, reconhecidos e reamostrados para formar cadeias de caracteres de ajustes potencialmente mais elevados. Esse procedimento torna a busca de uma solução ótima mais eficiente, diminuindo a quantidade de processamento envolvido.

3.6 Conclusão

Algoritmos genéticos são métodos que manipulam um conjunto de soluções na busca de boa solução para um determinado problema. Nesse processo, a única informação significativa é o valor da função objetivo, que é empregado como forma de avaliação para detectar quão perto se está de uma melhor solução, utilizando a experiência adquirida nas tentativas de solução anteriores.

Estes algoritmos possuem um paralelismo implícito que lhes confere um grande poder de processamento, onde muitas similaridades são processadas em paralelo, numa única geração, ainda que poucas estruturas sejam manipuladas.

Na solução de muitos dos problemas constata-se, naturalmente, a adequação de algoritmos com alto paralelismo. Porém, as abordagens tradicionais tendem a obscurecer esse paralelismo inerente, com o uso de

caracterizações matemáticas, enquanto que o uso de algoritmos genéticos leva a caracterizações mais gerais que exploram esse paralelismo, permitindo melhores soluções.

4. UM ALGORITMO GENÉTICO PARA PROGRAMAÇÃO DA PRODUÇÃO

4.1 Introdução

Como visto anteriormente os problemas de Programação da Produção são considerados de difícil solução uma vez que apresentam uma forte complexidade computacional e usualmente envolvem algoritmos específicos para cada domínio.

Uma técnica atrativa para resolver eficientemente esses problemas do tipo combinatoriais baseia-se na utilização de algoritmos genéticos. Sua principal vantagem recai na habilidade de escapar de ótimos locais (LAWTON, 1982).

As propriedades dos algoritmos genéticos, teoricamente provadas, sugerem a sua eficiência em resolver problemas combinatoriais desses tipos (CLEVELAND e SMITH, 1989). Entretanto, as dificuldades associadas em formular esses problemas nos limites das restrições de representação e dos operadores desse método de busca, tem levado os pesquisadores a investigar representações e operadores alternativos.

Neste contexto, apresenta-se um algoritmo genético genérico para solução de problemas do tipo "flow-shop" baseado em representação e operadores alternativos.

4.2 Adaptação de Algoritmos Genéticos a Problemas Específicos

Para a resolução de um determinado problema, quatro componentes devem ser projetados e/ou adaptados quando utiliza-se um algoritmo genético (SYWERDA in DAVIS, 1991).

Os componentes envolvidos são: sintaxe do cromossoma, interpretação do cromossoma, avaliação do cromossoma e operadores a serem aplicados nos cromossomas.

Além desses componentes, outros parâmetros devem ser estipulados, tais como: método de seleção dos pais, percentual de elitismo, etc, mas esses, geralmente, são pouco dependentes da aplicação.

Na maioria das vezes, os cromossomas são constituídos por um vetor binário, representação esta que serve como base para a fundamentação teórica dos algoritmos genéticos. Porém, torna-se muito complicado, representar dessa forma problemas mais complexos.

No problema de Programação da Produção, a primeira representação que surge é designar a cada tarefa um número binário que representa a sua ordem de execução. Assim, a solução do problema poderia ser representada num cromossoma, como a concatenação desses números binários. Todavia, os operadores simples como mutação e cruzamento podem facilmente gerar cromossomas, onde mais de uma tarefa apareça em uma determinada ordem ou ordens onde não existam tarefas.

Uma outra abordagem seria designar a cada tarefa um número ou rótulo e construir os cromossomas como permutações de todas as tarefas. Tendo o cuidado de satisfazer as restrições de precedência e nesse caso os operadores, também, devem satisfazer essas restrições, o que os torna muito complexos e computacionalmente ineficientes.

Representações semelhantes a essas, para outros problemas de ordenação, como o problema do caixeiro viajante e o problema de coloração de grafos já foram utilizados (DAVIS, 1991).

Nessas representações, como afirma SYWERDA in DAVIS (1991), uma objeção importante que pode ser feita é de que tal abordagem carece de fundamentação teórica. Mas, a experiência tem mostrado que outras representações e outros operadores trabalham da mesma forma que os vetores binários.

DAVIS (1991) sugere três princípios para adaptar a técnica de algoritmo genético com outros algoritmos em uso, para a solução de problemas. Esses princípios são: usar a codificação atual , hibridizar onde for possível e adaptar os operadores genéticos.

Usando-se a codificação atual, a experiência acumulada, nela embutida, é preservada. Além disso, garante-se que o algoritmo genético parecerá mais natural ao usuário.

Com referência à hibridação, se existem algoritmos rápidos a solução ou soluções podem ser incorporadas à população inicial. Um algoritmo com elitismo aplicado à essa população garante que uma solução tão boa quanto a dos algoritmos existentes será encontrada, levando provavelmente a melhorias.

4.3 Definição do Problema

Um problema de programação da produção, no qual n produtos devem ser processados em m máquinas é dito ser "flow shop" se : "as máquinas são numeradas de tal forma que, para cada produto considerado, a operação k é realizada numa máquina com número maior que a máquina da operação j, se j precede k"(CONWAY, MAXWELL e MILLER, 1967).

O problema a ser tratado neste capítulo consiste de um problema de "flow shop" geral, no sentido definido por WIDLLER e HERTZ (1989), no qual algumas das operações ou tarefas podem apresentar tempos de processamento nulos em uma ou mais máquinas.

As seguintes premissas são consideradas:

- Um conjunto de n produtos encontra-se disponível para processamento no instante de tempo zero;

- Cada produto necessita da realização de até m tarefas, sendo que essas tarefas requerem processamentos em máquinas diferentes;

• Os tempos de processamento do produto i na máquina j é t_{ij} (i= 1,....,n; j=1,....,m);

• Os tempos t_{ij} (i= 1,....,n; j=1,....,m) são conhecidos "a priori", portanto bem determinados e finitos;

• A ordem de processamento dos produtos não interfere no tempo de processamento requerido pelos mesmos;

• Os tempos t_{ij} (i= 1,....,n; j=1,....,m) abragem os tempos de preparação, processamento e deslocamento;

• m diferentes máquinas estão disponíveis, de forma contínua, a partir do instante de tempo zero;

• Cada tarefa é realizada uma única vez em cada uma das máquinas disponíveis;

• Uma tarefa não se torna disponível para uma máquina enquanto a tarefa predecessora na máquina não estiver completamente concluída;

• Uma tarefa não se torna disponível para uma máquina enquanto a tarefa predecessora, na rota de produção do produto, não tiver sido concluída;

• Não há limitações de estocagem entre máquinas sucessivas.

A partir das considerações feitas anteriormente, as quais definem o ambiente de produção, pode-se estudar:

• Problema de obtenção de uma solução que minimize o tempo total de processamento;

• Problemas de obtenção de uma solução que minimize o tempo de atraso médio;

• Problema de obtenção de uma solução que minimize o estoque de produtos em processamento.

O problema aqui tratado consiste, da adequação da técnica de algoritmo genético para a solução dos problemas enunciados.

4.4 Codificação

O problema de programação da produção em ambientes tipo "flow-shop", como visto anteriormente, pode ser descrito como um problema de ordem, ou seja, é necessário encontrar uma ordem para as tarefas serem executadas. Essa ordem deve respeitar as relações de precedências existentes entre as tarefas de um mesmo produto.

Um fator importante na escolha da forma de codificação das soluções, é que todas as possíveis soluções tenham uma representação e que essa seja única. É, também, desejável que cada codificação corresponda a uma solução viável. Além disso, a estrutura dos operadores será muito simplificada se o espaço de codificação e o espaço de soluções tiverem uma correspondência 1:1.

Uma estrutura que, naturalmente, surge para os cromossomas é a codificação para problemas de ordem, onde as soluções são representadas como uma lista de tarefas, onde a ordem indica a seqüência em que as mesmas serão realizadas.

Essa codificação permite gerar cromossomas que por não atender as relações de precedências entre as tarefas do mesmo produto correspondem a soluções inviáveis, e nesse caso, para cada solução representada pelo cromossoma, teria que ser verificada sua viabilidade. Dessa forma, comprometendo a eficiência do algoritmo.

No problema em questão, uma solução viável é aquela em que:

- As relações de precedência entre as tarefas são satisfeitas;
- Cada tarefa aparece uma, e somente uma vez na solução.

Além disso, as máquinas podem ser ordenadas de tal forma que para um determinado produto, suas tarefas sejam executadas nessa mesma ordem. Assim, todas as tarefas a serem executadas em uma determinada máquina não possuem relação de precedência entre si.

A codificação utilizada agrupa as tarefas em cada uma das máquinas, eliminando a necessidade de se considerar as relações de precedência entre as tarefas.

O cromossoma é representado por um conjunto de listas de tarefas. Cada lista representa as tarefas a serem executadas numa máquina, havendo uma correspondência entre as ordens das máquinas e das listas. Em cada lista, a ordem das tarefas corresponde a ordem de execução.

Exemplificando, o TABELA 4.1 apresenta três produtos com suas respectivas tarefas, que devem ser processadas em quatro máquinas.

Produtos	Máquina 1	Máquina 2	Máquina 3	Máquina 4
X	x_1	x_2	x_3	x_4
Y	y_1	-	y_3	y_4
Z	z_1	z_2	z_3	z_4

TABELA 4.1 - Um Problema a ser Codificado

A codificação de uma solução viável envolve quatro listas, correspondendo cada uma delas as respectivas máquinas (TABELA 4.2).

Listas	Solução 1	Solução 2
L_1	$[x_1, y_1, z_1]$	$[x_1, z_1, y_1]$
L_2	$[z_2, x_2]$	$[x_2, z_2]$
L_3	$[y_3, x_3, z_3]$	$[y_3, x_3, z_3]$
L_4	$[x_4, z_4, y_4]$	$[y_4, z_4, x_4]$

TABELA 4.2 - Soluções Viáveis

Os cromossomas, que representam as soluções viáveis são:

- cromossoma 1 [x_1, y_1, z_1, z_2, x_2, y_3, x_3, z_3, x_4, z_4, y_4]
- cromossoma 2 [x_1, z_1, y_1, x_2, z_2, y_3, x_3, z_3, y_4, z_4, x_4]

4.5 População Inicial

4.5.1 Descrição

Preliminarmente à inicialização da população, é necessário determinar o número de listas (uma para cada máquina) e o conjunto de tarefas em cada uma das listas. O número de indivíduos na população depende do problema a ser resolvido e é um dos parâmetros do algoritmo.

As diferentes soluções (indivíduos) são geradas aleatoriamente, através de permutações nos conjuntos de tarefas de cada uma das listas.

Além disso, soluções encontradas empregando regras heurísticas são incorporadas à população. As regras heurísticas utilizadas são: SPT (tarefa com menor tempo de processamento primeiro), EDD (tarefa com menor data de entrega prevista primeiro) e LST (tarefa com menor folga primeiro).

4.5.2 Algoritmo de Criação da População inicial (CP)

CP1- [Inicializar] Faça **P** = número de cromossomas na população e **M** = número de máquinas;
CP2- [Repetir M vezes] Repita o passo **CP3 M** vezes;
CP3- [Criar lista] Crie lista com todas as tarefas da máquina **i**;
CP4- [Repetir P vezes] Repita o passo **CP5 P** vezes;
CP5- [Criar cromossoma] Crie um cromossoma, com cópias das listas das tarefas, realizando permutações aleatórias em cada lista.
CP6- [Incorporar soluções heurísticas] Solucione o problema usando as regras heurísticas e incorpore as três soluções à população.

4.6 Avaliação

O algoritmo proposto, possibilita a otimização da programação da produção, de acordo com diferentes objetivos. Como, minimização do tempo total de processamento e minimização de produtos atrasados. Para cada um desses objetivos é implementada uma função que avalia o cromossoma, sendo que em cada uma dessas situações é necessário calcular a data mais cedo de início e de término de cada tarefa.

4.6.1 Algoritmo de Cálculo da Data Mais Cedo de Início das Tarefas (MI)

Sejam : **PREDP(j)** a tarefa que precede a tarefa **j** no processo produtivo;

PREDM(j) a tarefa que precede a tarefa **j** na máquina;

MI1- [Inicializar] Faça **N** = número de listas nos cromossomas;
MI2- [Repetir N vezes] Repita os passos **MI3** até **MI9** n vezes;
MI3- [Inicializar número de tarefas] Faça **K** = número de elementos da lista;
MI4 - [Inicializar duração de cada tarefa (D(i))] Inicialize **D(i)** , para **i** de 1 até **K**;
MI4- [Inicializar data mais cedo de início (EST(j))] Faça **EST(j)** = 0, para **j** de 1 até **K**;
MI5- [Repetir para j de 1 até K] Repita os passos **MI6** a **MI9**, para **j** de 1 até **K**;
MI6- [Tarefa tem predecessor no processo produtivo] Se **PREDP(j)**$\neq\varnothing$ faça passo **MI7** ;senão retorne ao passo **MI5**, e incremente **j**;
MI7- [Tarefa tem predecessor na máquina] Se **PREDM(j)**$\neq\varnothing$ faça passo **MI8**;senão faça passo **MI9**;
MI8- [Calcular data mais cedo de início] Faça **EST(j)** = MAX { **EST(PREDP(j))+;EST(PREDM(j))+D(PREDM(j))**}, para todas as tarefas da lista;
MI9- [Calcular data mais cedo de início] Faça **EST(j)** = **EST(PREDP(j))** + **D(PREDP(j))**, para todas as tarefas da lista.

4.6.2 Algoritmo de Cálculo da Data Mais Cedo de Término das Tarefas(MT)

MT1- [Inicializar] Faça **N** = número de listas nos cromossomas;
MT2 - [Calcular data mais cedo de início] Calcule data mais cedo de início **(EST)** para todas as tarefas, usando o algoritmo **MI**;
MT3- [Repetir n vezes] Repita os passos **MT4** até **MT6 N** vezes;
MT4- [Inicializar número de tarefas] Faça **K**= número de elementos da lista;
MT5 - [Iniciaiizar duração de cada tarefa (D(i))] Inicialize **D(i)** , para **i** de 1 até **K**;
MT6- [Repetir para j de 1 até K] Repita o passo **MT7**, para j de **1** até **K**;
MT7- [Calcular data mais cedo de término] Faça **EFT(j)** =**EST(j)+D(j)**.

4.6.3 Minimização do Tempo Total de Processamento

4.6.3.1 Descrição

Com o objetivo de minimizar o tempo total de processamento, determina-se para cada cromossoma, o tempo que todas as tarefas levam para serem executadas. No cálculo, além das precedências entre as tarefas de um mesmo produto, são consideradas as precedências em cada máquina, representadas pela ordem em que aparecem em cada uma das listas do cromossoma.

4.6.3.2 Algoritmo de Cálculo do Tempo Total de Processamento (TT)

TT1- [Inicializar] Faça **K** = número de elementos no cromossoma;
TT2 - [Calcular data mais cedo de início] Calcule data mais cedo de início (**EST**) para todas as tarefas, usando o algoritmo **MI**;
TT3 - [Calcular tempo total de processamento] Faça
AVALIAÇÃO = Máx{**EST**(**j**)},para **j** de 1 até **K**.

4.6.4 Minimização de Produtos Atrasados

4.6.4.1 Descrição

Para cada cromossoma determina-se a data de finalização da última tarefa de cada produto, considerando as precedências em cada máquina, de acordo com a ordem em que as tarefas aparecem em cada uma das listas dos cromossomas e as precedências entre as tarefas de um mesmo produto.

Comparando essa data com a prevista para a entrega do produto. Dessa forma, encontrando-se o seu tempo de atraso.

O atraso médio é, então, calculado pela fórmula:

$$A.M. = \frac{\sum_{i=1}^{n} L_i}{n} \qquad (4.1)$$

$$L_i = Dd_i - Dt_i \text{ , para } L_i > 0 \qquad (4.2)$$

onde: A.M. = atraso médio

L_i = tempo de atraso do produto i

n = número de produtos atrasados

Dd_i = data de entrega prevista do produto i

Dt_i = data de término da produção do produto i

4.6.4.2 Algoritmo de Cálculo do Atraso Médio (AM)

AM1- [Inicializar] Faça **P**= número de produtos; **SOMA** = 0;
AM2 - [Inicializar data de entrega (DE(i))] Inicialize **DE(i)** , para **i** de 1 até **P**;
AM3 - [Calcular data mais cedo de término] Calcule data mais cedo de término de cada tarefa (**EFT**), usando o algoritmo **MT**;
AM4 - [Repetir p vezes] Repita o passo **AM5** para **i** de 1 até**P**;
AM5 - [Calcular atraso dos produtos] Se **DE(i)** < **EFT(i)**, onde **EFT(i)** = tempo mais cedo de término da última tarefa do produto **i**, faça
SOMA = **SOMA** +[**EFT(i)** - **DE(i)**];
AM6 - [Calcular atraso médio dos produtos]Faça:
AVALIAÇÃO=SOMA /(número de produtos com atraso);

4.6.5 Minimização do Nível de Estoques Intermediários

4.6.5.1 Descrição

Visando minimizar os estoques de produtos em processamento é calculada a proporção do tempo que os produtos ficam estocados intermediáriamente, esperando liberação das máquinas.

Para esse fim, são avaliadas a data mais cedo de início e término de cada tarefa. Neste cáculo são utilizadas as precedências em cada máquina de acordo com a ordem em que as tarefas aparecem em cada lista do cromossoma e as precedências das tarefas nos produtos.

A proporção é então calculada pela fórmula:

$$Ei = \sum_{i=1}^{n} \frac{(Dt_i - Rt_i) - \sum_{j=1}^{m} t_{ij}}{(Dt_i - Rt_i)} \qquad (4.3)$$

onde: Ei = estoques intermediários

Dt_i = data de término da produção do produto i

Rt_i = data de entrada do produto i no ambiente de produção

t_{ij} = tempo de processamento da tarefa j do produto i

4.6.5.2 Algoritmo de Avaliação do Nível de Estoques Intermediários (EI)

EI1- [Inicializar] Faça **N** = número de listas nos cromossomas; **K** = número de produtos; **SOMA** = 0;
EI2 - [Inicializar tempo de processamento de cada produto TP(i))] Inicialize **TP(i)**, para **i** de 1 até **K**;
EI3 - [Calcular data mais cedo de término] Calcule data mais cedo de término (**EFT**) para todas as tarefas, usando o algoritmo **MT**;
EI4 - [Repetir K vezes] Repita o passo **EI5** para **i** de 1 até **K**;
EI5 - [Calcular Nível de Estoques] Faça
SOMA = **SOMA** + ((**EFT(i)** - **TP(i)**)/**EFT(i)**,sendo **EFT(i)** a data mais cedo de término da última tarefa do produto **i**;
EI6 - [Calcular avaliação] Faça **AVALIAÇÃO** = **SOMA.**

4.7 Aptidão

4.7.1 Descrição

Visando determinar a aptidão do cromossoma, é realizada uma transformação de avaliação de custos para aptidão. Com essa finalidade determina-se primeiro o custo máximo (Tempo Total de Processamento ou Atraso Médio) de toda a população. A seguir, cada uma das avaliações é transformada em aptidão:

$$f(x) = Cmáx - g(x), \text{ com } g(x) < Cmáx \qquad (4.4)$$

onde: f(x) = aptidão do cromossoma
g(x) = avaliação do cromossoma
Cmáx = máxima avaliação

Com o objetivo de controlar o número de descendentes é realizada uma transformação de escala na aptidão através da fórmula:

$$f' = af + b \qquad (4.5)$$

onde: f' = aptidão transformada
f = aptidão
a e b = constantes

As constantes a e b foram escolhidas de forma que f'máx = 2 f, o que torna o número esperado de descendentes do melhor cromossoma o dobro do número esperado de descendentes do cromossoma "médio".

4.7.2 Algoritmo de Cálculo da Aptidão (AP)

AP1- [Inicializar] Faça C_{max} = 0 e P= número de cromossomas;
AP2 - [Inicializar avaliação do cromossoma(AV(i))] Inicialize **AV(i)**, para **i** de 1 até **P**;
AP2- [Repetir P vezes] Repita o passo **AP3 P** vezes;
AP3- [Atualizar C_{max}] Se **AV(i)** > C_{max}, então faça C_{max}= **AV(i)**;
AP4- [Repetir p vezes] Repita o passo **AP5 P** vezes;
AP5- [Calcular aptidão do cromossoma] Faça
APTIDÃO = C_{max} - **AV(i)**.

4.8 Reprodução com Elitismo

4.8.1 Descrição

Na reprodução dos cromossomas para gerar uma nova população é utilizada a reprodução com elitismo. Nesse tipo de reprodução, um percentual

dos indivíduos (parâmetro do algoritmo) da população permanece na nova população, sendo os demais gerados a partir daquela população.

4.8.2 Algoritmo de Reprodução com Elitismo (RE)

RE1- [Inicializar] Faça EL = percentual de elitismo x número de indivíduos na população;
RE2- [Selecionar melhores] Da população a ser reproduzida, selecione os **EL** indivíduos com aptidão mais alta;
RE3- [Repetir EL vezes] Repita o passo **RE4 EL** vezes;
RE4- [Gerar cópia dos melhores] Gere um novo cromossoma igual a um dos melhores da população e inclua na nova população.

4.9 Seleção de Pais

4.9.1 Descrição

Com o objetivo de proporcionar maiores chances de reprodução aos cromossomas com maior aptidão é utilizada a técnica da "roleta com viés". Onde cada cromossoma possui a probabilidade de ser escolhido, proporcional a sua avaliação. E, então, é calculado para cada cromossoma uma probabilidade igual a:

$$\frac{f_i}{\sum f_i} \qquad (4.6)$$

onde: f_i = aptidão do cromossoma i.

4.9.2 - Algoritmo de Seleção dos Pais (SP)

SP1- [Inicializar] Faça P= número de cromossomas na população;
SP2- [Construir a roleta] Faça **SOMA** = soma da aptidão de todos os indivíduos na população. Construa **SOMA** posições e atribua a cada cromossoma um número de posições igual à sua aptidão;
SP3- [Selecionar cromossoma pai] Gere um número aleatório entre 0 e **SOMA** e use o número para selecionar uma posição e encontrar o cromossoma correspondente.

4.10 Operadores de Cruzamento

A combinação de dois cromossomas para obtenção de duas novas soluções é um dos princípios em que se fundamenta os algoritmos genéticos. Isso relacionado com o problema da programação da produção, significa admitir a hipótese de que partes do programa da produção, representadas nos cromossomas, podem pertencer a solução ótima.

Os operadores de cruzamento procuram combinar várias das partes "ótimas", na tentativa de encontrar a solução ótima do problema.

Na concepção dos operadores, especial atenção deve ser dada a possibilidade de gerar codificações que não representam soluções viáveis. Se o número de codificações que correspondem as soluções inviáveis geradas pelo operador é pequeno, ele pode ser utilizado e, posteriormente, descartadas essas soluções. Por outro lado, se o operador tende a gerar muitas soluções inviáveis, isto pode comprometer a eficiência do algoritmo genético. É desejável, portanto, que o operador sempre gere codificações representando soluções viáveis, mesmo que seja necessário introduzir alguma complexidade. Deve-se sempre manter um compromisso entre a capacidade do operador para pesquisar diferentes esquemas (porções ótimas), a complexidade do mesmo, e o custo de gerar soluções inviáveis, verifica-las quanto a viabilidade.

No algoritmo genético desenvolvido foram usados dois operadores. O primeiro deles denominado cruzamento em pontos fixos e o segundo operador uma adaptação do MPX de Goldberg (GOLDBERG, 1975).

4.10.1 Cruzamento em Pontos Fixos

4.10.1.1 Descrição

Este operador gera sempre codificações que representam soluções viáveis. Basicamente, é um operador de cruzamento com um ponto. O ponto de cruzamento é selecionado aleatoriamente entre as posições fixas que correspondem aos limites entre as diferentes listas de tarefas. Dessa forma, os descendentes possuem a mesma ordem de tarefa nas máquinas que os pais, sendo que um grupo dessas ordenações pertence a um dos pais, e o restante ao outro.

4.10.1.2 Algoritmo de Cruzamento em Pontos Fixos (CP)

CP1- [Iniciaiizar] Faça **NPONT** = número de listas -1;
CP2- [Escolher os pais] Escolha 2 cromossomas pais utilizando o algoritmo **SP**;
CP3- [Reproduzir] Faça cópia dos dois cromossomas escolhidos (**C1** e **C2**);
CP4- Determinar pontos de cruzamento] Faça **PC** = número aleatório entre 1 e **NPONT**;
CP5- [Trocar listas] Coloque as listas 1 até **PC** de **C1** em **C2** e vice-versa.

4.10.2 MPX Modificado

4.10.2.1 Descrição

Este operador é uma adaptação do operador MPX de Goldberg.

Nesse operador, o cruzamento é realizado em uma única lista (máquinas) dos dois cromossomas pais. Em uma primeira etapa, a posição da lista onde será executado o cruzamento é escolhida aleatoriamente. Em seguida, um intervalo igual em cada lista é também, determinado aleatoriamente. Os intervalos são, então, trocados entre as duas soluções. Na continuação, alterações nos descendentes são realizadas, de forma a garantir a viabilidade da solução por ele codificada.

O operador apresenta uma complexidade maior que a anterior, mas permitindo alterar os esquemas dentro de uma determinada máquina.

4.10.2.2 Algoritmo MPX Modificado (MM)

MX1- [Inicializar] Faça **L** = número de listas;
MX2- [Escolher uma lista] Faça **LE** = número aleatório entre 1 e **L**;
MX3- [Escolher intervalo] Escolha dois números aleatórios **C1** e **C2** entre 1 e tamanho da lista **LE**;
MX4 - [Retirar sublistas] Faça **I1** = sublista de **C1** a **C2** no cromossoma 1
I2 = sublista de **C1** a **C2** no cromossoma 2
MX5 - [Trocar subiistas] Coloque a sublista **I1** no cromossoma 2 e **I2** no cromossoma 1;
MX6 - [Trocar cromossomas viáveis] Usar as sublistas para realizar trocas nos novos cromossomas que os tornem soluções viáveis.

4.11 Operadores de Mutação

Com a finalidade de evitar convergência prematura do algoritmo e ao mesmo tempo gerar esquemas ainda não explorados são utilizados dois tipos de operadores de mutação: mutação de posição e mutação de ordem.

4.11.1 Mutação de Posição

4.11.1.1 Descrição

Num operador de mutação de posição são selecionados aleatoriamente uma das listas do cromossoma e, a seguir, duas posições dentro desta. O descendente é, então, gerado com a mesma estrutura do pai, a exceção das tarefas situadas nas posições sorteadas que são trocadas entre si. A utilização desse operador gera sempre soluções viáveis.

4.11.1.2 Algoritmo de Mutação de Posição (MP)

MP1- [Iniciaiizar] Faça **L** = número de listas;
MP2- [Escolher uma lista] Faça **LE** = número aleatório entre 1 e **L**;
MP3- [Escolher tarefas] Na lista **LE**, escolha aleatoriamente duas tarefas (**T1** e **T2**);
MP4- [Trocar tarefas] Coloque a tarefa **T1** na posição de **T2** e vice-versa.

4.11.2 Mutação de Ordem

4.11.2.1 Descrição

Neste operador, uma lista é selecionada aleatoriamente, e nessas duas tarefas são escolhidas, também de forma aleatória, sendo a segunda delas colocada na posição que antecede a primeira, para a geração do descendente. As soluções geradas são sempre viáveis.

4.11.2.2 Algoritmo de Mutação de Ordem (MO)

MO1- [Inicializar] Faça **L** = número de listas;
MO2- [Escolher uma lista] Faça **LE** = número aleatório entre 1 e L;
MO3- [Escolher tarefas] Na lista **LE**, escolha aleatoriamente duas tarefas (**T1** e **T2**);
MO4- [Trocar tarefas] Coloque a tarefa **T2** na frente de **T1**.

4.12 Algoritmo Completo

4.12.1 Descrição

O algoritmo proposto utiliza as estruturas anteriormente descritas na busca de uma solução para o problema da programação da produção em ambientes do tipo "flow shop".

Dada as características da abordagem, onde não é necessário o conhecimento das propriedades da função objetivo (continuidade, existência de derivada, etc), vários critérios de programação são propostos. Esses critérios são

utilizados para avaliar os cromossomas, devendo ser escolhido um deles de acordo com o objetivo do usuário.

Além da função de avaliação, o percentual de elitismo, o tamanho da população e o peso relativo dos operadores constituem parâmetros do algoritmo, afetando o seu desempenho.

Um dos critérios de parada do algoritmo é o número de populações avaliadas. Uma vez que cada geração envolve a avaliação de um número de soluções do tamanho da população, esse critério, indiretamente, expressa o número máximo de soluções a serem avaliadas. Um outro critério de parada ocorre quando todos os cromossomas da população convergem para uma única solução, dessa forma descaracterizando o algoritmo genético. Sendo o critério mais utilizado, o primeiro deles, e portanto considerado o principal.

4.12.2 Algoritmo

AG1 - [Inicializar] Faça **K** = número de iterações;
AG2 - [Criar população inicial] Crie população inicial (Algoritmo **CP**);
AG3 - [Repetir K vezes] Faça passos **AG4** a **AG11 K** vezes;
AG4 - [Calcular avaliação] De acordo com a função objetivo escolhida, calcule avaliação de cada cromossoma (Algoritmos **TT**, **AM** ou **EI**) ;
AG5 - [Calcular aptidão] Calcule aptidão de cada cromossoma (Algoritmo **AP**);
AG6 - [Realizar reprodução] Realize reprodução com elitismo (Algoritmo **RE**);
AG7 - [Realizar cruzamentos e mutações] De acordo com as probabilidades especificadas, faça os passo **A8** a **A15** o número de vezes necessário para a criação de nova população;
AG8 - [Escolher pais] Escolha 2 cromossomas (usando Algoritmo **SP**);
AG9 - [Realizar cruzamento] Realize cruzamento (Algoritmo **CF**);
AG10 - [Escolher pai] Escolha 1 cromossoma (usando Algoritmo **SP**);
AG11 - [Realizar mutação de posição] Realize mutação de posição (Algoritmo **MP**);
AG12 - [Escolher pai] Escolha 1 cromossoma (usando Algoritmo **SP**);
AG13 - [Realizar mutação de ordem] Realize mutação de ordem (Algoritmo **MO**);
AG14 - [Escolher pais] Escolha 2 cromossomas (usando Algoritmo **SP**);
AG15 - [Realizar cruzamento MPX] Realize cruzamento MPX (Algoritmo **MX**);
AG16 - [Calcular avaliação] Calcule avaliação de cada cromossoma (Algoritmos **TT, AM ou EI**, de acordo com função objetivo);
AG17 - [Encontrar solução] Faça **SOLUÇÃO** = Melhor avaliação.

5. Calibração do Algoritmo Genético Proposto: Uma Abordagem Utilizando Raciocínio Difuso

5.1 Introdução

Raciocínio Difuso é o processo ou processos pelos quais uma conclusão possivelmente imprecisa é deduzida de uma coleção de premissas imprecisas. Tal raciocínio é, na maioria dos casos, antes qualitativo que quantitativo, e quase sempre cai fora do domínio de aplicação da lógica clássica (DUBOIS e PRADE, 1980).

Em situações nas quais uma modelagem matemática precisa se torna inviável, dada a imprecisão dos termos envolvidos e a presença de informações incompletas, a lógica difusa asssume particular importância.

Sistemas reais complexos são, normalmente, constituídos de inúmeros parâmetros, de tal forma que impossibilitam ao agente atuar sempre com a precisão requerida. Sistemas especialistas têm sido desenvolvidos, principalmente para auxiliar em tarefas que envolvam o estabelecimento de valores para parâmetros que apresentam uma grande variação.

Na aplicação de algoritmos genéticos, uma série de parâmetros deve ser estabelecida, determinando parte do seu sucesso. A calibração desses parâmetros requer a intervenção de um especialista familiarizado com essa técnica e com o problema objeto da aplicação, uma vez que as relações entre os mesmos são extremamente complexas. Dessa forma, propõe-se, para a calibração do algoritmo genético desenvolvido nesse trabalho, uma abordagem utilizando raciocínio difuso.

5.2 Fundamentos Teóricos

5.2.1 Sistemas Especialistas

As técnicas de inteligência artificial permitem abordar problemas para cuja solução não existem algoritmos definidos ou eficientes. Nesses casos, o conhecimento do processo de solução possibilita que sejam utilizadas regras heurísticas que, apesar de não garantirem a melhor solução, fornecem uma solução viável, próxima da ótima. As técnicas de sistemas especialistas, em especial, permitem que diversas fontes de conhecimento sejam combinadas, na procura de uma solução.

Sistemas reais complexos constituem-se de inúmeras regras que impossibilitam ao agente humano agir sempre com a precisão requerida. Sistemas especialistas têm sido desenvolvidos para auxiliar tarefas que demandam tempo excessivo, por exemplo, elevado número de cáculos, avaliação de situações múltiplas, conhecimento empírico, etc.

5.2.1.1 Definição de Sistemas Especialistas

Sistemas que são organizados de forma a separar o conhecimento sobre o domínio do problemas de outros conhecimentos do sistema, como conhecimento geral sobre como solucionar problemas ou conhecimento sobre como interagir com o usuário, são sistemas especialistas (WATERMAN, 1986).

Nesses sistemas, a coleção do domínio de conhecimento é chamada base de conhecimento e o conhecimento para solucionar problemas é chamado motor de inferência. Portanto, esses são sistemas baseados em conhecimento.

A base de conhecimento contém fatos e regras (ou outras representações) e usa os fatos como base para a tomada de decisão.

O motor de inferência pode ser comparado aos algoritmos dos programas convencionais, exercendo a estratégia de controle na busca de uma solução.

Em um sistema especialista, vários são os atributos desejáveis. Entre os principais, pode-se citar (WILLIAMS in GAUTHIER, RODRIGUES FILHO e BARCIA, 1990):

- Deve separar o conhecimento do domínio específico e os procedimentos de resolução do problema, e incluir os conceitos de base de conhecimento e de motor de inferência;

- Deve simular o pensamento do ser humano;

- Deve ser capaz de aprender através da experiência;

- Deve produzir uma base de conhecimento dinâmica, de modo a ser facilmente colocada em diferentes módulos de conhecimento;

- Deve interagir em linguagem natural, no domínio do conhecimento;

- Deve apresentar para o usuário uma estratégia de controle simples e transparente;

- Deve ser computacionalmente rápido e não demandar excessivos recursos de hardware;

- Deve ser capaz de raciocinar em condições de incerteza e de insuficiência de informações.

5.2.1.2 Aquisição e Representação do Conhecimento

A aquisição do conhecimento envolve dedução, análise e interpretação do conhecimento que um especialista humano usa, quando resolve um problema particular (KIDD in NIX, COLLINS e TSAY, 1989).

De forma geral, o objetivo da aquisição de conhecimento consiste na transferência e transformação do conhecimento de alguma fonte, freqüentemente humana, para um programa de computador.

A aquisição de conhecimento pode ser encarada, sob o ponto de vista psicológico, como a construção de um modelo que expresse o modo de pensar do especialista. Compreende as atividades realizadas por uma pessoa, o engenheiro de conhecimento, para obter o material de alguma fonte relevante, analisar e interpretar esse material e colocá-lo de uma forma que possa ser utilizado em uma linguagem de programação.

O processo de aquisição de conhecimento divide-se em três fases distintas: familiarização, entrevistas e análises.

Na fase de familiarização, o domínio sob consideração é completamente estudado, para que o engenheiro de conhecimento torne-se familiarizado com a essência do domínio.

Na fase seguinte, da entrevista, o engenheiro de conhecimento induz o especialista a revelar o seu conhecimento em termos de problemas, objetivos e soluções disponíveis. Essa fase pode ser considerada a mais difícil, pois os especialistas podem não estar certos de suas experiências, ou podem ser incapazes de expressá-las, ou ainda a experiência que expressam pode ser irrelevante ou incorreta.

Na fase de análise, o conhecimento do especialista é organizado em termos de subcomponentes ou subproblemas do domínio. Cada componente é estudado pela sua freqüência, objetivos, soluções possíveis, cenários e habilidades requeridas.

A representação do conhecimento é composta por um conjunto de mecanismos usados para armazenar e manipular o conhecimento. Esse conhecimento apresenta-se sob a forma de objetos, classes, categorias e descrições; eventos; habilidades e metaconhecimento.

Várias são as formas de representação de conhecimento. Entre elas, pode-se citar: regras de produção, redes semânticas e "frames".

Regras de produção consistem de um conjunto de condições e de um conjunto de ações. Se todas as condições são verdadeiras, então as ações são executadas. Dado que os especialistas humanos nem sempre têm certeza sobre essas regras, medidas ou fatores de certeza podem ser atribuídos tanto às condições como às ações.

As redes semânticas enfatizam as relações entre os diferentes objetos e consistem em "nós", que representam esses objetos ou conceitos, e "arcos", que, ligando-os, representam as relações entre eles.

"Frames" representam objetos ou conceitos, através do conjunto de seus atributos e operadores de manipulação. Na sua forma geral, "frame" é representado por um nome com uma série de atributos, com os seus respectivos valores, e um conjunto de métodos para a manipulação do conceito por ele representado.

5.2.1.3 Conhecimento Impreciso em Sistemas Especialistas

A grande maioria das ferramentas utilizadas na construção de sistemas especialistas usa fatores de certeza ou de confiança, para manipular incertezas no conhecimento ou nos dados. Porém, essas não podem enfrentar conceitos como: baixo, ruim ou frio, os quais constituem uma parte muito significante da linguagem natural (LEUNG e LAM, 1988).

O conhecimento humano, na sua grande parte, é vago e impreciso. Tanto o pensamento, como o raciocínio humano envolvem informações inexatas. As técnicas utilizadas nos sistemas especialistas permitem trabalhar com esses

tipos de informações, através das seguintes fontes: conceitos difusos inerentemente humanos, informações não seguras, informações incompletas e opiniões diferentes de especialistas.

Neste contexto, LEUNG e LAM (1988) classificam o conhecimento inexato em quatro tipos: incerteza, difusão, incerteza e difusão simultâneas e incerteza difusa.

A incerteza ocorre quando, além de imprecisões derivadas de processos de mensurações, têm-se também, imprecisões derivadas de processos de valoração. O grau de certeza é, usualmente, representado por um valor numérico.

Exemplo: X é um pássaro (0,7)
Se X é um pássaro
Então ele pode voar (0,9)

Os fatores de certeza são 0,7 e 0,9 (também chamados de fatores de confiança).

A difusão ocorre quando o limite de uma parte da informação não é bem definido.

Exemplo: Marta é muito gorda.
Se o preço é baixo, então o lucro é pouco.

Muito gorda, baixo e pouco são termos difusos.

Em algumas situações, podem ocorrer incerteza e difusão simultaneamente.

Exemplo: Maria é muito baixa (0,8).

0,8 é fator de certeza e muito baixa é um termo difuso.

Existem situações em que a incerteza é difusa.

Exemplo: Joana é muito bonita (em torno de 0,8).

Muito bonita é um termo difuso e em torno de 0,8 é a incerteza difusa.

5.2.2 Conjuntos Difusos

A teoria dos conjuntos difusos desenvolvida por ZADEH (1965) fornece um instrumento adequado para modelar situações em que ocorram imprecisões ou incertezas. Os recentes progressos verificados nessa teoria têm contribuído para dissipar aspectos não muito claros e tornado viável o desenvolvimento de ferramentas de análise. Essa teoria, além de permitir a generalização de medidas de incerteza, possibilita a ampliação considerável de aplicações de caráter prático.

5.2.2.1 Definição de Conjuntos Difusos

A teoria clássica dos conjuntos caracteriza-se pelo fato de que um elemento pertence ou não pertence a um conjunto, não existindo uma situação intermediária. Considere-se que X seja um conjunto clássico de objetos, denominado universo, cujos elementos são designados por x, e que A seja um subconjunto de X. Uma função f_A é definida por:

$f_A(x) = 1$ se e somente se $x \in A$
$f_A(x) = 0$ se e somente se $x \notin A$

f_A é uma função característica de X em {0,1}, sendo {0,1} chamado de conjunto de avaliação.

A idéia fundamental da teoria dos conjuntos difusos, proposta por ZADEH (1965), é que um elemento pertence com certo grau a um conjunto, chamado de conjunto difuso.

Matematicamente, pode-se definir conjunto difuso da seguinte forma: "seja X um espaço de pontos (objetos) denotados genericamente por x . Um conjunto difuso A em X é caracterizado por uma função de pertinência f_A (x), que associa com cada ponto em X um número real no intervalo [0,1], com um valor de f_A (x) em X representando o grau de pertinência de x em A. Então, quanto mais próximo da unidade está o valor da f_A (x), mais alto é o grau com que x pertence a A " (ZADEH, 1965).

A partir dessa definição, foi desenvolvida uma teoria, englobando conceitos de operações algébricas com conjuntos difusos, relações difusas, números difusos, lógica difusa e outros.

A utilização desses conceitos permite modelar situações do mundo real, onde as proposições não necessariamente são verdadeiras ou falsas, sendo o grau de veracidade (e, em consequência, de falsidade) medido através do "grau de pertinência".

Na lógica tradicional, o principal meio de raciocínio é constituído por tautologias como o "modus ponens", ou seja:

premissa: A é verdadeiro
implicação: se A então B
conclusão: B é verdadeiro

Uma generalização para "modus ponens" difuso inclui caracterizar as premissas como conjuntos difusos (ZIMMERMANN, 1985):

premissa: x é A
implicação: se x é A' então y é B'
conclusão: y é B

5.2.2.2 Funções de Pertinência

A função de pertinência é usada para medir o grau de pertinência de um elemento a um conjunto. Quando se trata de um conjunto difuso, assume

um valor no intervalo [0,1]. Ela reflete o conhecimento que se tem em relação à intensidade com que um elemento pertence a um conjunto (PAO, 1989).

Para a construção de funções de pertinência, deve-se verificar as seguintes propriedades (DOMBI, 1990):

- Todas as funções de pertinência são contínuas;

- Todas as funções de pertinência mapeiam um intervalo [a, b]→ [0,1] → f [a,b] → [0,1];

- As funções de pertinência são:
  - monotonicamente crescentes,
  - monotonicamente decrescentes, ou
  - subdivididas em parte crescente e parte decrescente;

- As funções de pertinência monótonas sobre um intervalo completo são:
  - funções convexas, ou
  - funções côncavas, ou
  - existe um ponto "c" no intervalo [a,b] tal que [a, c] é convexo e [c, b] é côncavo;

- Funções monotonicamente crescentes têm a propriedade f (a) = 0 e f (b) = 1, enquanto funções monotonicamente decrescentes têm a propriedade f (a) = 1 e f (b) = 0;

- As funções de pertinência devem apresentar uma forma linear ou devem ser linearizadas.

Na aplicação da teoria dos conjuntos difusos, um dos tópicos mais importantes é a estimação da função de pertinência, pois uma estimativa adequada valida os resultados obtidos.

De acordo com DEVI e SARMA (1985), os pontos importantes a serem considerados para a estimação da função de pertinência são a forma, os parâmetros e o domínio da função.

Para estimação da função de pertinência, vários métodos podem ser utilizados. Entre eles, citam-se o método baseado em histograma e o método baseado no consenso de especialistas.

5.2.2.3 Lógica Difusa

A lógica clássica fundamenta-se em três princípios básicos:

- Princípio da identidade: X = X;

- Princípio do terceiro excluído: dados "a" e "não a", uma é verdadeira;

- Princípio da contradição: dados "a" e "não a", uma é falsa.

As demais lógicas diferem da lógica clássica pela violação a esses princípios.

O raciocínio humano é, por sua própria natureza, aproximado, sendo que o uso de técnicas com o objetivo de modelá-lo não está considerando a principal vantagem do ser humano, que é a de tratar conceitos inexatos.

A lógica difusa leva em consideração o conhecimento humano e coloca que esse conhecimento deve ser tratado como tal, ou seja, sem modelá-lo para a "exatidão".

Nesse contexto, as lógicas, como base para o raciocínio podem ser distinguidas por: valores verdade, vocabulário e processo de raciocínio.

A lógica clássica e a matemática clássica apresentam uma característica típica, que é a presença de decisão dicotômica, onde os valores verdade são "0 ou 1", e o vocabulário é definido através desses valores verdade sob forma de tabelas verdade,. enquanto a lógica difusa baseia-se nas lógicas multivaloradas, em que os valores verdade variam no intervalo [0,1]. Para ZADEH (1973), os valores verdade da lógica difusa são variáveis lingüísticas.

Verifica-se que, no mundo real, os fatos não são caracterizados somente como verdadeiro ou falso, e sim como verdadeiro, mais ou menos verdadeiro, mais ou menos falso, etc. A similaridade existente entre essas expressões e os valores de uma variável lingüística induz a definição da variável lingüística verdade. O tratamento da verdade como uma variável lingüística conduz à lógica difusa. E essa, por sua vez, fornece uma base para o raciocínio difuso.

Raciocínio difuso é entendido como o processo ou processos pelos quais uma conclusão, possivelmente imprecisa, é deduzida de uma coleção de premissas imprecisas. Tal raciocínio é, na maioria dos casos, antes qualitativo do que quantitativo por natureza, e quase sempre cai fora do domínio de aplicabilidade da lógica clássica (DUBOIS e PRADE, 1980).

Os modelos de raciocínio difuso são considerados muito úteis na resolução de problemas práticos. O uso de um controlador lógico difuso, baseado em lógica difusa, permite converter uma estratégia de controle lingüístico, baseada em conhecimento de especialistas, em uma estratégia de controle automático.

Para calibrar os parâmetros de um algoritmo genético, em um sistema computacional, verifica-se a necessidade de um método que descreva a estratégia utilizada, ou seja, as descrições verbais devem passar para uma relação funcional, e essa deve representar o conhecimento do especialista.

5.3 Problema da Calibração

A calibração de um algoritmo genético é uma tarefa árdua. No seu trabalho original, DE JONG (in DAVIS 1979) descreve que levou muito tempo para sua determinação. Estudando a calibração de algoritmos genéticos na minimização de funções SHAFFER, CARUNA, ESHELMAN e DAS (1989) utilizaram experimentação intensiva (1,5 anos de CPU), para concluir que os parâmetros variam de problema para problema, mas alguns parâmetros robustos se comportavam de maneira satisfatória nos problemas por eles estudados.

A calibração com prâmetros robustos derivados dos algoritmos genéticos tradicionais perde sua eficiência quando se trabalha com algoritmos genéticos adaptados a problemas específicos.

5.4 Organização de um Sistema Difuso para Calibração do Algoritmo Genético Proposto

A calibração de um algoritmo genético deve ser efetuada de acordo com o problema a ser resolvido. Essa calibração merece uma atenção especial, quando o algoritmo é proposto para a solução de vários problemas.

Para a calibração, é requerida a intervenção de um especialista que tenha conhecimento teórico e prático, pois as relações envolvidas contêm informações qualitativas e subjetivas, cujo escopo pode ser formalizado através da lógica difusa.

Nesse contexto, a abordagem proposta incorpora o conhecimento subjetivo do especialista, através da introdução dos conceitos de raciocínio difuso, variáveis lingüísticas e regras difusas.

Esquematicamente, o módulo proposto é da forma representada na FIGURA 5.1.

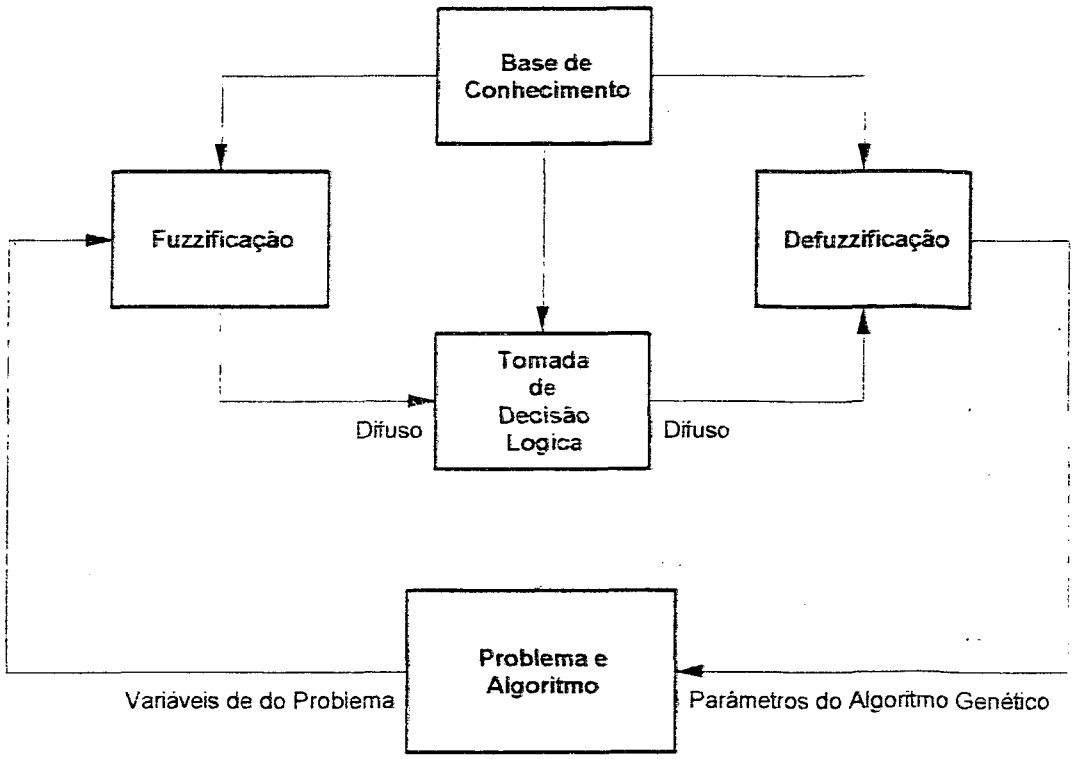


FIGURA 5.1- Módulo de Calibração Proposto

Este módulo caracteriza-se pelos mecanismos de "fuzzificação", "desfuzzificação" e inferência,. atuando em conjunto com uma base de conhecimento composta por regras de inferência difusa e fatos difusos. Esses itens são descritos detalhadamente a seguir.

5.4.1 Base de Conhecimento

A base de conhecimento contém as regras e uma base de dados. Na base de dados, são armazenadas as variáveis lingüísticas. Foram utilizados o número de máquinas e o número de produtos como variáveis do problema; como parâmetros, o número de cromossomas, o número de gerações, e um parâmetro para fazer uma combinação linear entre os dois operadores de cruzamento.

Zadeh (1975) define variável lingüística como um sistema composto por cinco informações <L,T,X,G,M>. O nome da variável é representado por L

(e.g., número de gerações). Os diferentes rótulos de subconjuntos difusos de um universo de discurso, (e.g., grande, médio, baixo) são representados pelo conjunto T. O símbolo X refere-se ao universo de discurso. Regras sintáticas, que definem uma sentença completa em T, e semânticas, que permitem, através de regras, determinar a pertinência de uma sentença em T, são representadas no símbolo G. Assim, na base de dados, para cada variável lingüística é definido um conjunto de rótulos(T) e as funções de pertinencia de cada rótulo(M). As funções de pertinência foram definidas através de oito parâmetros, que compreendem a parte crescente e descrescente da função (DOMBI, 1990). A função de DOMBI contém uma parte crescente (5.1) e uma decrescente(5.2)

$$\mu(x) = \frac{1}{1 + \left(\frac{v}{1-v}\right)^{\lambda-1} \left(\frac{b-x}{x-a}\right)^{\lambda}} \qquad (5.1)$$

$$\mu(x) = \frac{1}{1 + \left(\frac{v}{1-v}\right)^{\lambda-1} \left(\frac{x-a}{b-x}\right)^{\lambda}} \qquad (5.2)$$

Para cada uma das variáveis, definem-se três rótulos: pequeno, médio e grande.

Por exemplo, para a variável número de máquinas tem-se:

- L - Maq
- T - { p, m, g}
- X - { 1, 50}
- G - { Se Maq entre 5 e 10 então T(Maq) = p
  Se Maq entre 7 e 20 então T(Maq) = m
  .....}
- M - $\{(x,\mu_p(x));(x,\mu_m(x));(x,\mu_g(x))\}$

Além, das definições das variáveis envolvidas, a base de conhecimento mantém regras de decisão que relacionam parâmetros do problema com parâmetros do algoritmo. As regras difusas relacionam os parâmetros do problema a ser solucionado com os do algoritmo genético, e são construídas utilizando as variáveis lingüísticas.Por exemplo:

Se **Maq** é **p** e **Prod** é **p**
Então: **Ger** é **p** com grau 0.9;
Pop é **p** com grau 0.9
Cr é **p** com grau 0.9

5.4.2 Mecanismo de "Fuzzificação"

Os valores das variáveis do problema são adquiridos de forma "crisp"e transformados em valores difusos dos diferentes termos das variáveis lingüísticas, através do mecanismo de "fuzzificação". Esse mecanismo utiliza as funções de pertinência de cada rótulo para realizar essa operação. Assim, obtém-se uma série de fatos difusos a respeito do probiema, que vão permitir concluir sobre quais os parâmetros adequados.

5.4.3 Mecanismo de Raciocínio Difuso

O mecanismo de de raciocínio foi adaptado do trabalho apresentado por CAO, KANDEL e LI (1990). Nele, todas as regras são avaliadas simultaneamente.

Dadas as variáveis lingüísticas X_1, X_2 ....X_n, cada uma com k rótulos, tem-se:

$A_{11}, A_{21}, \ldots A_{k1}$; rótulos de X_1
$A_{12}, A_{22}, \ldots A_{k2}$; rótulos de X_2
.....
.....
$A_{1n}, A_{2n}, \ldots A_{kn}$; rótulos de X_n

Foram utilizadas como variáveis o número de produtos(**Prod**), e o número de máquinas(**Maq**). Os rótulos utilizados são pequeno (**p**), médio(**m**) e grande(**g**), para todas as variáveis lingüísticas. Os parâmetros a ser

determinados são: número de cromossomas(**Pop**), numero de gerações(**Ger**) e relação entre os operadores de cruzamento(**Cr**).

As regras podem ser escritas como:

Se X_1 é A_{11} e X_2 é A_{12} .....X_n é A_{1n} então Y é B_1
Se X_1 é A_{21} e X_2 é A_{22} .....X_n é A_{2n} então Y é B_2

.....
Se X_1 é A_{k1} e X_2 é A_{k2} .....X_n é A_{kn} então Y é B_k

onde: B_1, B_2, ... B_k = conjuntos difusos de todos os m valores lingüísticos da variável Y

Tem-se :

$B_1 = b_{11}/Y_1 + b_{12}/Y_2 + ...+ b_{1m}/Y_m$
$B_2 = b_{21}/Y_1 + b_{22}/Y_2 + ...+ b_{2m}/Y_m$
.....
$B_k = b_{k1}/Y_1 + b_{k2}/Y_2 + ...+ b_{km}/Y_m$

Por exemplo:

Se **Maq** é **p** e **Prod** é **p** então: **Ger** é **p** com grau 0.9,
Ger é **m** com grau 0.1.

Define-se, então, uma matriz de relação (b_{ij})

b_{11} b_{12} ... b_{1m}
b_{21} b_{22} ... b_{2m}
.....
b_{k1} b_{k2} b_{km}

O mecanismo de raciocínio consiste na implementação do seguinte algoritmo:

1) Dado um conjunto de valores das variáveis $X_i = a_i$, monta-se uma matriz que representa os antecedentes das regras(a_{ij}). Nessa matriz, estão representados os graus de pertinência de cada rótulo de cada uma das variáveis;

2) Determina-se, para cada regra, o valor mínimo de cada pertinência dos antecedentes (a_i);

3) Calculam-se os valores f_j como máximo em j de (a_1b_{1j}, a_2b_{2j},... a_kb_{kj}). Os valores f_j representam os graus de pertinência de cada um dos rótulos de um parâmetro de calibração;

4) Obtém-se o valor do parâmetro pelo processo de "desfuzzificação".

5.4.4 "Desfuzzificação" das Variáveis de Calibração

Como saída do raciocínio difuso tem-se a pertinência de cada um dos rótulos de cada variável lingüística. Os parâmetros do algoritmo são, por outro lado, valores crisp, sendo necessário, portanto, desfuzzificar os resultados.

O método adotado para realizar essa tarefa é baseado no centro de gravidade da distribuição de possibilidade (LEE, 1990). Nesse método, um ponto centróide é calculado para cada rótulo da variável lingüística, e sua função de pertinência é limitada em altura pelo valor de sua pertinência. A média dos centróides é calculada usando como peso a área abaixo de cada uma das funções de pertinência dos rótulos. O valor obtido é atribuído à variável que está sendo desfuzzificada.

6. Desenvolvimento Computacional do Algoritmo Genético Proposto e do Método de Calibração

6.1 Introdução

O algoritmo genético proposto foi implentado computacionalmente com o intuito de observar, principalmente, seu desempenho.

Além do algoritmo genético, rotinas para gerar problemas com determinadas características para testes, também foram desenvolvidas.

Durante toda a implementação, a utilização prática futura do software foi uma preocupação constante. O pacote aqui apresentado pode ser facilmente adaptado a situações específicas, dadas as suas características de modularidade, encapsulamento dos dados e rotinas, portabilidade etc.

No presente capítulo é apresentada essa implementaçao, assim como, são discutidos os seus fundamentos teóricos, e a possibilidade de futuras adaptações a outros ambientes diferentes dos escolhidos para testar o algoritmo.

6.2 Características do Sistema Computacional

Por se tratar de um algoritmo complexo, que pode ser adaptado a ambientes diferentes, foi utilizado como paradigma de implementação a modelagem orientada a objeto. Também, visando facilitar a utilização, foi implementada uma interface amigável tirando partido do ambiente de desenvolvimento e execução.

6.2.1 Programação Orientada a Objeto

A programação orientada a objeto é definida por BOOCH (1991) como um método no qual os programas são organizados como uma coleção de objetos. O estilo de programação orientada a objeto é adequada para aplicações nas quais a complexidade é um fator dominante.

Esse estilo de programação tem como base conceitual o modelo de objeto. Alguns dos benefícios práticos desse paradigma são:

- O modelo de objeto ajuda a explorar o poder das linguagens orientadas a objeto;

- O modelo de objeto estimula a reutilização de software e de análises já realizadas;

- O modelo de objeto permite ao sistema evoluir no tempo sem ser completamente replanejado frente a grandes mudanças nos requisitos;

- O modelo de objeto reduz os riscos e aumenta a confiança na análise realizada.

A evolução de um sistema combina os testes da codificação e a integração com outros softwares. O desenvolvimento através de protótipos que evoluem, estimula a avaliação de estruturas alternativas, o que na prática significa adicionar classes, alterar a implementação de classes e o redesenho de interfaces.

Os quatro elementos principais do modelo de objeto são:

- Abstração;
- Encapsulamento;
- Modularidade;
- Hierarquia.

Uma abstração permite mostrar as características essenciais de um objeto e portanto proporciona fronteiras conceituais bem definidas para o mesmo.

Por exemplo, no programa desenvolvido, um conjunto de tarefas representando produtos (ou lotes de produtos) a serem processados e estruturado como um objeto. Esse objeto que foi denominado carteira de ordens de produção, pode ser resolvido pelo algoritmo proposto, por métodos heurísticos, impresso, gravado em arquivos, etc. Da mesma forma a solução de um problema constitue também uma carteira de ordens, só que nesse caso, já programadas em todas as máquinas.

O encapsulamento esconde do resto do programa todas os detalhes do objetos que não contribuem para suas características essenciais. Por exemplo, no programa desenvolvido um cromossoma (solução) possui uma série de procedimentos que permitem calcular o valor da função objetivo do problema. Esses cálculos permanecem invisíveis para o resto do programa.

A modularidade permite decompor o sistema em módulos coerentes e fracamente ligados. Por exemplo, o programa desenvolvido possui um módulo especialista baseado em raciocínio difuso para calibrar o algoritmo genético. Esse módulo, fracamente ligado ao algoritmo é por outro lado totalmente coerente com o mesmo.

A hierarquização permite classificar as abstrações em nível decrescente. Por exemplo, no programa desenvolvido, a classe das ordens, anteriormente citada, é derivada de uma classe mais abstrata, a dos vetores ordenados.

6.2.2 Ambiente de Desenvolvimento

O ambiente de desenvolvimento é responsável por duas características importantes do sistema desenvolvido, a portabilidade e a comunicabilidade com o usuário.

A portabilidade permite que o sistema seja utilizado em hardwares diferentes do de desenvolvimento. Assim, uma maior facilidade de implementação de futuras aplicações práticas em outros ambientes encontra-se disponível.

Em relação a comunicabilidade com o usuário, o software deve permitir um processo de aprendizado rápido e eficiente que facilite a sua utilização. Além disso, o sistema deve fornecer aos usuários um grande número de informações gráficas que auxiliem a compreensão do desenrolar de uma aplicação. As funções do sistema devem estar ao alcance do usuário no menor número de passos possíveis.

O algoritmo genético proposto foi implementado utilizando como linguagem de programação o C++ da Borland Inc. Essa linguagem permite alta portabilidade entre diferentes plataformas, além de ser altamente concisa e produzir programas executáveis ágeis e rápidos. Como sistema operacional foi escolhido o sistema Windows 3.1 da MicroSoft. O ambiente Windows permite uma interface bastante amigável, baseada principalmente em informações gráficas. A migração para outras plataformas, também é facilitada, uma vez que suporta vários adaptadores de vídeo e impressão, e além disso, permite compartilhar recursos como memória as unidades de disco etc.

6.3 O Sistema Computacional Desenvolvido

O pacote desenvolvido foi estruturado utilizando o conceito de MDI (Multiple Document Interface) do ambiente Windows. O objeto MDI, constitue uma estrutura de dados e métodos que permite gerenciar vários objetos simultaneamente. Assim, o programa principal gerencia as diferentes janelas criadas. Na FIGURA 6.1, é mostrado um exemplo no qual várias janelas (gráfico, grafos etc.) encontram-se abertas. O MDI permite também o gerenciamento de janelas inativas e iconizadas.

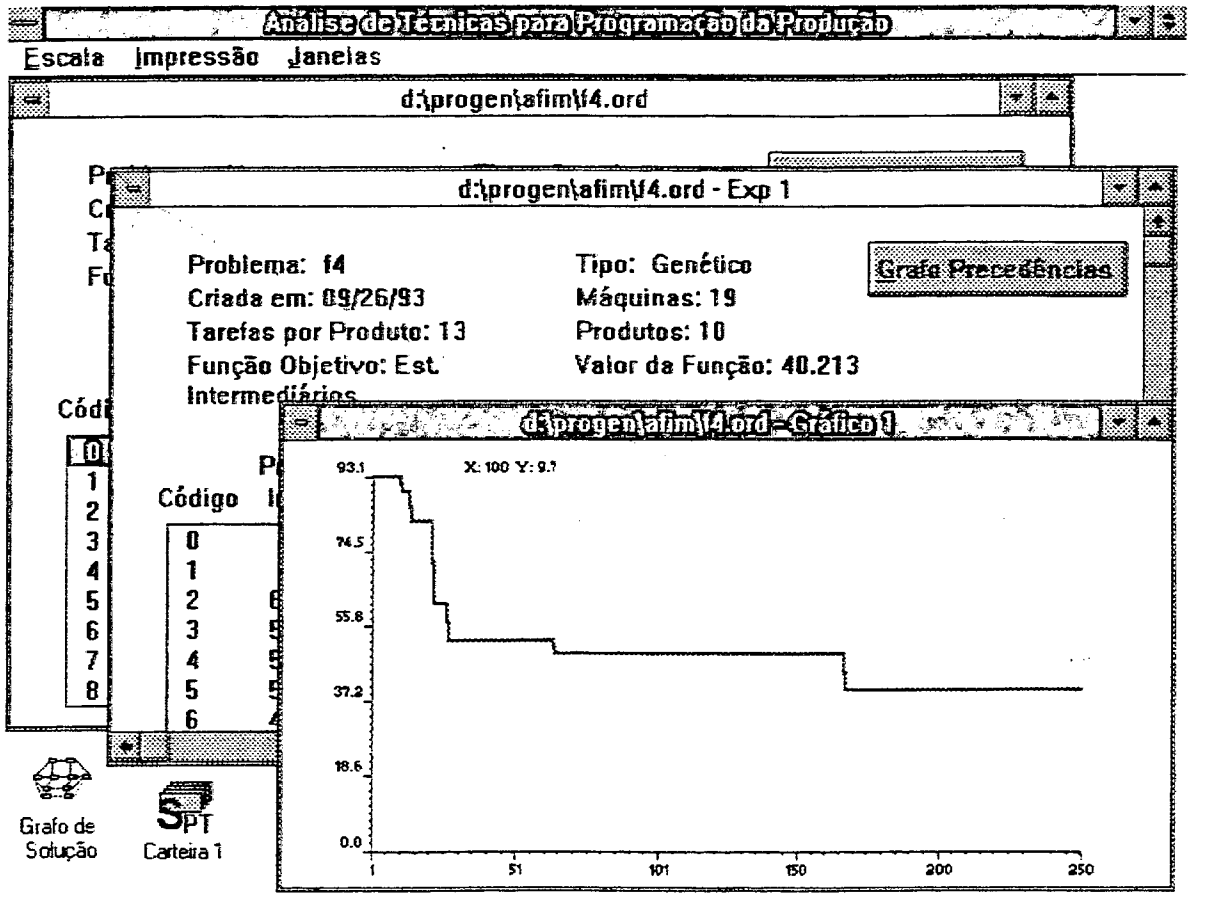


FIGURA 6.1 - Exemplo de Janelas Gerenciadas pelo Programa Principal

O conjunto de produtos (ou lotes de produtos) a serem programados foi modelado como um objeto denominado carteira de ordens. Esse objeto possui uma lista de todas as tarefas ou ordens de produção para cada máquina, além de informações como o número de produtos por máquinas, etc. A mesma estrutura é utilizada para armazenar cada uma das soluções do problema. Nesse caso, o tipo da função objetivo (critério de programação) e o método utilizados também são armazenados. Na FIGURA 6.2 e mostrada uma carteira a ser programada e uma solução obtida pela regra heurística SPT (menor tempo de processamento primeiro).

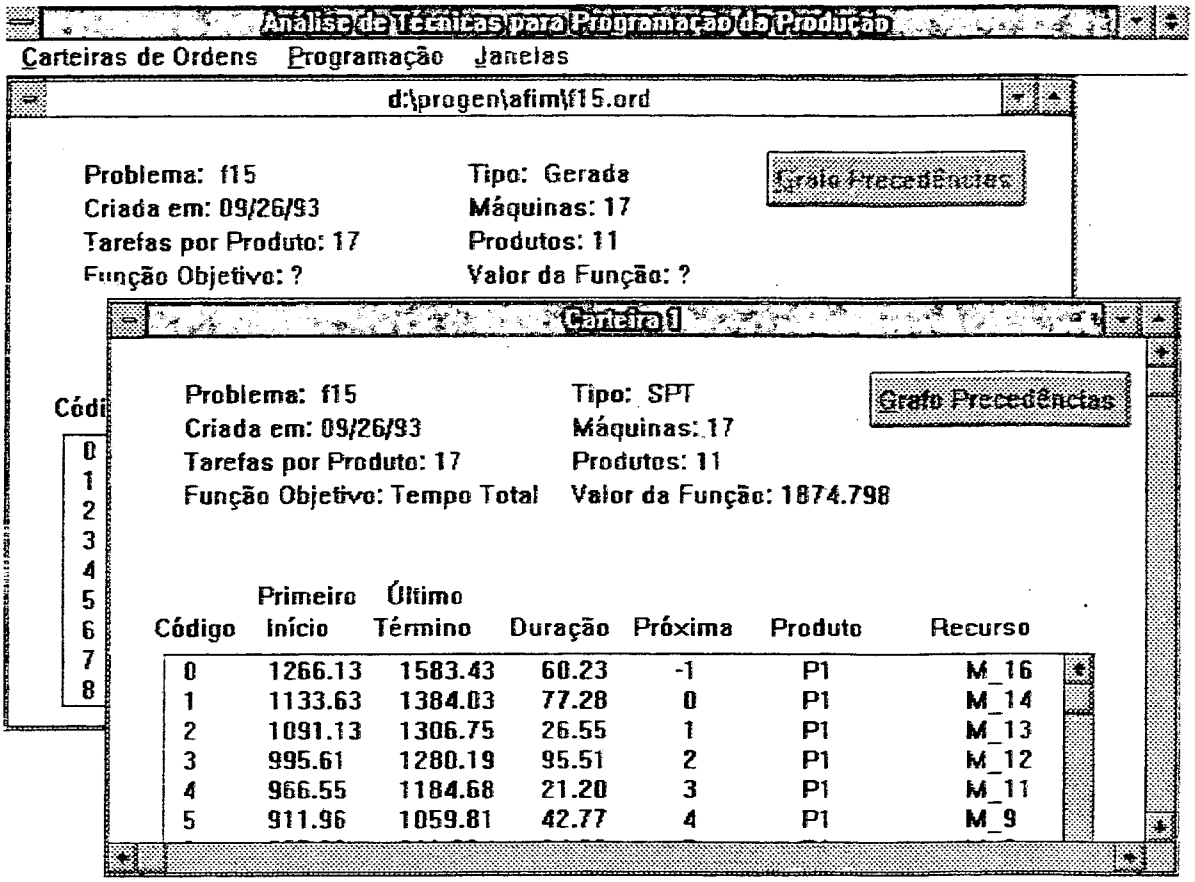


FIGURA 6.2 - Carteira a ser Programada e sua Solução (SPT)

As carteiras de ordens podem ser gravadas em arquivos (FIGURA 6.3), impressas (FIGURA 6.4) ou carregadas de arquivos anteriormente gravados FIGURA 6.5).

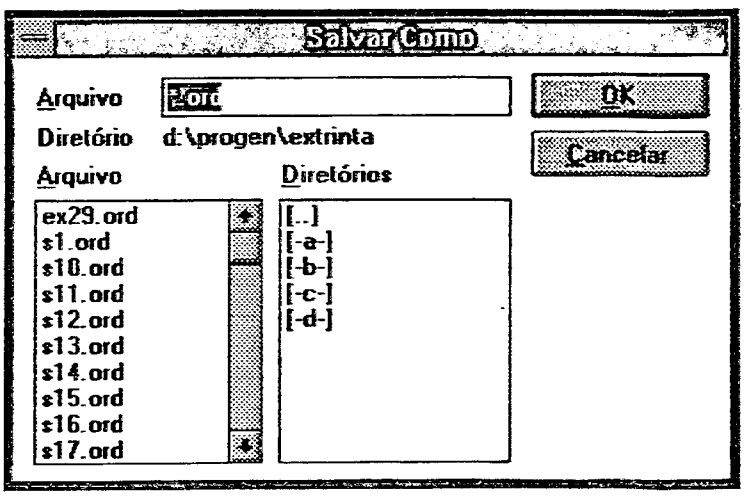


FIGURA 6.3 - Janela Utilizada para Salvar uma Carteira de Ordens

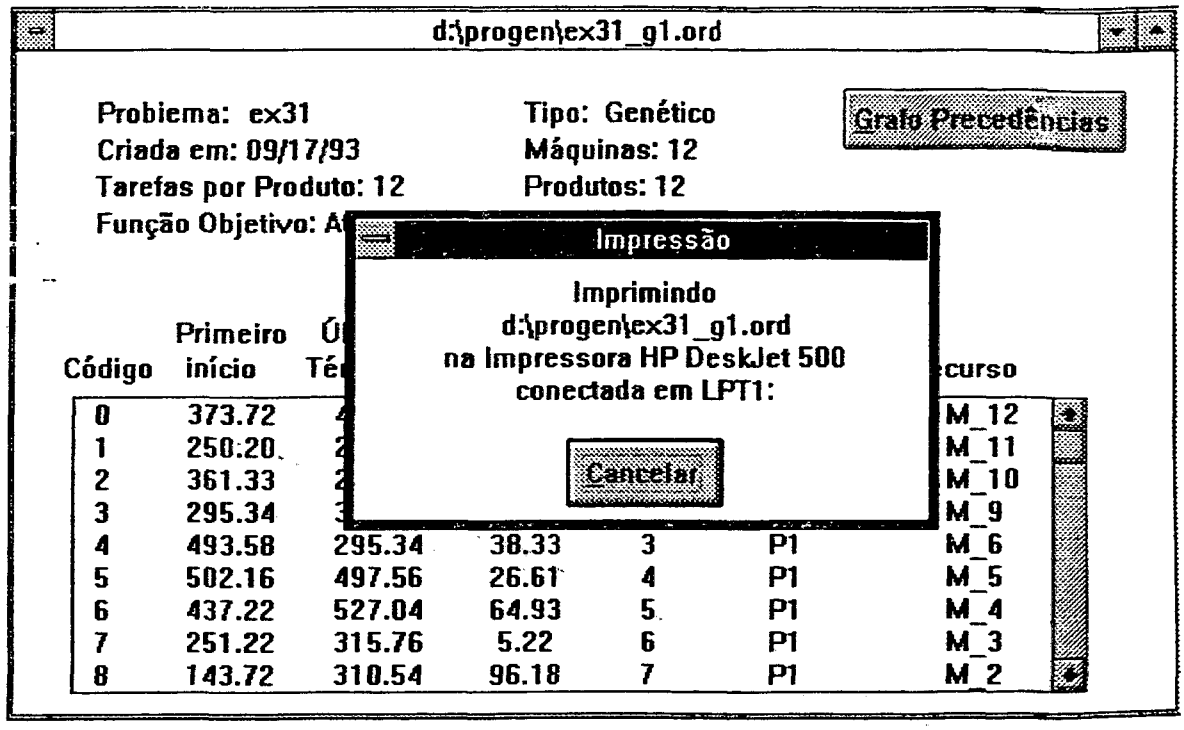


FIGURA 6.4 - Janela Mostrada durante a Impressão da Carteira de Ordens

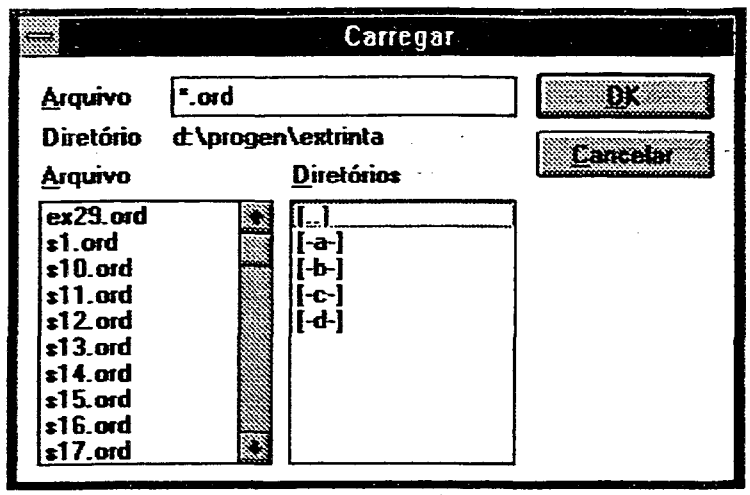


FIGURA 6.5 - Janela Utilizada para Carregar uma Carteira de Ordens

Informações contidas no conjunto de ordens podem ser alteradas, permitindo dessa forma realizar diversos testes com um conjunto de produtos a serem programados. (FIGURA 6.6).

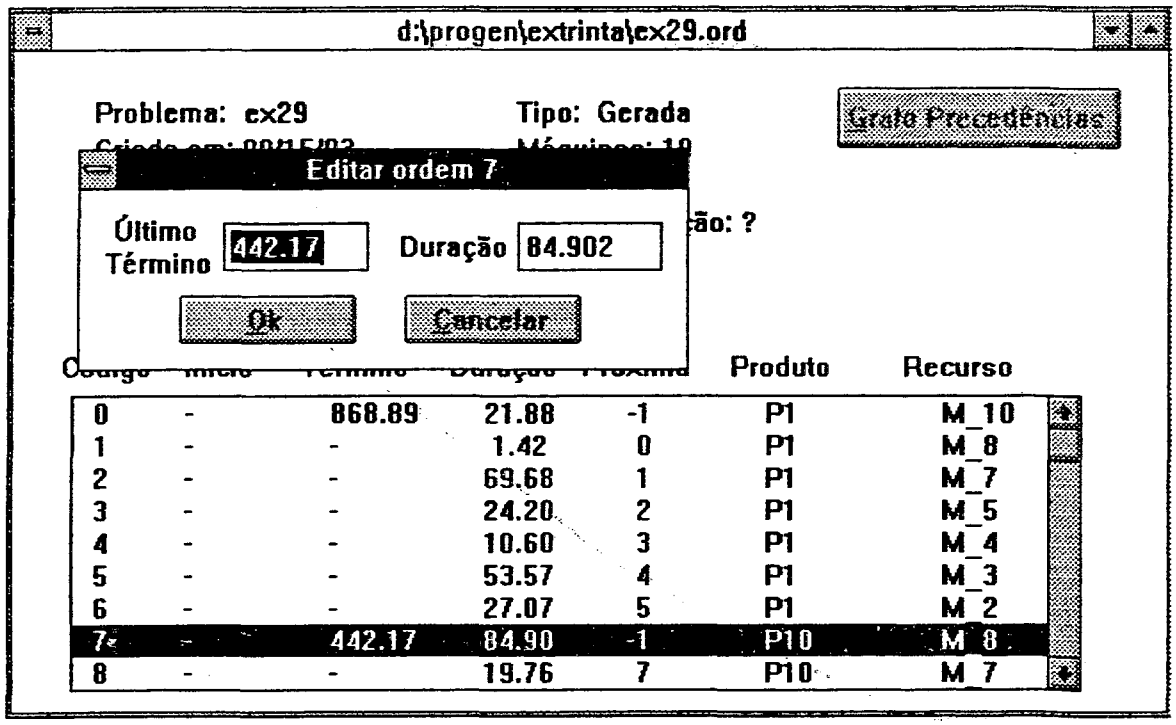


FIGURA 6.6 - Janela Mostrando um Conjunto de Produtos a serem Programados, Passíveis de Alteração

Também com o objetivo de testar o algoritmo proposto em diferentes situações e sem tendenciosidade, foi implementado um módulo que permite a geração de um conjunto de tarefas de acordo com determinados parâmetros. O módulo solicita ao usuário através de uma janela de diálogo esses parâmetros (FIGURA 6.7). Assim devem ser informados: o número de produtos diferentes a serem programados, o número de máquinas, informações sobre a data de entrega de cada produto e sobre as diferentes tarefas. A respeito das datas de entrega pode ser escolhida uma data única, para todos os produtos, ou pode-se gerar essa data aleatoriamente, com distribuição uniforme entre um valor mínimo e um valor máximo. O número de tarefas por produto, é também gerado com distribuição uniforme entre um mínimo e um máximo. A duração individual de cada tarefa é gerada aleatoriamente. Na geração dessa duração pode ser utilizada uma distribuição uniforme ou uma distribuição normal.

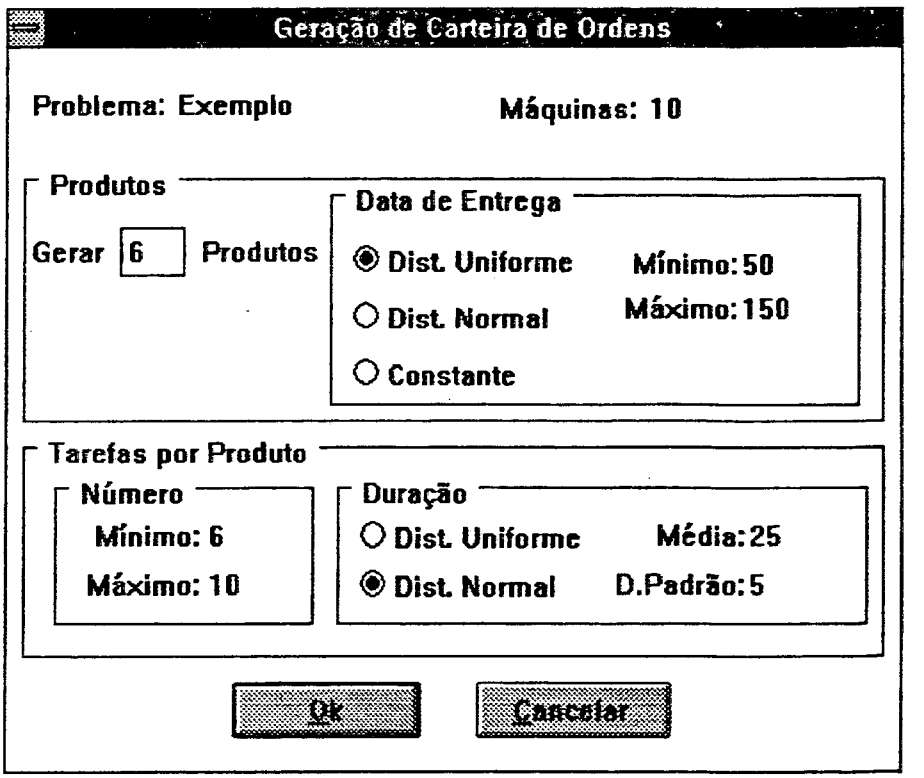


FIGURA 6.7 - Geração de Carteira de Ordens de acordo com os Parâmetros Estabelecidos

Os problemas podem ser resolvidos de acordo com diferentes critérios de programação. Encontram-se implementados o tempo total de processamento (makespan), o atraso médio, e o nível de estoques intermediários.

O usuário pode escolher programar o "flow shop" através de uma das regras heurísticas ou do algoritmo genético (FIGURA 6.8). No caso de regras heurísticas imediatamente após a escolha é solicitado qual a função objetivo.

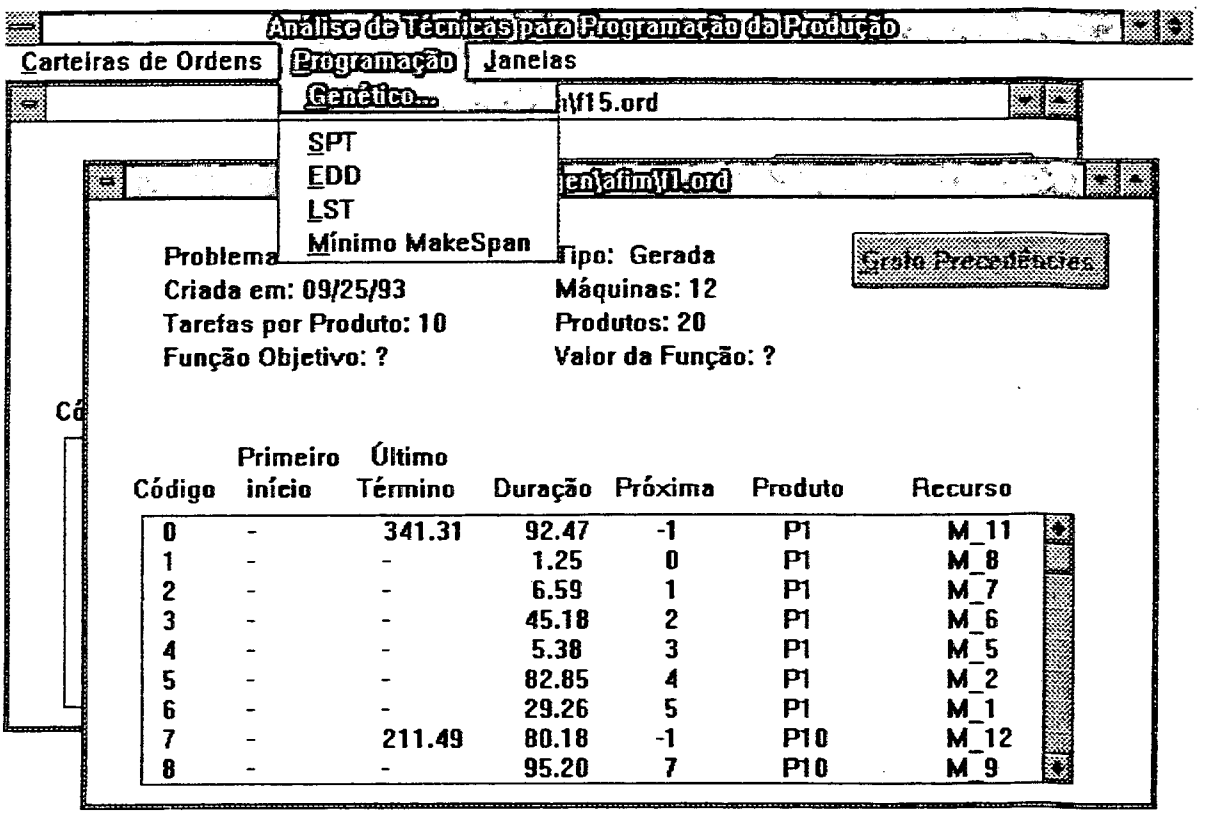


FIGURA 6.8 - Programação - Algoritmo Genético e Regras Heurísticas

No caso de utilização do algoritmo genético, informações adicionais são solicitadas (FIGURA 6.9.).

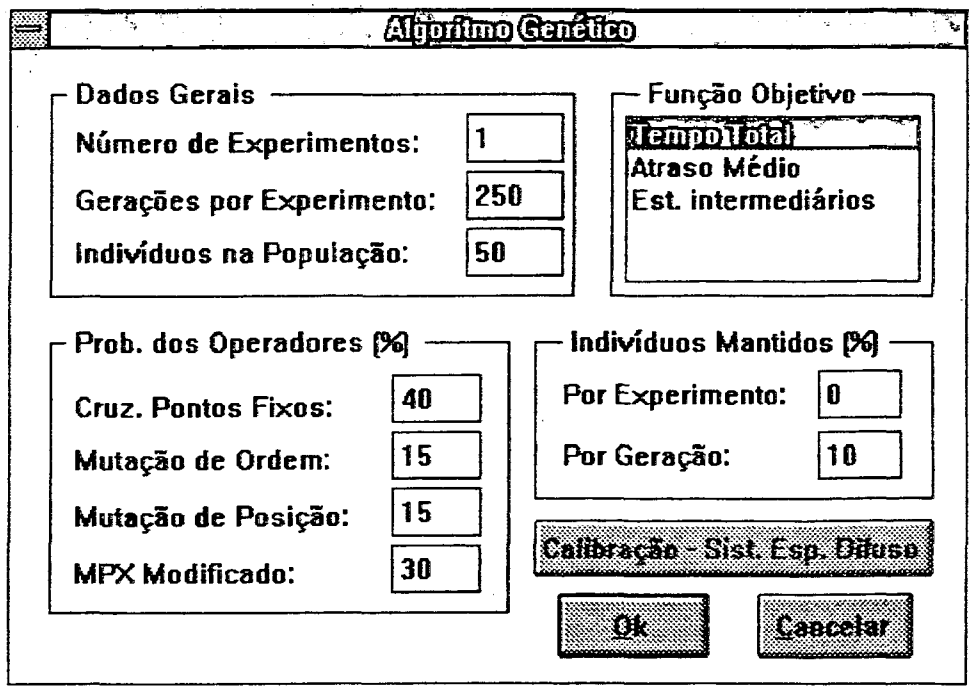


FIGURA 6.9 - Informações Adicionais Solicitadas - Algoritmo Genético

O usuário deve informar o número de experimentos a serem realizados. Cada experimento corresponde a uma execução do algoritmo genético. A implementação permite que uma parcela dos melhores cromossomas da população seja mantida de um experimento para o outro.

O número de gerações e de cromossomas na população, também deve ser informado. O algoritmo mantém pelo menos uma solução da geração anterior, podendo ser mantidas mais soluções.

Deve ser estabelecido, ainda, a probabilidade de escolha dos diferentes operadores que serão utilizados.

O algoritmo pode utilizar diferentes critérios para avaliar os cromossomas, dessa forma a função objetivo deve ser também escolhida.

Os parâmetros de número de cromossomas, total de gerações, probabilidade dos operadores podem ser sugeridos pelo sistema. Com essa finalidade aciona-se o algoritmo de calibração, e esse fornece os valores necessários para o usuário se guiar.

Durante o desenrolar da aplicação do algoritmo, os valores da função objetivo correspondentes a melhor solução, são mostrados graficamente (FIGURAS 6.10 e 6.11).

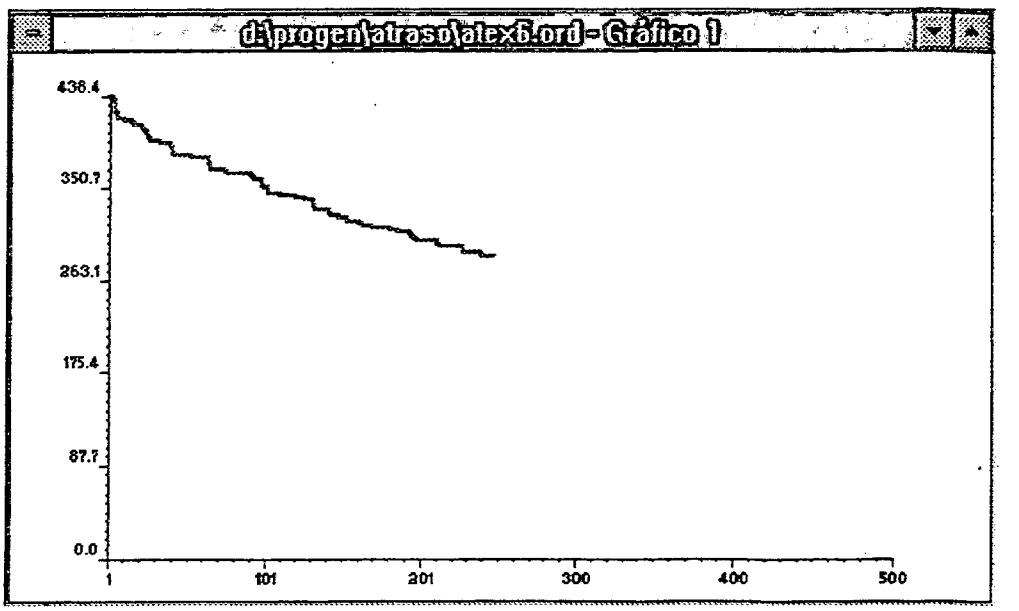


FIGURA 6.10 - Evolução dos Valores da Função Objetivo

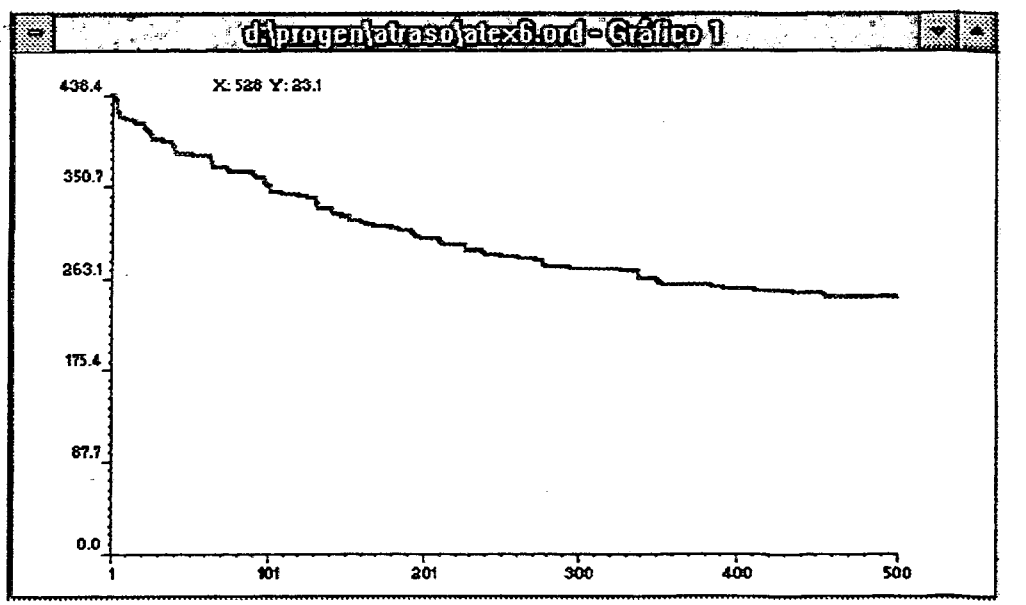
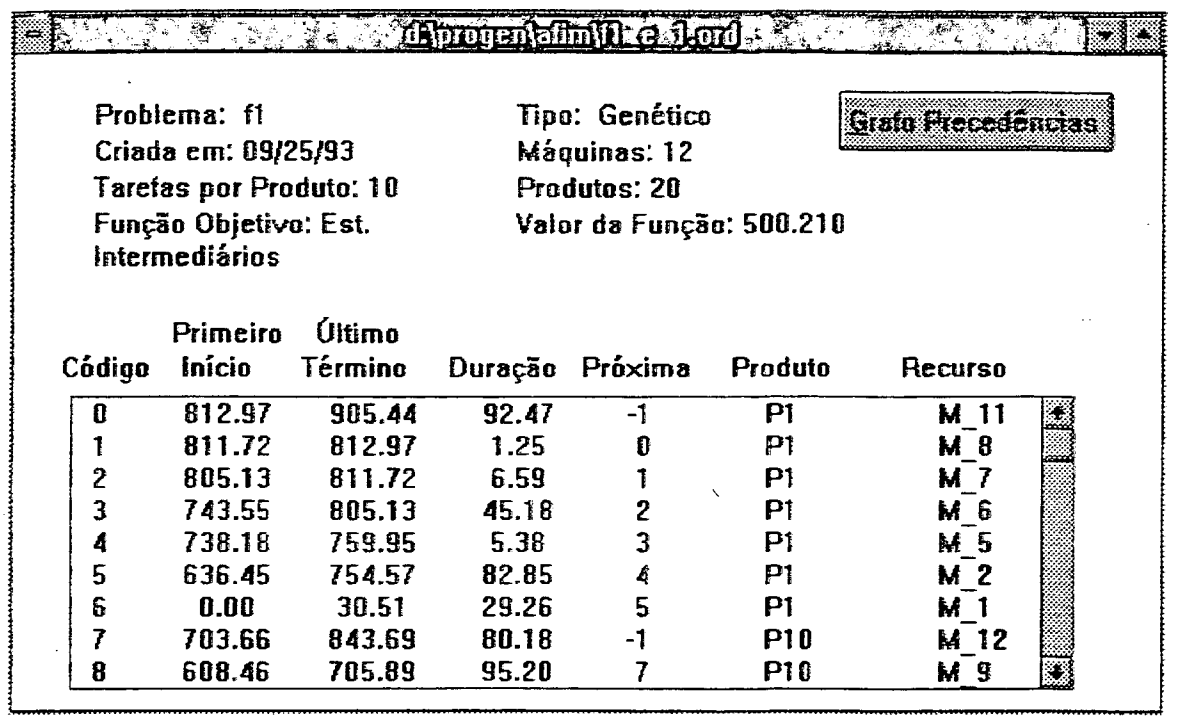


FIGURA 6.11 - Valores da Função Objetivo Correspondentes as Melhores Soluções

Finalizada a aplicação do algoritmo, uma janela é criada, com as informações da melhor solução encontrada (FIGURA 6.12). São apresentados, o

nome do problema com suas principais informações, o tipo e o valor da função objetivo, e as diferentes tarefas programadas. Para cada tarefa é fornecida a data mais cedo de início e a data mais tarde de término, na solução. O gráfico com a evolução do valor da função objetivo a cada geração permanece, podendo ser impresso. A escala do gráfico pode ser alterada, ou dado um "zoom" em determinada área do mesmo. A solução pode ser gravada em disco ou impressa.



Código	Primeiro Início	Último Término	Duração	Próxima	Produto	Recurso
0	812.97	905.44	92.47	-1	P1	M_11
1	811.72	812.97	1.25	0	P1	M_8
2	805.13	811.72	6.59	1	P1	M_7
3	743.55	805.13	45.18	2	P1	M_6
4	738.18	759.95	5.38	3	P1	M_5
5	636.45	754.57	82.85	4	P1	M_2
6	0.00	30.51	29.26	5	P1	M_1
7	703.66	843.69	80.18	-1	P10	M_12
8	608.46	705.89	95.20	7	P10	M_9

FIGURA 6.12 - Informações da Melhor Solução

Encontra-se, também, disponível o grafo de precedências das tarefas na solução (FIGURA 6.13). Nesse grafo, os arcos horizontais representam as precedências na estrutura do produto, e os demais, as precedências nas máquinas.

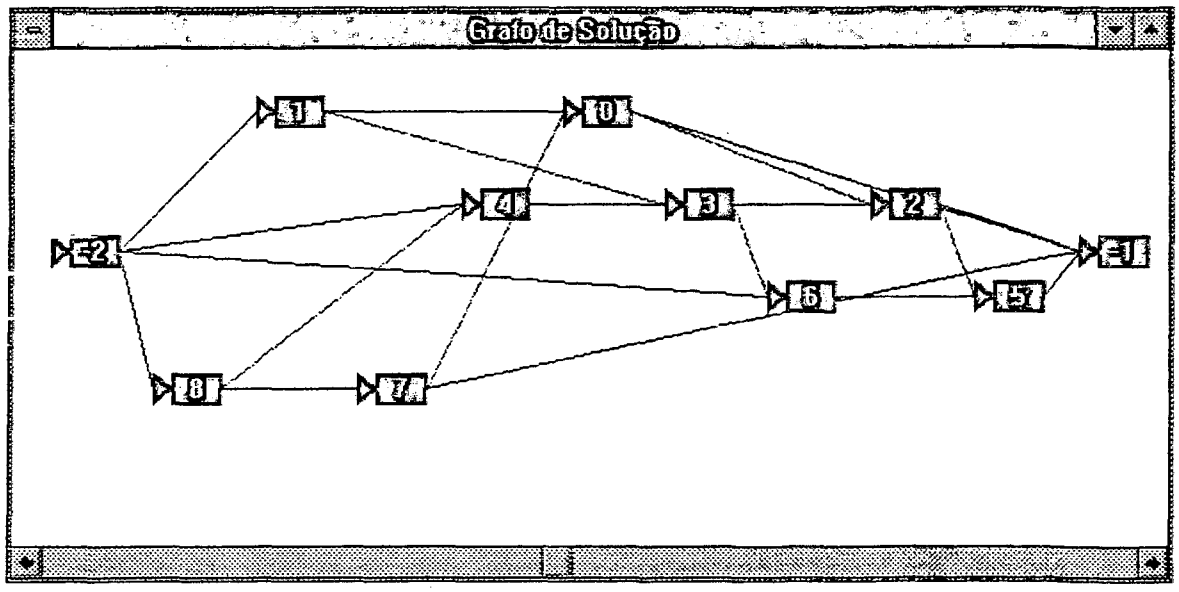


FIGURA 6.13 - Grafo de Precedência das Tarefas

6.4 Conclusão

O sistema computacional desenvolvido incorpora os princípios da programação orientada à objetos, sendo de fácil adaptação a diferentes ambientes, dado as suas características de abstração, encapsulamento, modularidade e hierarquia.

A implementação permite gerar problemas aleatoriamente, para serem utilizados em testes e/ou experimentos.

Na aplicação do algortimo proposto, os diferentes parâmetros são facilmente alterados, podendo-se observar suas influências no desempenho do algortimo.

O ambiente WINDOWS 3.1 utilizado para o desenvolvimento e execução torna extremamente rápido a familiarização do usuário com o sistema, além de proporcionar uma interface amigável.

No próximo capítulo, experimentos realizados utilizando o sistema desenvolvido são apresentados e discutidos.

7. RESULTADOS E DISCUSSÕES

7.1 Introdução

Analisar as aplicações de algoritmos genéticos em problemas de produção constitui-se em uma tarefa extremamente complexa. Diversos fatores conduzem a uma combinação exponencial de testes possíveis. Neste capítulo, uma série de experimentos são relatados. Procurou-se elaborar um leque de aplicações e análises que proporcionassem indicativos sobre o desempenho do algoritmo proposto para solução de problemas relacionados à programação da produção.

O capítulo está estruturado da seguinte forma: o ítem 7.2 apresenta uma discussão a respeito da complexidade da avaliação de algoritmos genéticos em problemas de programação da produção. O ítem 7.3 descreve os experimentos elaborados para a realização dos testes. A seguir, os resultados obtidos são discutidos e analisados. Posteriormente, é realizada uma aferição com relação ao desempenho do Sistema Especialista Difuso desenvolvido para a calibração do algoritmo proposto no trabalho.

7.2 Complexidade da Avaliação do Algoritmo Proposto

O modelo proposto tem como finalidade programar a produção em um ambiente "flow shop". Testar a efetividade do algoritmo genético em vários ambientes "flow shop" e em várias condições, dentro de um mesmo "flow shop", é uma tarefa extremamente complexa.

Num ambiente de produção do tipo "flow shop", com referência à programação da produção muitos são os atributos que o definem. Por exemplo:

- número de produtos
- número de máquinas;

- número de tarefas por produto;
- designação da data de entrega de cada produto;
- capacidade das máquinas.

Com referência ao número de produtos (ou lotes de produtos), a complexidade de programar 10 produtos ou lotes é totalmente diferente de fazê-lo com 100 produtos. O número de soluções possíveis, no caso de uma única máquina, é n!, onde n é o número de produtos. Testes podem ser feitos com referência a diferentes números de produtos. Na literatura sobre problemas de seqüenciamento em "flow shop", os números de produtos utilizados em testes variam de 4 a 20 (WIDMER e HERTZ, 1989), 3 a 50 (DANNENBERG, 1977), 3 a 30 (PARK, PEGDEN e ENSCORE., 1984). Observa-se que problemas com mais de 20 produtos são considerados grandes.

O número de máquinas, do mesmo modo, apresenta uma relação imediata sobre o espaço de soluções possíveis. Como já foi colocado anteriormente, existem, no caso da minimização do tempo total de processamento, $(n!)^{m-2}$ soluções diferentes (CONWAY, 1967). Os trabalhos de avaliação de técnicas na área realizam testes com: 4 a 20 (WIDMER e HERTZ, 1989), 3 a 50 (DANNENBERG, 1977), 4 a 20 (PARK, PEGDEN e ENSCORE., 1984) máquinas. Problemas envolvendo de 20 a 50 máquinas são considerados grandes.

Na maioria das pesquisas, é considerado que o número de tarefas por produto é igual ao número de máquinas. Contudo, quando o número de tarefas por produto é variável, obviamente, tal variabilidade implica no acréscimo da complexidade dos testes a serem realizados.

O atraso médio depende tanto da data de término do processamento de cada produto quanto da sua data de entrega. Assim, os testes de eficiência do algoritmo devem levar em consideração a forma de atribuição dessas datas. Existem diferentes metodologias para atribuir as datas de entrega. Dentre elas, pode-se citar: data de entrega proporcional ao tempo de processamento, data de entrega proporcional ao número de tarefas, data de entrega constante e data de entrega aleatória.

Com relação à capacidade das máquinas, nota-se que a capacidade de processamento influencia na duração de cada tarefa. Estudando o

desempenho de heurísticas. PARK, PEGDEN e ENSCORE (1984) utilizam uma geração aleatória com distribuição de Erlang. DANNENBERG (1977) utiliza uma distribuição de probabilidade uniforme (0;99), para gerar randomicamente durações inteiras. Já WIDMER e HERTZ (1989) utilizam tempos de processamento distribuídos no intervalo (0;10).

Além das características do ambiente, critérios de programação também devem ser considerados. Testes devem ser realizados com diferentes funções objetivo: tempo total de processamento de todos os produtos, atraso médio e nível de estoques intermediários, por exemplo.

Algoritmos genéticos em geral, e o algoritmo proposto em particular, devem ser calibrados quanto aos seus parâmetros. Influenciam no desempenho do algoritmo o número de cromossomas, o número de gerações, probabilidades de aplicação de diferentes operadores de reprodução e o percentual de elitismo. Observa-se que existe um compromisso entre o número de cromossomas e o total de gerações, conseqüentemente acarretando num tamanho de população "ótimo" para cada problema.

Os algoritmos genéticos não são determinísticos. Portanto, existe a necessidade de executar o algoritmo proposto mais de uma vez. Assim, as próprias soluções obtidas na aplicação do algoritmo genético exigem tratamento adequado.

Das considerações anteriores, pode ser observado que o número de combinações para testar o algoritmo genético proposto nos diferentes ambientes "flow shop" é extremamente elevado.

Os experimentos a seguir desenvolvidos não são completos, porém os resultados proporcionam um indicativo do desempenho do algoritmo proposto. Um estudo da eficiência do algoritmo constitui uma extensão importante desta pesquisa.

7.3 Planejamento dos Experimentos

Os experimentos foram planejados com o objetivo de observar a variabilidade dos resultados para um dado problema, observar o comportamento do algoritmo em um conjunto de problemas com diferentes características e, ainda, observar o comportamento frente a diferentes critérios de programação.

Um conjunto de trinta problemas, com as seguintes características, foi gerado aleatóriamente:

- Número de produtos: entre 10 e 30;
- Número de máquinas: entre 10 e 30;
- Número de tarefas por produto:
  - Máximo: entre 1 e número de máquinas
  - Mínimo: entre 1 e Máximo
- Duração das tarefas: Uniforme (1;100);
- Data de entrega de cada produto: Uniforme (Mínimo*50; Máximo*50);

Cada um dos problemas foi processado três vezes para cada uma das funcões objetivo consideradas. Desse modo, uma amostra com 270 observações foi gerada.

A TABELA 7.1 descreve o conjunto de problemas utilizados para testar o algoritmo proposto.

NÚMERO PROBLEMA	NÚMERO MÁQUINAS	NÚMERO PRODUTOS	TAREFAS POR PRODUTO		TOTAL TAREFAS
			MÍNIMO	MÁXIMO	
1	12	20	4	10	112
2	20	12	8	18	153
3	23	19	10	20	268
4	19	10	1	13	82
5	18	26	5	17	267
6	27	31	1	12	230
7	12	17	4	9	107
8	11	29	3	5	117
9	17	11	17	17	187
10	13	17	13	13	221
11	30	28	1	7	117
12	12	13	9	10	123
13	20	20	3	13	189
14	21	24	4	17	221
15	17	11	9	17	183
16	17	14	17	17	238
17	26	10	19	21	201
18	19	11	1	17	93
19	28	26	18	21	507
20	28	11	13	13	143
21	11	27	7	9	215
22	23	12	6	22	141
23	19	23	4	5	101
24	14	24	8	11	235
25	16	10	3	8	60
26	23	25	10	13	285
27	23	15	8	17	173
28	16	20	9	11	202
29	13	27	7	10	228
30	10	19	10	10	190

TABELA 7.1 - Dados dos Problemas Teste Gerados

A TABELA 7.2 apresenta a relação tarefas por máquina para cada um dos problemas-teste.

NÚMERO PROBLEMA	TAREFAS/ MÁQUINAS	NÚMERO PROBLEMA	TAREFAS/ MÁQUINAS
1	9.33	16	14.00
2	7.65	17	7.73
3	11.65	18	4.89
4	4.32	19	18.11
5	14.83	20	5.11
6	8.52	21	19.55
7	8.92	22	6.13
8	10.64	23	5.32
9	11.00	24	16.79
10	17.00	25	3.75
11	3.90	26	12.39
12	10.25	27	7.52
13	9.45	28	12.63
14	10.52	29	17.54
15	10.76	30	19.00

TABELA 7.2. Relação Tarefas por Máquina de Cada Problema Gerado

7.4 Análise dos Resultados Obtidos

As análises realizadas englobam três aspectos: em uma primeira etapa, os resultados obtidos foram comparados com os melhores resultados verificados na aplicação das heurísticas implementadas. Posteriormente, eles foram avaliados considerando-se uma tipificação dos problemas em termos de suas complexidades. Por fim, foi realizado um procedimento de análise de variância, para identificar a influência dos fatores complexidade do problema e tipo de função objetivo na melhoria obtida pelo algoritmo proposto.

7.4.1 Resultados do Algoritmo Proposto x Melhores Resultados das Regras Heurísticas

Nesta análise, os resultados obtidos com o algoritmo proposto foram comparados com os melhores resultados verificados na aplicação das heurísticas implementadas. O critério de comparação adotado foi:

$$\text{Melhoria da Função Objetivo: } 100\frac{F_h - F_g}{F_h}$$

onde: F_g = o valor da função objetivo na solução encontrada pelo algoritmo genético

F_h = melhor valor obtido através das heurísticas para a função objetivo.

Para a função objetivo Tempo Total de Processamento (TABELA 7.3), as melhorias alcançadas pelo algoritmo proposto atingiram 6.04%, apresentando um valor médio de 1.82 %. TAILLARD (1990), estudando todas as soluções de um problema flow-shop com respeito ao tempo total de processamento, observa que o conjunto das soluções possíveis desse problema segue uma distribuição tendendo a uma normal, com média 1.2 vezes o valor ótimo. Conseqüentemente, pode-se inferir que uma solução aleatória tem probabilidade igual a 0.5 de estar 20% acima da solução ótima. Uma heurística eficiente, certamente, deverá produzir resultados abaixo desse valor médio. Portanto, melhorias em soluções obtidas por processos heurísticos eficientes são extremamente difíceis de serem alcançadas, na minimização do tempo total de processamento.

NÚMERO PROBLEMA	MELHOR HEURÍSTICA	MELHOR GENÉTICO	MELHORIA DA HEURÍSTICA	NUMERO PROBLEMA	MELHOR HEURÍSTICA	MELHOR GENÉTICO	MELHORIA DA HEURÍSTICA
1	919.07	894.26	2.70	16	1988.43	1988.43	0.00
2	1370.24	1348.65	1.58	17	1687.47	1617.93	4.12
3	1788.20	1787.41	0.04	18	932.58	893.98	4.14
4	827.58	827.58	0.00	19	2636.05	2636.04	0.00
5	1464.68	1436.24	1.94	20	1075.21	1026.99	4.48
6	1177.13	1160.20	1.44	21	1733.51	1702.22	1.81
7	953.21	953.21	0.00	22	1332.34	1262.17	5.27
8	993.08	945.58	4.78	23	644.17	605.28	6.04
9	1928.39	1928.39	0.00	24	1706.83	1701.46	0.31
10	1817.35	1817.35	0.00	25	560.77	531.16	5.28
11	904.36	904.36	0.00	26	1680.67	1659.51	1.26
12	1231.94	1193.13	3.15	27	1116.79	1116.79	0.00
13	1533.25	1509.94	1.52	28	1592.35	1592.35	0.00
14	1494.49	1440.59	3.61	29	1626.19	1626.19	0.00
15	1874.80	1854.50	1.08	30	1736.72	1736.72	0.00
						MÉDIA	1.82

TABELA 7.3 - Soluções Encontradas para a Função Objetivo Tempo Total de Processamento

Para a função atraso médio, o algoritmo proposto obteve ganhos substanciais em relação às heurísticas. Os resultados do algoritmo variam no intervalo (1.2% ; 25.47%) e apresentam como valor médio 10.03% (TABELA 7.4).

NÚMERO PROBLEMA	MELHOR HEURÍSTICA	MELHOR GENÉTICO	MELHORIA DA HEURÍSTICA	NÚMERO PROBLEMA	MELHOR HEURÍSTICA	MELHOR GENÉTICO	MELHORIA DA HEURÍSTICA
1	374.85	331.68	11.52	16	800.90	791.30	1.20
2	447.89	420.92	6.02	17	480.09	470.40	2.02
3	705.81	659.77	6.52	18	443.34	359.16	18.99
4	267.48	235.33	12.02	19	1201.48	1114.50	7.24
5	491.43	438.57	10.76	20	340.72	268.56	21.18
6	513.18	424.25	17.33	21	809.56	760.77	6.03
7	304.71	280.84	7.83	22	448.06	381.80	14.79
8	408.39	364.28	10.80	23	290.45	216.96	25.30
9	705.13	694.86	1.46	24	767.78	693.80	9.64
10	796.97	780.78	2.03	25	216.92	161.67	25.47
11	572.38	431.44	24.62	26	812.37	767.04	5.58
12	557.37	538.30	3.42	27	398.02	357.17	10.26
13	580.99	556.81	4.16	28	722.30	669.18	7.35
14	612.05	532.95	12.92	29	664.91	618.39	7.00
15	948.06	894.62	5.64	30	686.23	674.83	1.66
						MÉDIA	10.03

TABELA 7.4 - Soluções Encontradas para a Função Objetivo Atraso Médio

Os melhores resultados foram alcançados com relação à minimização dos estoques intermediários, (Tabela 7.5). Os melhorias neste critério variaram de 6.89 % a 58.41% com um valor médio de 22.23 %.

NÚMERO PROBLEMA	MELHOR HEURÍSTICA	MELHOR GENÉTICO	MELHORIA DA HEURÍSTICA	NÚMERO PROBLEMA	MELHOR HEURÍSTICA	MELHOR GENÉTICO	MELHORIA DA HEURÍSTICA
1	566.52	494.44	12.72	16	458.39	376.95	17.77
2	162.27	129.24	20.35	17	184.15	152.61	17.13
3	423.12	376.81	10.94	18	146.97	62.24	57.65
4	96.69	40.21	58.41	19	877.56	817.13	6.89
5	607.95	533.14	12.31	20	216.14	182.00	15.80
6	642.03	526.38	18.01	21	1165.14	918.26	21.19
7	391.85	282.27	27.96	22	231.46	166.03	28.27
8	1062.18	753.15	29.09	23	607.58	377.14	37.93
9	259.91	215.52	17.08	24	897.02	807.34	10.00
10	579.42	509.91	12.00	25	135.09	63.07	53.31
11	266.74	133.81	49.84	26	666.40	617.25	7.38
12	324.46	274.58	15.37	27	301.73	243.06	19.44
13	483.26	354.06	26.74	28	655.97	592.56	9.67
14	619.44	485.71	21.59	29	940.40	854.14	9.17
15	259.62	233.65	10.00	30	670.43	583.64	12.95
						MÉDIA	22.23

TABELA 7.5 - Soluções Encontradas para a Função Objetivo Estoques Intermediários

7.4.2 Complexidade dos Problemas X Resultados Obtidos

Para analisar os resultados obtidos, em termos da complexidade dos problemas gerados, foi realizada uma análise de agrupamentos, com o objetivo de identificar padrões de problemas. Para a realização dessa análise, as seguintes variáveis foram consideradas: número de máquinas, número de produtos e número total de tarefas dividido pelo número de máquinas. A razão da escolha dessas variáveis para efeitos de classificação em grupos se justifica pelo fato de serem indicativos da complexidade do problema em análise. O processo de agrupamento utilizado foi o das K-médias com distância euclidiana. A FIGURA 7.1 e a TABELA 7.6 apresentam os cinco grupos resultantes.

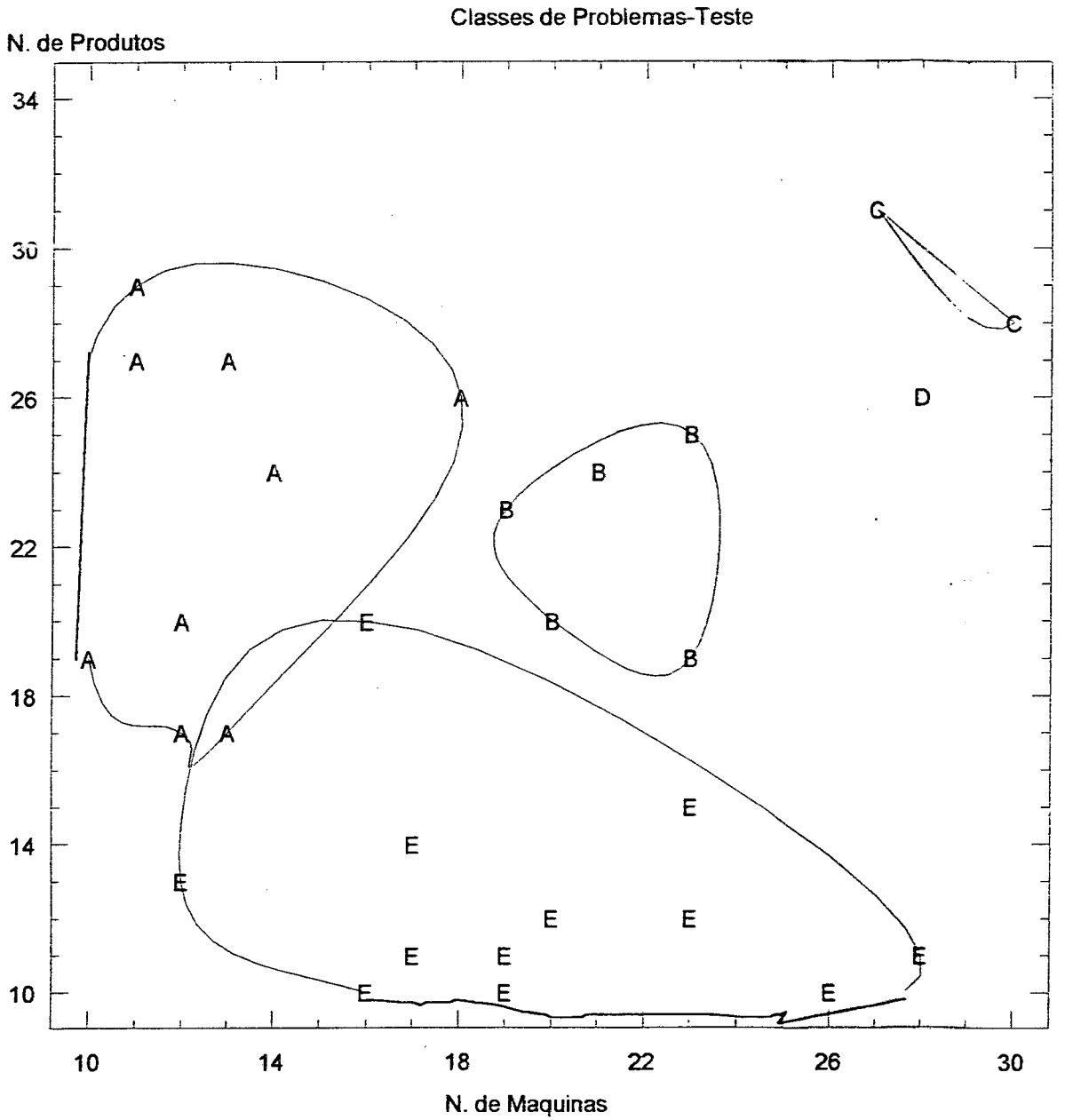


FIGURA 7.1 - Análise de Agrupamentos - Problemas-Teste

Grupo	Problemas
1	1,5,7,8,10,21,24,29,30
2	3,13,14,23,26
3	6,11
4	2,4,9,12,15,16,17,18,20,22,25,27,28
5	19

TABELA 7.6 - Agrupamento dos Problemas-Teste

Em termos de complexidade dos problemas, os grupos apresentam-se conforme a TABELA 7.7.

Grupo de	Máquinas		Produtos		Tarefas/Máquinas	
Problemas	Mínimo	Máximo	Mínimo	Máximo	Mínimo	Máximo
1	11	18	17	29	8.92	19.55
2	12	28	10	20	3.75	14.0
3	19	23	19	25	6.13	12.39
4	27	30	28	31	3.90	8.52
5	28	28	26	26	18.11	18.11

TABELA 7.7 - Complexidade dos Problemas por Grupo

O grupo cinco apresenta um elevado número de máquinas, produtos e tarefas por máquina. O grupo quatro contém menor número de produtos e uma baixa relação tarefas por máquina. O grupo três, ao contrário do grupo cinco, apresenta elevado número de máquinas e produtos, porém a relação tarefas/máquina é baixa. O grupo dois, em relação aos demais grupos, pode ser considerado um grupo médio. Finalmente, o grupo um apresenta valores médios, contudo, com número de máquinas menor do que o observado no grupo dois.

Melhorias médias em relação às heurísticas, verificadas para cada tipo de função objetivo, são apresentadas na TABELA 7.8.

Grupo de Problemas	Função Objetivo		
	Tempo Total de Processamento	Atraso Médio	Estoques Intermediários
1	1.28	7.47	16.38
2	2.49	10.89	20.91
3	0.72	20.98	33.93
4	2.24	9.96	26.17
5	0.001	7.24	6.98

TABELA 7.8 - Melhorias Médias por Grupo de Problemas em relação às Heurísticas

7.4.3 Efeitos Complexidade do Problema e Tipo de Função Objetivo nos Resultados do Algoritmo Proposto

Para avaliar os efeitos da complexidade do problema e do tipo de função objetivo utilizada nas soluções obtidas pelo algoritmo implementado, foi realizada uma análise de variância. A TABELA 7.9 apresenta esses resultados.se.

Fonte de Variação	Soma dos Quadrados	Graus de Liberdade	Quadrado Médio	Valor de F	Nível de Significância
Efeitos Principais					
A:Função Objetivo	6230.70	2	3115.35	47.793	0.0000
B: Tipo de Problema	2071.68	4	517.92	7.945	0.0000
Interações					
AB	1772.56	8	221.57	3.399	0.0010
Residual	16622.09	255	65.18		
Total (corrigida)	36178.93	269			

TABELA 7.9 - Quadro de Análise de Variância - Melhorias em Relação às Heurísticas

O exame da TABELA 7.9 revela que tanto os efeitos principais quanto o efeito interação são significativos. Esse fato leva à conclusão que as melhorias alcançadas pelo algoritmo proposto diferem, significativamente, em relação ao tipo de função objetivo, à complexidade do problema e à interação entre esses dois fatores.

A TABELA 7.10 apresenta intervalos de confiança de 95% para melhorias médias obtidas para, respectivamente, função objetivo, tipo de problema e interações entre os fatores.

Nível de Análise	Número de Problemas	Melhoria Média da Heurística	Erro Padrão	Intervalo de Confiança de 95% para a Média	
				Limite Inferior	Limite Superior
A:Função Objetivo					
Tempo Total (TT)	90	1.06	1.28	-1.46	3.58
Atraso Médio (AM)	90	9.21	1.28	6.68	11.73
Est. Intermed (EI)	90	18.75	1.28	16.23	21.28
B: Grupos de Problemas					
Grupo 1 (A)	81	7.119	0.897	5.352	8.886
Grupo 2 (B)	45	9.695	1.203	7.324	12.066
Grupo 3 (C)	18	17.001	1.903	13.252	20.749
Grupo 4 (D)	117	11.091	0.746	9.620	12.561
Grupo 5 (E)	9	3.467	2.691	-1.834	8.768
Interações (AB)					
TT x A	27	1.124	1.554	-1.936	4.185
TT x B	15	1.988	2.085	-2.118	6.094
TT x C	6	0.677	3.296	-5.815	7.169
TT x D	39	1.507	1.293	-1.040	4.053
TT x E	3	0.004	4.661	-9.181	9.182
AM x A	27	6.120	1.554	3.060	9.181
AM x B	15	9.294	2.085	5.188	13.400
AM x C	6	18.025	3.296	11.533	24.518
AM x D	39	7.998	1.293	5.451	10.545
AM x E	3	4.621	4.661	-4.561	13.802
EI x A	27	14.113	1.554	11.053	17.174
EI x B	15	17.803	2.085	13.696	21.909
EI x C	6	32.300	3.296	25.807	39.792
EI x D	39	23.767	1.293	21.220	26.314
EI x E	3	5.781	4.661	-3.401	14.962
Geral	270	9.67	0.74	8.22	11.13

TABELA 7.10 - Intervalos de Confiança de 95% para Melhorias Médias da Heurística

Como resultado mais geral, pode-se afirmar, com 95% de confiança, que o intervalo (8.22;11.13) contém a melhoria média proporcionada pelo algoritmo desenvolvido. Para problemas do grupo 3, com 95% de confiança o intervalo (13,25; 20.75) contém o parâmetro relativo à melhoria média conseguida com a aplicação do algoritmo em relação à melhor heurística utilizada na solução desses problemas.

Foi realizado ainda um teste de contrastes, do tipo LSD ("Least Significant Difference"), para a análise de variância realizada. Esse teste foi utilizado para determinar a origem das diferenças quanto à melhoria proporcionada pelo algoritmo proposto. Inicialmente, foi feita a análise com relação aos tipos de funções objetivo, sendo utilizado um nível de significância de 5%. Os resultados são apresentados na TABELA 7.11.

Contraste	diferença	+/- limites
tempo total x atraso médio	-8.15243	3.56839 *
tempo total x estoq. intermed.	-17.6935	3.56839 *
atraso médio- estoq. intermed.	-9.54102	3.56839 *

* denota uma diferença estatisticamente significativa

TABELA 7.11 - Teste LSD para ANOVA
Tipos de Funções Objetivo

Como era de se esperar, o teste de contrastes confirma a existência de diferenças significativas entre as melhorias nos três tipos de funções objetivo. Os melhores resultados foram, nessa ordem: estoques intermediários, atraso médio e tempo total de processamento.

O mesmo teste foi realizado considerando os Grupos de Problemas. A TABELA 7.12 apresenta os resultados obtidos, também com um nível de significância de 5%.

Contraste	diferença	+/- limites
Grupo1- Grupo 2	-2.57553	2.95678
Grupo1- Grupo 3	-9.88138	4.14402 *
Grupo1- Grupo 4	-3.97128	2.29869 *
Grupo1- Grupo 5	3.65216	5.58780
Grupo2- Grupo 3	-7.30585	4.43518 *
Grupo2- Grupo 4	-1.39575	2.78960
Grupo2- Grupo 5	6.22769	5.80701 *
Grupo3- Grupo 4	5.91010	4.02644 *
Grupo3- Grupo 5	13.5335	6.49243 *
Grupo4- Grupo 5	7.62343	5.50116 *

* denota uma diferença estatisticamente significativa

TABELA 7.12 - Teste LSD para ANOVA - Grupos de Problemas

Percebe-se que não houve diferenças significativas entre os Grupos 1 e 2, 1 e 5, 2 e 4. O Grupo 3 se destacou dos demais, tendo dado diferenças significativas em relação a todos os outros grupos, e apresentando os melhores resultados. A diferença mais significativa ocorreu entre os Grupos 3 e 5

Finalmente, a FIGURA 7.2 mostra o gráfico das interações entre grupos de problemas e tipos de funções objetivo utilizadas.

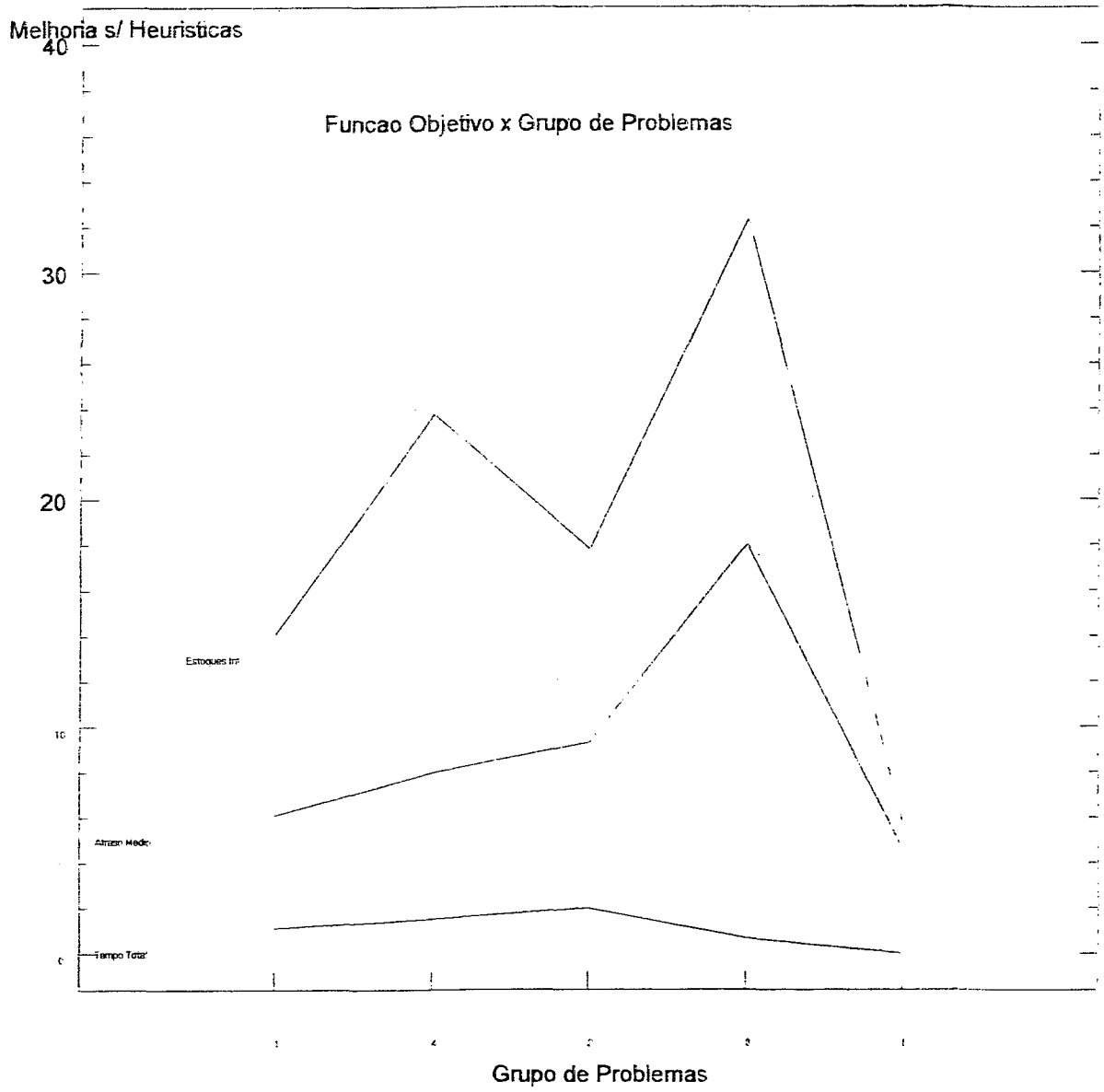


FIGURA 7.2 -Interações entre Grupos de Problemas x Função Objetivo.

Uma análise do gráfico apenas confirma, visualmente, o que os testes estatísticos demonstraram com relação aos melhores resultados obtidos com o uso de estoques intermediários, atraso médio e tempo total de processamento, nessa ordem.

7.5 Desempenho do Sistema Difuso de Calibração

O calibrador foi implementado e testado em problemas típicos de cada um dos cinco grupos e apresentou resultados que podem ser considerados satisfatórios.

Os testes demonstraram que ele coloca parâmetros consistentes. Como foi implementado segundo uma visão conservadora, apresenta uma tendência a definir um número muito elevado de iterações.

Deve-se ressaltar, entretanto, que, nesta pesquisa, o calibrador está a nível de protótipo, e novos testes devem ser feitos, possivelmente alterando as funções de pertinência.

Os resultados demonstram, entretanto, que o raciocínio difuso consegue calibrar o algoritmo. Sugere-se, numa nova pesquisa, colocar um sistema de aprendizado "on line" no calibrador, para melhorá-lo, uma vez que foi demonstrado que se trata de uma abordagem perfeitamente viável.

8. Conclusões e Recomendações

8.1 Conclusões

O estudo de problemas de programação da produção tem atraído a atenção de inúmeros pesquisadores desde que Johnson propôs soluções ótimas para problemas com duas e três máquinas.

Neste trabalho é apresentada uma abordagem baseada em algoritmos genéticos que dadas as suas características de paralelismo implícito e robustez, mostra resultados promissores. O paralelismo implícito, permite aos operadores genéticos pesquisar diferentes pontos do espaço de soluções, testando exponencialmente esquemas opcionais ou "building blocks", para construir boas soluções.

Uma abordagem baseada na técnica de algoritmos genéticos para o ambiente "flow shop" geral foi estudada. A representação especial dos cromossomas como uma lista de tarefas é proposta. Essa representação permite desconsiderar as relações de precedência entre tarefas a serem processadas em diferentes máquinas.

O algoritmo proposto separa a função objetivo da sua rotina principal, o que proporciona vantagens sobre as soluções heurísticas que são propostas para funções objetivo específicas. Essa separação permite rápida adaptação do algoritmo a qualquer ambiente do tipo "flow shop". Além disso, outras condições podem ser também implementadas como, por exemplo, considerar produtos semi- acabados. Dessa forma o algoritmo poderia ser utilizado para programar a produção diariamente, considerando produtos semi-acabados. Funções objetivo compostas por múltiplos também poderiam ser utilizadas.

A implementação computacional do algoritmo foi realizada utilizando a linguagem C++ no ambiente Windows 3.1. A utilização do modelo de objetos facilitará a integração do programa desenvolvido com outros sistemas.

Os algoritmos genéticos trabalham com um conjunto de soluções do problema, representadas na forma de listas de ordens ou cadeias de caracteres. Essas representações devem ser avaliadas a cada geração para reprodução de toda a população. A avaliação consome bastante tempo computacional, em torno de 60% do tempo total, mas, como sugerido por GOLDBERG (1989), poderia ser implementada em computadores com arquiteturas paralelas. Assim, o algoritmo proposto é adequado para essas arquiteturas de computadores que estão dia a dia se tornando mais acessíveis.

O sistema computacional desenvolvido foi utilizado para resolver problemas gerados aleatoriamente, através de três diferentes critérios de programação. Dadas as suas características híbridas o algoritmo obtém soluções iguais ou melhores que as das regras de programação heurísticas SPT, EDD e LST. Os resultados dos testes mostraram a viabilidade do algoritmo e, além disso, pode-se afirmar com 95% de confiança que o intervalo (8.22;11.13) contém a média da melhoria percentual proporcionada pelo algoritmo proposto.

A influência de parâmetros do algoritmo genético, tais quais, tamanho da população, número de gerações a serem avaliadas antes de ser finalizada a sua aplicação e a probabilidade de cada um dos operadores, é dependente do problema a ser considerado. Com a preocupação de tornar o algoritmo acessível, as pessoas não conhecedoras da técnica, um módulo de calibração é sugerido. Esse módulo calcula o valor dos parâmetros como função das características do problema, (número de produtos, número de máquinas). Utilizando para tal finalidade uma abordagem baseada em raciocínio difuso. Nessa abordagem sentenças condicionais difusas, fácil de serem definidas, são utilizadas. O referido módulo foi implementado no sistema, de forma a mostrar sua viabilidade, porém, são necessários ajustes nas funções de pertinência e nas regras utilizadas.

8.2 Recomendações

Problemas relacionados à Programação da Produção têm sido objeto continuado de estudos e pesquisas o que, sem dúvida, denota a abragência dessa área de conhecimento. Por outro lado, a utilização de técnicas

na busca de soluções de problemas de otimização. Partindo-se dessas duas premissas, é razoável admitir que novas pesquisas sobre aplicações de Algoritmos Genéticos em Programação da Produção devem constituir-se em um rumo a ser seguido.

Futuras pesquisas devem ser realizadas a respeito de diferentes critérios de otimização. Também, problemas onde os tempos de preparação são dependentes da seqüência de processamento das tarefas poderão ser investigados.

Os parâmetros do algoritmo genético proposto devem ser objeto de estudos. Para a realização desses estudos várias opções se apresentam, tais como, experimentação massiva, medição da efetividade dos operadores em gerar boas soluções, etc.

Um trabalho importante derivado desta pesquisa diz respeito à aplicação do algoritmo em diferentes situações práticas, pois, essas aplicações contribuirão para a validação da abordagem

Bibliografia

ADAM, N e SURKIS, J., "A comparison of capacity planning techniques in a job shop control system", *Managment Science*, 1977, 23/9,. 1011-1015.

ADELI, H "Expert system in construction and structural engineering", In; *Artificial intelligence and expert systems*, Chapman and Hall, 1988, 1-12.

ADSHEAD, N. S. e PRICE, D. H. R. "Adaptive scheduling: the use of dynamic due dates to accommodate demand variance in make-to-stock shop". *International Journal of Production Research,* 1988,16/7, 1241-1258.

ALASUVANTO, J.; ELORANTA, E; FUYUKI, M.; KIDA, T. e INOUE, I. "Object oriented programming in production management - two pilot systems". *International Journal of Production Research,* 1988, 26/5, 765-776.

ALIDAEE, B. "Schedule of n jobs on two identical machines to minimize weighted mean flow time". *Computers Industrial Engineering,* 1993, 24/1, 53-55.

ANTUNES JUNIOR, J. A., KLIEMANN NETO, F. J. e LIMA, I.S., *Reorganização da produção pela utilização da filosofia Justo-a-Tempo: O caso do setor metal-mecânico do estado do Rio Grande Do Sul,* 1990.

BAKER, K. R. *Introduction to sequencing and scheduling*, New York, John Wiley,1974.

BAKER, K. R. e SCHRAGE, L., "Finding an optimal sequence by dinamic programing: an extension to precedence-related tasks", *Operations Research,* 1978, 26,. 111-120.

BAKER, K. R. e BERTRAND, J. W. M., "An investigation of due-date assignment rules with constrainted tightness", *Journal of Operations Management,* 1981, 1/3, 109-120.

BAKER, K. R. e BERTRAND, J. W. M., "A dynamic priority rule for scheduling against due-dates", *Journal of Operations Management*, 1982, 3/1, 37-42.

BAKER, K. R., "Sequencing rules and due-date assignment in a job Shop", *Management Science*, 1984'30/9, 1093-1104.

BARR, A e FEIGENBAUM, E. A. *The Handbook of Artificial Intelligence*, Reading, Addison-Wesley, 1986. 409 p.

BECTE, W., "Theory and practice of load-oriented manufacturing control". *International Journal of Production Research*, 26/3, 375-395.

BECTOR, C.R., GUPTA, Y. P. E GUPTA, M. C., "Determination of on optimal common due date and optimal sequence in a single machine job shop". *International Journal of Production Research*. 1988, 26/4, 613-628.

BENSANA, E. *Utilisation de techniques d'inteligence artificielle pour l'ordonnancement d'atelier.* Thèse de docteur de L'ECOLE NATIONALE DE L'AERONAUTIQUE ET DE L'ESPACE, 1986.

BENSANA, E. , BEL, G., E DUBOIS, D., "Opal: a multi-knowledge-based system for industrial job-shop scheduling", *international Journal of Production Research*, 1988, 26/5, 795-819.

BERRY, W. L., SCHMITT, T. G., e VALLMANN, T. E., "Capacity planning techniques for manufacturing control systems: information requirements and operational features". *Journal of Operations Management*, 1982, 3/1. 13-25.

BERTRAND, J. W. M., "The effect of work load dependent due-dates on job shop performance", *Management Science*, 1983, 29/7, 799-816.

BERTRAND, J. W. M., "The use of work load information to control job lateness in controlled and uncontrolled release production systems". *Journal of Operations Management*, 1983, 3/2, 79-92.

BESTWICK, P. F., e LOCKYER, K. G., "A pratical approach to production scheduling", *International Journal of Production Research*, 1979, 17/2, 95-109.

BITRAN, G. R., HAAS, E. A. e HAX, A. C., "Hierarchical Production planning: aA two stage system", *Operations Research*, 1982, 30/2, 232-251.

BLACKSTONE, J. H., PHILLIPS, D. T. e HOGG, G. L., "A state-of-the-art survey of dispatching rules for manufacturing job shop operations", *International Journal of Production Research*, 1988, 26/5, 777-793.

CAO, Z.; KANDEL, A. and LI, L. "A. new model of fuzzy reasoning". *Fuzzy Sets and Systems*, 1990, 30.

CLEVELAND, G. A. and SMITH, S. F. "Using genetic algorithms to schedule flow-shop releases". *Proceedings of the Third International Conference on Genetic Algorithms,* 1989, San Mateo, 160-169.

CHIODINI, V., "An Expert system for dynamic manufacturing rescheduling", *Symposium on Real Time Optimization in Automated Manufacturing Facilities*, National Bureau of Standarts, Gaithersgurg, MD, January, 1986.

CONWAY, R. W., MAXWELL, W. L. and MILLER, L. M. *Theory of Scheduling*. London, Addison-Wesley, 1967. 294 p.

DANIELS, R. L and CHAMBERS, R. J. "Multiobjective flow-shop scheduling". *Naval Research Logistics*, 1990, 37, 981-995.

DANNENBRING, D. G. "Un evaluation of flow-shop sequencing heuristics". *Management Science*, 1977,11, 1174-1182.

DAVIS, L.D. "Job shop scheduling with genetic algorithms", *Proceedings of the 1st International Conference on Genetic Algorithms and their Applications*, 1985.

DAVIS, L.D. *Handbook of Genetic Algorithms*. New York, Van Nostrand Reinhold, 1991. 385p.

DE JONG, K. A. E SPEARS, W. M. "Using genetic algorithms to solve NP-complete problems". *Proceedings of the Third International Conference on Genetic Algorithms*. San Mateo, 1989, 124-132.

DEVI, B.B. e SARMA, V.V.S. "Estimation of fuzzy memberships from histograms". *Information Sciences*, 1985, 35, 43-59.

DOMBI, J. "Membership function as an evaluation", *Fuzzy Sets and Systems*, 1990, 35, 1-21.

DUBOIS, D. e PRADE, H. *Fuzzy Sets and Systems: Theory and Applications*,New York, Academic Press, 1980.

ERCHLER, J. e ESQUIROL, P. "Decision-aid job shop scheduling: a knowledge base approach", *IEEE Conference on robotics and automation*, San Francisco, 1986, 1651-1656.

ERMAN, L. D.; HAYES-ROTH, F; LESSER, V. R. e REDDY D. R. "The HEARSAY-II speech understanding system: integrating knowledge to resolve uncertainty", *Computer Surveys*, 1980, 12, 213-253.

ETIENNE, E. C., "MRP may not be right for you: at least not yet", *Production and Inventory Management*, 1983, 24/3, 33-46.

FERNANDEZ, D. O. *Introduction a la inteligencia artificial y los sistemas expertos*, Instituto Cubano de Investigaciones de los Derivados de la Cana de Azúcar, Havana, 1989.

FOX, M. S. e SMITH, S. F., "ISIS: A knowledged based system for factory scheduling", *Expert System Journal*, 1984, 1/1, 25-49.

GALLANT, S. I. "Connectionist expert systems". *Comunications of the ACM*, 1988, 31/2,.152-168.

GAREY, M.R. e JOHNSON, D. S. *Computers and Intractability: a guide to the theory of NP-completness*, W. H. Freeman and Company, New York, 1979.

GASCON, A. e LEACHMAN, R. C., "A dynamic programing solution to the dynamic, multi-item, single machine scheduling problem", *Operations Research*, 1988, 36/1, 50-56.

GAUTHIER, F.O., RODRIGUES FILHOE, I.W. e BARCIA, R.M. *Introdução à Inteligência Artificial* . Apostila do Curso Introdução à Inteligência Artificial, editada pelo Núcleo de Tecnologia de Software Ltda., Florianópolis, 1989.

GELDERS, L. F. e VAN WASSENHOVE, L. N., "Production planning: a review" *European Journal of Operational Research*, 1981, 7, 101-110.

GELDERS, L. F. e VAN WASSENHOVE, L. N., "Hierarchical integration in production planning: theory and pratice", *Journal of Operations Management,* 1982, 27-35.

GOLDBERG, D. E. *Genetic Algorithms in Search Optimization and Machine Learning*. New York, Addison-Wesley, 1989. 412p.

GOLDBERG, D. E. e LINGLE Jr.,R. " Alleles, Loci, and the Traveling Salesman Problem", *Proc. 1st Inter. Conf. on Genetic Algorithms and their Applications,* J.J. Grefenstette (ed), 1985.

GOLDARTT, E. M., "Computerised shop floor scheduling", *International Journal of Production Research*, 1988, 26/3, 443-455.

GRANT, T. J., "Lessons for OR from AI: a scheduling case study", *Journal of the Operational Society*, 1986, 37/1, 41-57.

GRAVES, S. C. "A review of production scheduling". *Operations Research*, 1981, 29/4.

GUPTA, J.N.D. e DARROW, W.P. "The two-machine sequence dependent folow-shop scheduling problem", *European Journal of Operations Research*, 1986, 24, 439-446.

HAN, W. e DEJAX, P. "An efficient heuristic based on machines workloads for the flow-shop scheduling problem with setup and removal".*Laboratoire Economique, Industriel et Social, Ecole Centrale, Paris.* 1991, 1-17.

HASTINGS, N. A. J., MARSHALL, P. H., e WILLIS, R. J., "Schedule based MRP: an integrated approach to production scheduling and material requirements planning", *Journal of the Operational Research Society*, 1982, 33, 1021-1029.

HENDRY, L. C. e KINGSMAN, B. G. "Production planning systems and their applicability to make-to-order companies", *European Journal of Operational Research*, 1989, 40, 1-15.

HILLIARD, M.R., LIPINS, G.E., PALMER, M., MORROW, M. e RICHARDSON, J. "A classifier-based system for discovering scheduling heuristics", *Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms* I Cambridge, 1987. p. 231-235.

HOLLAND, J.H. *Adaptation in Natural and Artificial Systems.* Cambridge, MIT Press, 1993. 211 p.

HUSBANDS, P., MILL,F. e WARRINGTON,S. "Genetic Algorithms, Production Plan Optimizacion and Scheduling" *Proceedings,* 1992.p 80-84.

JACOBS, F. R., "The OPT scheduling system: a review of a new scheduling system", *Production and Inventory Management*, 1983, 24/3, 47-51.

JACOBS, F. R., "OPT uncovered: many production planning and scheduling cancepts can be applied with or without the software", *Industrial Engeneering (US)*, 1984, 16/10, 32-41.

JANSEN, K. "Scheduling with constrained processor allocation for interval orders", *Computers Operations Research*, 1993, 20/6, 587-595.

KANET, J. J. e ADELSBERGER, H. H., "Expert systems in production scheduling", *European Journal of Operational Research*, 1987, 29/1, 51-59.

KERR, R. M., e EBSAY, R. V., "Implementation of an expert system for production scheduling". *European Journal of Operational Research*, 1988, 33, 17-29.

KHANNA, T. *Foundations of Neural Networks*, Reding, Addison-Wesley, 1990.

KING, J. R., e SPACHIS, A. S., "Scheduling: bibliography and review", *International Journal of Phisical Distribution and Materials Management*, 1980, 10/3, 105-132.

KUMARA S. R. T.; JOSHI, S.; KASHYAP, R. L.; MOODIE, C. L. e CHANG, T. C. "Expert systems in industrial engineering", *International Journal of Production Research*, 1986, 24/5, 1107-1125.

KUSIAK, A. e CHEN, M., "Expert systems for planning and scheduling manufacturing systems", *European Journal of Operational Research*, 1988, 34, 113-130.

LEE, C. C. "Fuzzy logic in control systems: fuzzy logic controller - part II" *IEEE Transactions on Systems Man and Cybernetics*. 1990, 20/2, 419-432.

LEUNG, K.S. e LAM, W. "Fuzzy concepts in expert systems", *IEEE Coomputer*, 1988, 43-56..

LEVITIN, G. e RUBINOVITZ, J. "Genetic algorithm for linear and cyclic assignment problem"., *Computers Operations Research*, 1993, 20/6, 575-586.

LUNDRIGAN, R. "What is that thing called Opt?", *Production and Inventory Management*, 1986, 2-11.

MALAKOOTI, B. & TSURHIMA, A. "An expert system using priorities for solving multiple-criteria facility layout problemas", *International Journal of Production Research*, 1989, 27/ 5, 793-808.

MANDANI, E. H. "Applications of fuzzy logic to approximate reasoning using linguistic synthesis". *IEEE Transactions on Systems, Man and Cybernetics*. 1987, C-26/12, 1182-1191.

MARUCHECK, A. S. e McCLELLAND, M. K., "Strategic issues in make-to-order manufacturing", *Production and Inventory Management*, 1986, 82-95.

McAEAEY, D.; HOEY, J. e LEONARD, R., "Designing the closed loop element of a materials requirements planning system in a low volume, make-to-order company (with case study)", *International Journal of Production Research*, 1988, 26/7, 1141-1159.

McKAY, K. N.; SAFAYENI, F. R. e BUZACOTT, J. A. "Job-shop scheduling theory: what is relevant?", *Interfaces*, 1988, 18/ 4, 84-90.

MELETON, M. P. Jr. "OPT - fantasy or breakthrough?", *Production and Inventory Management*, 1986.

MELNYK, S. A.; VICKERY, S. K. e CARTER, P. L., "Scheduling, sequenccing, and dispatching: alternative perspectives", *Production and Inventory Management*, 1986.

MIZUTO, M. and ZIMMERMAN, H. J. "Comparison of fuzzy reasoning methods". *Fuzzy Sets and Systems*. 1982, 8, 253-283.

NAKAMURA, N. e SALVENDY, G., "An experimental study of human decision-making in computer-based scheduling of flexible manufacturing system", *International Journal of Production Research*, 1988, 26/4, 567-583.

NELSON, M. M. e ILLINGWORTH, W. T. . *A pratical guide to neural nets*. Reading, Addison-Wesley, 1990.

NIX, S.J., COLLINS, A.G. e TSAY, T. *Knowledge-Based Expert Systems in Water Utility Operation and Management*. Denver, American Water Works Association, 1989. 137 p.

O'GRADY, P. J., e AOZA, M. A., "An adaptive approach to shop loading", *OMEGA International Journal of Management Science*, 1987, 15/2, 21-128.

ORCCIUCH, E. e fROST, J., "ISA: Intelligent scheduling assistant", *The First Conference on Artificial Intelligence Applications*, New York, 1984, 314-320.

PANWALKER, S. S. & ISKANDER, W., "A survey of scheduling rules", *Operations Research*, 1977, 25, 45-61.

PAO, Y. *Adaptative Pattern Recognition and Neural Networks*, Reading, Addison-Wesley, 1989.

PAPADIMITRIOU, D. H. e STEIGLIITZ, K., *Combinatorial Optimization: Algorithms and Complexity*, Englewood Cliffs, Prentice Hall, 1982.

PARK, Y. B.; PEGDEN, C. D. and ENSCORE, E. E. "A survey and evaluation of static flow-shop scheduling heuristics". *International Journal of Production Research.*,1984, 22, 127-141.

PARLAR, M. "EXPIM: a knowledge-based expert systems for production/inventory modelling", *International Journal of Production Research*, 1989, 27/1, 101-118.

PLENERT, G. e BEST, T. D. MRP, "JIT and OPT: what's "best"?", *Production and Inventory Management*, 1986.

QUADDUS, M. A., "A generalised model of optimal due-date assignment by linear programming", *Journal of the Operational Research Society*, 1987, 38/4, 353-359.

RAMAZANI, R. e YOUNIS, N. "Repetitive pure flowshop problems: a permutation approach", *Computers Industrial Engineering*, 1993, 24/1, 125-129.*i*Aroh

RICH, E. *Artificial Intelligence*, McGraw-Hill Book Company., 1983.

SARIN, S. C., AHN, S., e BISHOP, A., "An improved branching scheme for the branch and bound procedure of scheduling n job on m parallel machines to minimize total weighted tardiness", *International Journal of Production Research*, 1988, 26/7, 1183-1191.

SCHRAGE, L. e BAKER, K. R., "Dynamic programming Solution of Sequencing problems with precedence constraints", *Operations Research*, 1978, 26/3, 444-449.

SEN, T. e GUPTA, S.K. "A state-of-art survey of static scheduling research involving due dates". *The International Journal of Management Science,* 1984, 12/1, 63-76.

SHAW, M. J. "Knowledge-based scheduling in flexible manufacturing systems: an integration of pattern-directed inference and heuristic search", *International Journal of Production Research*, 1988, 26/5, 821-844.

SHAW, M. J. & WHISTON, A. B., "An artificial intelligence approach to the scheduling of flexible manufacturing systems", *IIE transactions*, 1989, 21/2.

SHIMOYASHIRD, S., ISODA, K., e AWANE, H., "Input scheduling and load balance control for a job-shop", *International Journal of Production Research*, 1989, 22/4, 597-605.

SMITH, S. F., FOX, M. S. e OW, P. S. "Constructing and maintaining detailed production plans: investigations into the development of knowledge-based factory scheduling systems", *AI Magazine*, 1986, Fall.

SWANN, D. "Using MRP for optmized scheduling (emulatiing OPT)", *Production and Inventory Management*, 1986, 27/2, 30-37.

TAILLARD, E. "Some efficient heuristc methods for the flow shop sequencing problem". *European Journal of Operational Research*, 1990, 47,.278-285.

TAILLARD, E. "Benchmarks for basic scheduling problems", *European Journal of Operational Research*, 1993, 64, 278-285.

WATERMANN, D. A. *A Guide to Expert Systems*, Addison-Wesley, 1986..

WEISS, S. M. e KUBKOWSKI, C. A. *A Pratical Guide to Designing Expert Systems*, Rowman & Allanheld, 1984.

WERNER, F. "On the heuristic solution of the permutation flow shop problem by path algorithms", *Computers Operations Research*, 1993, 10/7, 707-722.t

WESTBROOK, R., "Time to forget 'Just-in-Time'?: observations on recent visit to Japan", *International Journal of Operational and Production Management*, 1988, 8/4, 5-21.

WHITLEY, D. STARKWEATHER, T. e FUQUAY, D. "Scheduling problems and traveling salesmen: the genetic edge recombination operator". *Proceedings of the third international conference on genetic algorithms*, San Mateo, 1989, 133-140.

WIDMER, M. e HERTZ, A. "A new heuristic method for the flow shop sequencing problem". *European Journal of Operational Research*, 1989, 41, 186-193.

YU, C.; CAO, Z. e KANDEL, "Applications of fuzzy reasoning to the control of an activated sludge plant". *Fuzzy Sets and Systems*, 1990, 38, 01-14.

ZADEH, L. A. "Fuzzy sets", *Information and Control*, 1965, 338-353.

ZADEH, L. "Outline of a new approach to the analysis of comples systems and decision processes". *IEEE Transactions on Systems, Man and Cybernetics*, 1973, SMC-3/1, 28-44.

ZADEH, L. "The concept of a linguistic variable and its applications to approximate reasoning- part I". *Information Sciences*. 1975, 8, 199-249.

ZIMMERMANN, H. J. *Fuzzy Set Theory and Its Applicccations*, Boston, Kluwer, 1985.