**UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE FÍSICA DE SÃO CARLOS**

Utilização de Algoritmo Genético para Apoiar a Simulação de Sistemas Complexos

Júnia Coutinho Anacleto Silva

OK

Orientador: Prof. Dr. Claudio Kirner

**Tese apresentada ao Instituto de Física
de São Carlos da Universidade de São
Paulo, como parte dos requisitos para
obtenção do título de Doutor em
Ciências: Física Aplicada.**

São Carlos - SP

1996

IFSC-USP SERVIÇO DE BIBLIOTECA E
INFORMAÇÃO

Silva, Júnia Coutinho Anacleto

Utilização de algoritmo genético para apoiar a simulação de sistemas complexos/Júnia Coutinho Anacleto Silva.--São Carlos, 1996.

170 p.

Tese (Doutorado)--Instituto de Física de São Carlos, 1996.

Orientador: Prof. Dr. Claudio Kirner

1. Algoritmo Genético. 2. Simulação de sistemas.

I. Título.

MEMBROS DA COMISSÃO JULGADORA DA TESE DE DOUTORADO DE **JUNIA COUTINHO ANACLETO SILVA** APRESENTADA AO INSTITUTO DE FÍSICA DE SÃO CARLOS, UNIVERSIDADE DE SÃO PAULO, EM 22/11/1996.

COMISSÃO JULGADORA:

Prof. Dr. Claudio Kirner/UFSCar

Prof. Dr. Carlos Antonio Ruggiero/IFSC-USP

Prof. Dr. José Fernando Fontana/IFSC-USP

Prof. Dr. Sérgio Takeo Kofuji/EP-USP

Prof. Dr. Paulo Lício de Géus/UNICAMP

*Ao meu marido, Odney, e aos meus filhos,
Gustavo e aquele que se foi.*

Aos meus pais.

Agradecimentos

Ao Prof. Dr. Claudio Kirner pela sua orientação, viabilizando a realização deste trabalho.

À Profa. Dra. Maria do Carmo Nicoletti, pela minuciosa revisão deste trabalho.

À Renata, pelo grande apoio e pela troca de experiências.

À Sandra, pela amizade constante.

Ao Binho, pelo carinho e amizade tão preciosos.

Ao Silas, pelo paciente apoio técnico e incentivo.

Aos meus alunos de Iniciação Científica, Fábio, Renato, Patrícia e Juliano, pelo excelente trabalho desenvolvido.

Sumário

Lista de Figuras

Lista de Tabelas

Resumo

Abstract

1 INTRODUÇÃO	1
2 APLICAÇÃO DE SIMULAÇÃO NA RESOLUÇÃO DE PROBLEMAS	5
2.1 Conceitos Básicos sobre Simulação	6
2.2 Etapas do Processo de Simulação	9
2.3 Simulação de Eventos Discretos	15
2.4 Projeto do Modelo de Simulação	17
2.5 Análise dos Dados de Entrada e de Saída	23
2.6 Características de um Software de Simulação	26
2.7 Uma Alternativa para Análise de Dados de Simulação Utilizando AG	35
3 PRINCÍPIOS BÁSICOS DE UTILIZAÇÃO DE ALGORITMO GENÉTICO	38
3.1 Computação Evolucionária	40
3.2 Conceituação de Algoritmos Genéticos	43

3.3 Princípios Básicos na Aplicação do Algoritmo Genético	46
3.4 Reprodução	49
3.4.1 Operadores Genéticos	50
3.4.2 Parâmetros Genéticos	54
3.5 Fundamentos Teóricos de AG	55
3.5.1 Uma Abordagem Informal a Esquemas e ao Teorema dos Esquemas	56
3.5.2 Hipótese dos <i>Building Blocks</i>	57
3.5.3 Investigação e Exploração	59
3.6 Aspectos Práticos na Utilização do AG	60
3.6.1 Função Objetivo	61
3.6.2 Problemas de Limites do Desempenho	62
3.6.2.1 Convergência Prematura	63
3.6.2.2 Finalização Lenta	64
3.6.3 Técnicas de Seleção	64
3.6.4 Falhas de Gerações e Reposição Equilibrada	65
3.7 Aplicações com AG	67
3.7.1 Exemplos de Aplicações	68
3.7.2 Ambientes de Programação com AG	72
3.8 Simulação Apoiada por Algoritmo Genético	74
3.9 Considerações Finais.....	76
4 SIMAG - FERRAMENTA PARA SIMULAÇÃO DE SISTEMAS	
BASEADA EM AG	78
4.1 Desenvolvimento da Ferramenta	80
4.1.1 Módulo de Geração	82
4.1.2 Módulo de Execução	89

4.1.2.1 Operadores Genéticos Implementados	91
4.1.2.2 Política de Penalização	96
4.1.2.3 O Processo de Simulação	100
4.1.3 Módulo de Visualização da Evolução	104
4.1.4 Módulo de Estatísticas	108
4.2 Justificativas para o Desenvolvimento da Ferramenta e sua Utilização	111
5 ESTUDO DE CASO: UM EXEMPLO DE APLICAÇÃO DA TÉCNICA PROPOSTA	113
5.1 Descrição do Problema	114
5.2 Resolução do Problema Utilizando um Método Matemático	118
5.3 Resolução do Problema Através de Simulação Apoiada por AG	122
5.4 Análise Comparativa de Resultados Computacionais	126
5.4.1 Análise da Qualidade da Solução	129
5.4.2 Análise do Tempo de Execução	138
5.5 Análise da Aplicabilidade da Técnica	149
6 CONCLUSÕES	152
6.1 Atividades Realizadas	153
6.2 Principais Contribuições	155
6.3 Análise de Resultados	156
6.4 Trabalhos Futuros	158
7 REFERÊNCIAS BIBLIOGRÁFICAS	160

Lista de Figuras

Figura 1 - O Estudo de Simulação Computacional	19
Figura 2 - Versão Simplificada do Processo de Modelagem	22
Figura 3 - Uma Taxonomia para Modelos de Dados de Entrada	25
Figura 4 - Algoritmo Genético Tradicional	46
Figura 5 - Esquema de Aplicação do Algoritmo Genético	47
Figura 6 - Um Exemplo de Convergência de Desempenho em Aplicação de AG	49
Figura 7 - Indivíduos de uma População e sua Correspondente Roleta de Seleção .	51
Figura 8 - Exemplo de Aplicação do Operador Cruzamento	52
Figura 9 - Exemplo de Aplicação do Operador Mutação	53
Figura 10 - Esquema de um Simulador Apoiado por Algoritmo Genético	75
Figura 11 - Estrutura da Ferramenta para Simulação de Sistemas Baseada em AG .	81
Figura 12 - Tela do Menu Principal	82
Figura 13 - Exemplo de um Grafo Modelado pela Ferramenta.	82
Figura 14 - Menu de Opções do Editor Gráfico	84
Figura 15 - Tela de Edição de um Grafo	85
Figura 16 - Edição das Entradas Diretas do Grafo	85

Figura 17 - Grafo Gerado pela Ferramenta	86
Figura 18 - Edição da Função Objetivo	87
Figura 19 - Estabelecimento das Restrições da Entradas de um Grafo	87
Figura 20 - Definição do Nome do Arquivo do Sistema a ser Recuperado	88
Figura 21 - Visualização da Função Objetivo e Restrições	89
Figura 22 - Determinação do Tamanho da População e Número de Gerações	90
Figura 23 - Limitação dos Valores do Vetor de Entradas	90
Figura 24 - Definição do Nome do Arquivo de Resultados	91
Figura 25 - Tela de Execução do AG e Simulador	91
Figura 26 - O Processo de Simulação Apoiada por AG Proposto	95
Figura 27 - Variação da Taxa de Mutação Exponencial	96
Figura 28 - Política de Penalização Adotada	98
Figura 29 - Estrutura de um Nó do Grafo	101
Figura 30 - Estrutura do Grafo Manipulada pela Ferramenta	102
Figura 31 - Estrutura do Vetor de Entradas	103
Figura 32 - Estrutura do Vetor de Saída	104
Figura 33 - Algoritmo de Percorrimento do Grafo	105
Figura 34 - Escolha do Arquivo de Resultados a Ser Analisado	106
Figura 35 - Visualização da Evolução dos Valores Encontrados pelo AG	107
Figura 36 - Recuperação do Arquivo de Resultados para Análise Estatística	108
Figura 37 - Exemplo da Análise Estatística Feita pela Ferramenta	110

Figura 38 - Esquema de Entradas e Saídas da Refinaria	115
Figura 39 - Delimitação da Região Factível para o Problema da Refinaria	120
Figura 40 - Ampliação da Região Factível com Parametrização da F.O.	120
Figura 41 - Modelo Gráfico da Refinaria de Petróleo Proposto pela Ferramenta ..	122
Figura 42 - Modelo Gráfico da Refinaria de Petróleo Utilizando a Ferramenta	123
Figura 43 - Determinação da Função Objetivo e Restrições do Problema da Refinaria	124
Figura 44 - Resultados Computacionais para o Problema da Refinaria	124
Figura 45 - Evolução do Resultado Referente à Produção de Querosene.....	125
Figura 46 - Análise dos Resultados Computacionais Apresentados pela Ferramenta	126
Figura 47 - Qualidade das Soluções com AG com Taxa de Cruzamento Fixa e Taxa de Mutação Fixa	130
Figura 48 - Qualidade das Soluções com AG com Taxa de Cruzamento Fixa e Taxa de Mutação Exponencial	131
Figura 49 - Qualidade das Soluções com AG com Cruzamento Analítico e Taxa de Mutação Fixa	132
Figura 50 - Qualidade das Soluções com AG com Cruzamento Analítico e Taxa de Mutação Fixa	134
Figura 51 - Qualidade das Soluções com 10 Indivíduos na População	135
Figura 52 - Qualidade das Soluções com 50 Indivíduos na População	136
Figura 53 - Qualidade das Soluções com 100 Indivíduos na População	137

Figura 54 - Tempo de Processamento do AG Tradicional e do AG Proposto com População de 10 Indivíduos140

Figura 55 - Tempo de Processamento do AG Tradicional e do AG Proposto com População de 50 Indivíduos140

Figura 56 - Tempo de Processamento do AG Tradicional e do AG Proposto com População de 100 Indivíduos141

Figura 57 - Tempo de Processamento do AG com Taxa de Mutação Exponencial e Taxa de Cruzamento Fixa e do AG Proposto com População de 10 Indivíduos143

Figura 58 - Tempo de Processamento do AG com Taxa de Mutação Exponencial e Taxa de Cruzamento Fixa e do AG Proposto com População de 50 Indivíduos143

Figura 59 - Tempo de Processamento do AG com Taxa de Mutação Exponencial e Taxa de Cruzamento Fixa e do AG Proposto com População de 100 Indivíduos144

Figura 60 - Tempo de Processamento do AG com Taxa de Mutação Fixa e Cruzamento Analítico e do AG Proposto com População de 10 Indivíduos145

Figura 61 - Tempo de Processamento do AG com Taxa de Mutação Fixa e Cruzamento Analítico e do AG Proposto com População de 50 Indivíduos146

Figura 62 - Tempo de Processamento do AG com Taxa de Mutação Fixa e Cruzamento Analítico e do AG Proposto com População de 100 Indivíduos146

Figura 63 - Tempo de Processamento Considerando 100 Indivíduos na População148

Figura 64 - Tempo de Processamento Considerando 500 Indivíduos na População
.....148

Lista de Tabelas

Tabela 1 - Classes de Softwares de Simulação	29
Tabela 2 - Ambientes de Programação com AG e suas Categorias	73
Tabela 3 - Dados para Entradas e Produtos da Refinaria	115
Tabela 4 - Legenda usada nos Gráficos de Análise de Resultados	127
Tabela 5 - Resultados da Simulação Apoiada por AG variando as Implementações	128
Tabela 6 - Resultados Computacionais do AG Padrão	129
Tabela 7 - Resultados Computacionais do AG com Taxa de Mutação Exponencial e Cruzamento Padrão	131
Tabela 8 - Resultados Computacionais do AG com Taxa de Mutação Fixa e Cruzamento Analítico	132
Tabela 9 - Resultados Computacionais do AG com Taxa de Mutação Exponencial e Cruzamento Analítico	133
Tabela 10 - Resultados Computacionais de Cada Implementação do AG com População com 10 Indivíduos	135
Tabela 11 - Resultados Computacionais de Cada Implementação do AG com População com 50 Indivíduos	136

Tabela 12 - Resultados Computacionais de Cada Implementação do AG com População com 100 Indivíduos	137
---	------------

Tabela 13 - Tempos de Processamento do AG tradicional e do AG proposto	139
---	------------

Tabela 14 - Tempos de Processamento do AG com taxa de mutação exponencial e taxa de cruzamento fixa e do AG proposto	142
---	------------

Tabela 15 - Tempos de Processamento do AG com taxa de mutação fixa e cruzamento analítico e do AG proposto	145
---	------------

Resumo

Este trabalho apresenta a técnica de Algoritmo Genético para apoiar o processo de modelagem e simulação de sistemas complexos. Tal técnica pode ser vista como uma opção às técnicas tradicionais de modelagem e análise dos dados de simulação, simplificando todo esse processo, por suas características de simplicidade e generalidade, não exigindo conhecimento específico do domínio do problema. É apresentada uma ferramenta computacional - SimAG - baseada em Algoritmo Genético para modelagem e simulação de sistemas dessa natureza. Um exemplo de aplicação é estudado, onde pode ser constatada a viabilidade da utilização da técnica no processo de simulação.

Abstract

This work presents the Genetic Algorithm technique supporting the process of complex systems modeling and simulation. Such technique can be seen as an option to both the traditional modeling and the simulation data analysis techniques. Due to its inherent simplicity and generic application, it simplifies the whole process of simulation, demanding no specific knowledge over the problem domain. A genetic algorithm based computational tool - SimAG - for the modeling and simulation of such systems is here presented, and an example of its use is analyzed, thus demonstrating the feasibility of this new technique application in simulation processes.

Capítulo 1

Introdução

A simulação tem sido aplicada nas mais diversas áreas para resolver problemas complexos, sendo vista como uma ferramenta útil na análise e projeto de sistemas dessa natureza.

Entretanto, os métodos tradicionais para modelagem e análise de dados de simulação exigem do projetista um grande conhecimento sobre o comportamento do sistema, o que dificulta o trabalho de simulação. Além disso, essas técnicas podem se tornar inviáveis, dependendo da complexidade do problema em questão.

O processo de desenvolvimento de um modelo de simulação exige muito trabalho. Deve-se saber muito bem como usar o modelo (qual o tipo de distribuição estocástica dos dados de entrada), a fim de saber interpretar suas saídas (qual é a análise estatística mais adequada sobre os dados gerados).

O sucesso na interpretação dos dados de saída depende totalmente da escolha do modelo dos dados de entrada. Esse é o ponto crítico na aplicação de simulação

para resolução de problemas, sendo a causa de muitos dos problemas na sua utilização, como destaca Gogg; Mott (1993).

Os softwares para simulação computacional, através de métodos convencionais de geração e análise de dados de simulação, devem apresentar, segundo Pidd (1994), as seguintes características: um módulo de análise estocástica dos dados de entrada; um módulo para modelagem; um módulo para programação do modelo de simulação; e um módulo de análise estatística das saídas, sendo esta última a tarefa mais difícil do processo de simulação.

Algoritmo Genético (AG) é apresentado neste trabalho como uma alternativa às técnicas convencionais de geração e análise dos dados de simulação.

AG pode ser considerado como uma técnica computacional de otimização, inspirada na evolução dos seres vivos, baseada na seleção natural de Darwin, como descrevem Goldberg (1989) e Davis (1991), apresentando características de simplicidade e generalidade, além de não exigir um conhecimento específico do domínio do problema.

Utilizando AG para apoiar simulação, embora não se possa garantir que a solução ótima seja encontrada, têm-se vantagens em relação à simulação tradicional. A primeira vantagem é que o projetista fica livre da tarefa de analisar o comportamento dos dados de entrada do sistema. O AG gera um conjunto inicial de valores, ou população inicial, de maneira aleatória. Essa população é então utilizada para dar início ao processo evolutivo, sem necessidade de maior conhecimento sobre o comportamento do sistema.

Outra vantagem, é que os resultados gerados são avaliados pelo próprio AG, automaticamente, que verifica a qualidade de cada indivíduo como solução para o problema, independentemente de análises estatísticas.

Também o processo de realimentação dos dados de entrada, para a simulação prosseguir até que se consiga uma configuração ideal ou satisfatória para o sistema modelado, é feita pelo AG em função da análise da qualidade das soluções geradas pelos indivíduos, quando executado o modelo.

Seguindo a proposta de utilização de AG no apoio à simulação, é apresentado um ambiente computacional, que fornece ao projetista de sistemas complexos um meio adequado para desenvolvimento do modelo do sistema e a simulação desse modelo, pois a aplicação do AG implica em um processo simples mas repetitivo de geração e análise de dados. Um apoio computacional agiliza sensivelmente o processo e libera completamente o projetista desses sistemas, do processo de geração e análise dos dados de simulação.

O trabalho apresentado tem a organização descrita a seguir. No capítulo 2 são apresentados os conceitos básicos sobre simulação, discutindo as etapas do processo de simulação, simulação de eventos discretos e como projetar o modelo de simulação. Apresenta, também, a problemática relacionada com a análise dos dados de entrada e de saída de simulação, bem como as características que um software de simulação deve ter, além de apresentar uma proposta alternativa para análise de dados de simulação.

O capítulo 3 apresenta a técnica de Algoritmo Genético (AG), discutindo o contexto da chamada Computação Evolucionária. Os princípios básicos na aplicação do AG também são discutidos, bem como os aspectos práticos na utilização do AG. Um levantamento de um conjunto de exemplos de aplicação da técnica nas mais diversas áreas é apresentado. Também são apresentadas as características principais de um processo de simulação apoiada por AG.

No capítulo 4 é descrito o desenvolvimento da ferramenta computacional, utilizando a técnica proposta, sendo discutidas e reafirmadas as justificativas para o desenvolvimento da ferramenta e sua utilização.

O capítulo 5 apresenta uma descrição do problema estudado, sua resolução utilizando um método matemático, e a resolução do problema através de simulação apoiada por AG. É feita uma análise comparativa de resultados computacionais, quanto à qualidade da solução e quanto ao tempo de execução, considerando a resolução matemática e várias implementações de AG.

Por fim, no capítulo 6 são apresentadas as conclusões do trabalho realizado, suas principais contribuições, e os próximos trabalhos a serem desenvolvidos.

Capítulo 2

Aplicação de Simulação na Resolução de Problemas

Atualmente, a tecnologia de simulação está se tornando amplamente aceita para resolução de problemas mal definidos e de natureza adaptativa, que apresentam um certo grau de complexidade, e que envolvem um grande número de variáveis.

Neste capítulo são apresentados os conceitos sobre simulação, modelagem dos dados de entrada e análise dos dados de saída de simulação, baseados em estudos de Thesen; Travis (1991), Shannon (1992), e Gogg; Mott (1993), com o objetivo de utilizá-los na elaboração de uma ferramenta de apoio à tomada de decisão, voltada a projeto e operação de sistemas complexos.

Este capítulo está organizado da seguinte maneira: na seção 2.1 são apresentados os conceitos básicos sobre simulação; na seção 2.2 são discutidas as etapas do processo de simulação; na seção 2.3, discute-se a simulação de eventos

discretos; a seção 2.4 mostra como projetar o modelo de simulação; a seção 2.5 apresenta a problemática relacionada com a análise dos dados de entrada e de saída da simulação; a seção 2.6 mostra as características que um software de simulação deve ter; finalmente, na seção 2.7 é apresentada uma proposta alternativa para análise de dados de simulação.

2.1 Conceitos Básicos sobre Simulação

Da mesma maneira que a atividade de programação foi considerada por um bom tempo mais uma arte que uma ciência, assim tem ocorrido também com a simulação. Nesse sentido, a simulação pode ser definida como a arte e a ciência de criar uma representação de um processo ou sistema, visando a experimentação e avaliação de problemas, como diz Sadowski (1991). A simulação também tem sido vista como uma metodologia para resolução de problemas, indispensável para engenheiros, projetistas e gerentes, sendo usada como uma das mais poderosas ferramentas de análise para auxiliar no planejamento, projeto e controle de sistemas complexos, cita Shannon (1992).

De maneira geral, o funcionamento de um sistema a ser projetado pode ser testado de três maneiras, que são descritas por Pidd (1994):

- No mundo real, de maneira controlada, para que seus efeitos possam ser compreendidos e analisados.

- Um modelo matemático poderia ser desenvolvido para estudar o sistema, o que é uma especialidade da área de Pesquisa Operacional.
- A simulação do sistema através de um modelo computacional poderia ser usada, realizando experimentos sobre o modelo, permitindo que várias políticas sejam testadas, para ver qual a melhor política a ser adotada na prática.

Das três formas mencionadas, a simulação se apresenta como a melhor alternativa para projeto de sistemas complexos. A primeira alternativa, apesar de ser mais adequado no treinamento de pessoal, é um processo muito dispendioso, sendo muitas vezes inviável na prática. Já no caso de modelagem matemática, pode-se incorrer no uso excessivo de aproximações, levando a conclusões incorretas, relata Gogg, Mott (1993).

Como terceira alternativa, a simulação se apresenta como uma ferramenta para fazer previsões de resultados a curto e longo prazo, previsões essas que servirão como base para tomada de decisão gerencial.

A simulação leva em conta os efeitos das variâncias ocorridas em sistemas dinâmicos e estocásticos. Tanto os métodos analíticos convencionais, quanto os modelos matemáticos estáticos, têm seus cálculos de desempenho baseados na média, e desconsideram os efeitos das variâncias. Isso muitas vezes pode levar a conclusões errôneas.

O uso de modelagem de simulação para projetar novos sistemas e ajustar o desempenho de sistemas existentes continua a aumentar rapidamente devido ao aumento da complexidade dos sistemas atuais, custos reduzidos de computação,

melhores softwares de simulação, e o uso de animação, como relata Law; McComas (1991).

A simulação envolve a modelagem de um sistema, de tal forma que o modelo imita a resposta do sistema atual a eventos que acontecem no decorrer do tempo, relata Pritsker et al. (1991). É geralmente usada para determinar como alguns aspectos de um sistema deveriam ser ajustados ou operados. A modelagem dos dados de simulação é a atividade mais complexa de todo o processo, como afirma Law; McComas (1991).

Além disso, a simulação é um exercício de programação muito complicado, pois o programa computacional descreve como o sistema opera e coleta dados sobre o desempenho do sistema, deixando a cargo do analista propor situações a serem testadas e tirar conclusões a partir dos resultados da simulação, destaca Thesen; Travis (1991).

De qualquer maneira, a simulação é uma técnica amplamente aceita para resolução de problemas, que pode ajudar a determinar qual solução candidata é ótima com relação aos resultados desejados.

A seleção da solução candidata segue os seguintes passos: estabelecer os critérios para avaliar o seu desempenho e seus pesos relativos; calcular o seu desempenho com respeito aos critérios estabelecidos; comparar as soluções candidatas tendo como base o desempenho de cada solução. Nesse contexto, pode-se entender simulação como uma ferramenta para prever o desempenho de possíveis alternativas, transformando dados em conhecimento e conhecimento em experiência, afirma Gogg; Mott (1993).

Quase qualquer tipo de sistema pode ser simulado, diz Shannon (1992). A área de aplicação de simulação está em contínua expansão - manufatura, processos químicos e de alimentos, sistemas de distribuição, redes de comunicação, transportes, indústria de serviços, militar, sistemas computacionais, etc.

Os sistemas mais adequados para simulação têm as seguintes características: são dinâmicos, pois seu comportamento se modifica no decorrer do tempo; são complexos, pois muitos objetos interagem com o sistema, onde a dinâmica de cada um deve ser considerada e analisada, conforme relata Gogg, Mott (1993); e seus componentes apresentam grande interação entre si.

2.2 Etapas do Processo de Simulação

Quando se pensa em usar simulação na resolução de problemas, é necessário conhecer a metodologia de simulação, modelos estocásticos de pesquisa operacional, teoria de probabilidade e estatística. A simulação inclui técnicas de validação, seleção da distribuição de probabilidade dos dados de entrada, e análise dos dados de saída da simulação, destaca Law; McComas (1991).

Deve-se primeiro identificar e definir o problema a ser resolvido, tendo o cuidado de formular o problema corretamente, e então determinar se a simulação é a ferramenta apropriada. O processo continua com o desenvolvimento de uma definição de objetivos clara e concisa, de forma a obter boas informações sobre os procedimentos de operação do sistema e sua lógica de controle.

Definidos os objetivos, o projetista pode começar a formular o modelo, devendo modelar a aleatoriedade do modelo, determinando o comportamento das entradas do sistema e qual a distribuição de probabilidade que melhor representa esse comportamento. Inicia-se então o estágio de construção do modelo. Em algum ponto nesse processo, os requisitos de dados devem ser definidos e a geração dos dados de entrada necessários deve ser iniciada. Completada a construção do modelo, o projetista deve então verificar e validar o modelo, afirma Sadowski (1991).

O projetista pode então começar a testar o modelo, utilizando corretamente um software de simulação previamente escolhido. O estágio de experimentação fornecerá diretivas sobre o possível comportamento do sistema modelado. Esses resultados devem ser cuidadosamente interpretados, considerando os objetivos estabelecidos. Deve-se avaliar cada alternativa para determinar qual é a melhor com relação ao critério de desempenho, usando os procedimentos estatísticos apropriados para interpretar a saída da simulação. Finalmente, as conclusões sobre a melhor alternativa para resolução do problema devem ser tiradas e implementadas para completar o processo, conclui Gogg; Mott (1993).

Sadowski (1991) apresenta um conjunto de recomendações para definição do problema, desenvolver uma especificação funcional, formulação do modelo e construção do modelo, coleta de dados de entrada do sistema, verificação e validação do modelo, fazer a análise dos dados gerados pela simulação, e documentação do trabalho. O objetivo é fornecer técnicas para conduzir um projeto de simulação com sucesso. Essas recomendações são descritas as seguir.

Durante o processo de definição do problema deve-se definir cuidadosamente o “problema real” e não o sintoma do problema.

Uma definição acurada do problema pode especificar o nível de detalhamento exigido no modelo, indicando áreas específicas onde devem ser tomados cuidados especiais. A definição do problema geralmente leva diretamente ao estabelecimento dos objetivos que o modelo de simulação resultante deve alcançar. Embora isso pareça simples, pode exigir tempo e esforço significativos e ainda não resultar em um conjunto de objetivos bem definido, se todas as pessoas envolvidas no projeto não tiverem uma visão clara do problema.

A construção de um modelo de simulação com poucos detalhes pode resultar em informações pouco precisas para alcançar o objetivo real. Se o modelo contém muitos detalhes, vai exigir mais esforço para ser criado, tempos de execução mais longos, e pode conter mais erros.

Portanto, uma especificação funcional auxilia o projetista no dimensionamento do nível de detalhamento do modelo. O desenvolvimento de uma especificação funcional exige que o projetista defina todas as entradas e saídas da simulação, descreva cada componente da simulação, defina toda a lógica de controle para o modelo, e apresente pelo menos uma pequena discussão sobre possíveis extensões.

O objetivo geral da especificação funcional é especificar completamente os requisitos da ferramenta. As vantagens de desenvolver uma especificação funcional antes do desenvolvimento do modelo são:

- forçar o usuário a definir toda a lógica e dados no estágio inicial do processo;
- prevenir o usuário contra mudanças periódicas de escopo ou objetivos do projeto, com base nos eventos correntes;

- permitir que as estruturas de dados a serem usadas na simulação sejam organizadas previamente.

O processo de modelagem pode ser separado em dois estágios: formulação e construção. O estágio de formulação do modelo é uma atividade mental que se preocupa em desenvolver uma estratégia para a simulação e avaliar abordagens alternativas de modelagem. Idealmente, várias abordagens devem ser consideradas e avaliadas antes da construção do modelo. A fase de construção do modelo consiste na integração das estruturas de dados e do esquema de modelagem em um modelo de simulação articulado.

Na fase de coleta de dados de entrada, o projetista deve definir e iniciar o procedimento de coleta de dados. Se possível, isso deve ser feito no início do projeto de simulação, para que a informação requerida esteja disponível quando necessária.

Quando não for possível obter dados acurados, mas apenas algumas estimativas, deve-se ajustar estas estimativas a distribuições estocásticas, permitindo sua inclusão no modelo de simulação. Os parâmetros exatos necessários são dependentes da distribuição escolhida.

Para verificação e validação do modelo discutidos a seguir, não existem procedimentos passo a passo que ajudam o projetista nessas tarefas.

Verificação - objetiva garantir que o modelo se comporta da forma que se pretende - é a tarefa mais fácil das duas. É a garantia de que o programa computacional do modelo e sua implementação estão corretos, afirma Sargent (1994). Entretanto, pode ser difícil conduzir uma verificação completa de um modelo complexo. A dificuldade é abordar todas as possíveis situações que podem ocorrer,

quando o modelo completo está sendo experimentado. Durante essa fase, deve-se garantir a verificação da função do modelo sob condições limites.

Validação - objetiva garantir que o modelo se comporte da mesma maneira que o sistema real - pode ser difícil e, em alguns casos, impossível de ser feita. É a confirmação de que um modelo computacional, dentro do seu domínio de aplicação possui um nível satisfatório de precisão, consistente com a aplicação desejada do modelo, define Sargent (1994). Se não há um sistema real para ser comparado com o modelo de simulação, a validação não pode ser feita.

A lógica de controle, que dita como as decisões devem ser tomadas durante a execução da simulação, deve ser completamente definida durante o desenvolvimento da especificação funcional. Tentar capturar a lógica empregada por operadores ou controladores humanos pode ser uma tarefa extremamente desafiadora.

O melhor caminho para descobrir os procedimentos operacionais é durante o processo de verificação. Essa verificação é melhor executada com o auxílio de animação. Usar um modelo ilustrado do sistema é melhor que usar gráficos de fluxo ou tabelas de dados.

Na fase de análise dos dados gerados pela simulação, há muita literatura disponível para o analista; a maioria, entretanto, é irrelevante ou difícil de entender. As características básicas a serem consideradas em qualquer análise são de natureza estatística. A primeira recomendação é adquirir um conhecimento básico em estatística. Na fase de análise, existem duas questões a serem respondidas:

- Quanto tempo deve demorar cada simulação para obter informação significativa?
- Há diferenças entre conjuntos de resultados a serem comparados?

A primeira é a questão mais difícil de responder. Na dúvida, a sugestão é fazer simulações longas.

Para responder à segunda questão, a animação gráfica é uma excelente ferramenta para análise de gargalos e para identificar a causa de um comportamento estranho do sistema, identificados via diferenças entre conjuntos de resultados. Entretanto, não deve ser usada como única ferramenta de análise. Uma cuidadosa análise estatística nunca deve ser deixada de lado.

A documentação do modelo deve descrever as estruturas de dados, os elementos básicos do modelo, a lógica geral de fluxo e todas as variáveis, filas, etc.

Resumindo, o processo de simulação deve ter as seguintes etapas:

1. definição do problema - quais os objetivos do estudo.
2. especificação funcional.
 - 2.1 planejamento do projeto - quais os recursos necessários.
 - 2.2 definição do sistema - como o sistema trabalha e quais as restrições.
3. formulação e construção do modelo.
 - 3.1 formulação do modelo conceitual através do uso de diagrama de blocos, pseudocódigo, etc., para a especificação da lógica.
 - 3.2 projeto experimental preliminar:
 - 3.2.1 quais as medidas de eficácia a serem usadas.
 - 3.2.2 quais os fatores a serem variados e investigados.
 - 3.2.3 quais dados precisam ser colhidos do modelo.

4. preparação dos dados de entrada.
5. implementação do modelo.
6. verificação e validação - depuração e verificação se a saída do modelo reflete o sistema real.
7. projeto experimental final - projetar o conjunto de dados de entrada, tal que se obtenha a informação desejada, e definir como a execução dos testes deve ser conduzida.
8. experimentação - execução da simulação.
9. análise e interpretação - uso de inferências estatísticas.
10. implementação e documentação - usar os resultados, registrar as descobertas e documentar o modelo e seu uso.

2.3 Simulação de Eventos Discretos

De acordo com Thesen; Travis (1991), a simulação de eventos discretos descreve sistemas que mudam instantaneamente em resposta a um certo acontecimento, chamado de ocorrência ou evento discreto.

Quando se escolhe modelar um sistema do mundo real, usando simulação de eventos discretos, abre-se mão da capacidade de capturar um certo grau de detalhamento que pode ser descrito somente com mudanças contínuas suaves. Em contrapartida, obtém-se a simplicidade que permite captar as principais características de alguns tipos de sistemas, que são muito complexos para serem modelados com

simulação contínua. A simulação de eventos discretos se baseia nas seguintes estruturas, descritas por Pidd (1994):

- entidades: o comportamento do modelo é composto do comportamento de objetos individuais de interesse, chamados de entidades. O programa de simulação traça o comportamento de cada entidade, durante o tempo simulado, verificando a lógica individual de cada uma delas. Uma entidade ou pode ser um objeto, como uma máquina, uma pessoa, etc., ou então um conjunto de objetos, como uma célula de manufatura, um grupo de pessoas, etc.;
- eventos discretos: um evento é um ponto no tempo, quando a entidade muda de estado. O comportamento das entidades é modelado como uma sequência de eventos. O fluxo do tempo na simulação de eventos discretos não é suave, movendo-se de um tempo de evento a outro, sendo que esses intervalos podem ser irregulares;
- comportamento estocástico: os intervalos variantes entre eventos devem ser modelados estocasticamente, usando métodos de amostragem baseados na teoria de probabilidade.

Um programa que implementa um modelo de simulação discreta deve ter os seguintes componentes:

- a lógica de eventos: uma definição precisa das condições que controlam as mudanças de estado das entidades que serão incluídas no modelo. Isso pode ser feito através de modelagem, usando algum tipo de diagrama;

- o programa de controle ou programa executivo: que garante que o evento ocorre no momento exato e na sequência correta. Isso garante que o comportamento agregado é um modelo do sistema simulado;
- rotinas de geração de amostras, algoritmos de integração, gráficos e outras características necessárias para representar diversos tipos de modelo.

Gogg, Mott (1993) descreve que um modelo de simulação de eventos discretos executa a seguinte sequência de instruções repetitivas:

1. determina qual o tipo de evento irá ocorrer a seguir;
2. ajusta a variável relógio da simulação ao tempo do próximo evento;
3. atualiza todas as variáveis estáticas que foram usadas;
4. realiza as ações (cálculos) associadas ao último evento ocorrido;
5. escalona o tempo para a próxima ocorrência daquele tipo de evento.

2.4 Projeto do Modelo de Simulação

Os modelos de simulação têm sido cada vez mais usados na resolução de problemas e no apoio à tomada de decisão. Tanto a certeza de que o modelo está correto, quanto seus resultados, são fundamentais para o êxito da aplicação da simulação, afirma Sargent (1994).

Simulação é o processo de projetar um modelo de um sistema real e conduzir experimentos com esse modelo, através de sua execução, com o objetivo de analisar

Simulação de Eventos Discretos

e entender o comportamento do sistema, avaliando várias estratégias para a operação do sistema. Isso inclui a construção do modelo e o seu uso experimental para estudar o problema. É uma metodologia experimental e aplicada que procura descrever o comportamento dos sistemas, construir teorias ou hipóteses de acordo com o comportamento observado, e usar o modelo para prever o comportamento futuro frente às mudanças no sistema, define Shannon (1992). É um processo de 3 componentes fortemente acoplados e interativos que são: projeto do modelo, execução do modelo e análise da execução. Tal processo pode ser visto esquematicamente na figura 1.

Talvez o problema maior de simulação seja determinar o método exato que deve ser usado para criar um modelo, conforme discute Fishwick (1994). A modelagem computacional é um processo de simplificação e abstração do sistema real, no qual o projetista tem que isolar aqueles fatores que são cruciais para a operação do sistema que está sendo modelado. Esse processo de abstração depende de dados e informações sobre o sistema em questão, e também sobre os propósitos da simulação.

Quando executado de maneira adequada, o processo de construção do modelo de simulação traz uma grande quantidade de conhecimento, informação e experiência de várias fontes. O conhecimento sobre o problema é exposto sob múltiplos pontos de vista, o que possibilita uma compreensão melhor do sistema como um todo, ao mesmo tempo em que as interdependências de todos os componentes são compartilhadas e compreendidas por todos os indivíduos que contribuem para a confecção do modelo, destaca Gogg, Mott (1993).

Figura 1 - O Estudo de Simulação Computacional

O processo de desenvolvimento de um modelo de simulação exige muito trabalho. Entretanto, depois de construído, tem-se um excelente veículo para testar várias idéias diferentes sem tanto trabalho. Deve-se saber muito bem como usar o modelo (qual o tipo de distribuição estocástica dos dados de entrada), a fim de saber interpretar suas saídas (qual é a análise estatística mais adequada), destaca Pidd (1994).

Modelagem de simulação é atualmente uma atividade sofisticada de análise de sistemas, e a codificação do modelo representa somente 30 a 40% do esforço total em um estudo típico de simulação, como visto em Law, McComas (1991). Um modelo deve ser desenvolvido para uma aplicação ou propósito específico e sua validade é determinada com respeito àquele propósito.

Vários conjuntos de condições experimentais são geralmente necessários para definir o domínio de aplicabilidade desejado do modelo. Este pode ser válido para um conjunto de condições experimentais e inválido para outro.

Um modelo é considerado válido, para um conjunto de condições experimentais, se sua precisão está dentro de limites aceitáveis para o resultado gerado, que é a precisão exigida para o propósito específico do modelo.

Geralmente, várias versões do modelo são desenvolvidas até que se obtenha um modelo válido satisfatório. É caro e demorado determinar se um modelo é totalmente válido sobre o domínio completo de aplicação desejado. Sendo assim, são feitos testes e avaliações até que seja obtida a confiança suficiente para que um modelo seja considerado válido.

Com relação ao processo de validação do modelo, há três abordagens básicas de tomada de decisão para determinar se um modelo de simulação é válido:

- a equipe de desenvolvimento do modelo decide sobre a validade do modelo, o que é uma decisão subjetiva, baseada nos resultados dos vários testes e avaliações feitos como parte do processo de desenvolvimento do modelo. Essa é a abordagem mais utilizada;
- os processos independentes de validação e verificação baseiam-se no veredicto de uma terceira equipe para decidir se o modelo é válido. Esse tipo de validação é usado quando há um custo elevado associado ao problema de como o modelo de simulação está sendo usado, ou para ajudar na aceitação do modelo e para dar credibilidade ao modelo. Esse processo é muito dispendioso e demorado, além de necessitar do envolvimento da terceira equipe no final do processo de desenvolvimento do modelo;
- pode-se fazer uso de um método de pontuação para determinar se um modelo é válido, onde os pesos são determinados de forma subjetiva (o que pode ser um problema) quando são conduzidos vários aspectos do processo de validação. O modelo de simulação é considerado válido se sua pontuação geral e por categoria são maiores do que em qualquer outro modelo. É uma abordagem pouco usada.

A verificação e a validação do modelo podem estar relacionadas com o processo de desenvolvimento do modelo das seguintes maneiras:

- através de um processo de desenvolvimento de um modelo detalhado;
- através de um processo de desenvolvimento de um modelo simples, o que mostra mais claramente a verificação e a validação do modelo.

Considerando a segunda alternativa apontada acima, o processo de modelagem de simulação pode ser entendido, de maneira simplificada, como mostra a figura 2 apresentada por Sargent (1994). É necessário, então, fazer:

- validade do modelo conceitual - determinar que as teorias e suposições que definem o modelo conceitual estão corretas e que a representação do modelo da entidade problema é "razoável";
- verificação do modelo computacional - garantir que a programação e a implementação do modelo conceitual estão corretas;
- validade operacional - determinar que o comportamento das saídas do modelo tem precisão suficiente para o propósito proposto sobre o domínio de aplicação proposto;
- validade dos dados - garantir que os dados necessários para construção do modelo, avaliação e testes, bem como a condução dos experimentos do modelo para resolver o problema são adequados e corretos.

De acordo com Law; McComas (1991), o uso de animação gráfica é um apoio importante para comunicar a essência de um modelo de simulação, sendo útil na verificação do programa e descoberta de erros na lógica do modelo. Entretanto, a

animação não é um substituto para uma análise estatística dos dados de saída, e também a animação correta não garante a validade do modelo depurado.

Figura 2 - Versão Simplificada do Processo de Modelagem

No desenvolvimento de sistemas, através de técnicas convencionais, a primeira versão executável corresponde à própria implementação do sistema. Portanto, a existência de linguagens ou ferramentas, que propiciem a execução do modelo de um sistema, para que dessa forma o comportamento do mesmo possa ser simulado, permite que erros sejam detectados em fases iniciais do projeto e que a solução dos problemas não seja adiada para estágios finais de desenvolvimento.

2.5 Análise dos Dados de Entrada e de Saída

A seleção da distribuição de probabilidade adequada é crítica para a arte de construção do modelo, de acordo com Thesen; Travis (1991) e Gogg; Mott (1993). As distribuições de probabilidade são a base para gerar comportamento estocástico (variação aleatória) em simulação. Tal comportamento é um conjunto de valores, ou medidas, que mostram a frequência relativa com a qual um evento ocorre ou parece ocorrer.

As distribuições de probabilidade padrão são geralmente usadas para representar distribuição de dados empíricos, pois ajudam a diminuir as irregularidades que podem ocorrer nos dados, devido a perda de valores durante a fase de coleta dos dados.

Os modelos de simulação discreta têm componentes estocásticos que simulam a natureza probabilística do sistema considerado. Uma modelagem adequada das entradas exige uma forte relação entre o modelo da entrada e o mecanismo probabilístico associado ao sistema, segundo Leemis (1994).

As seguintes considerações devem ser feitas:

1. existe uma fonte confiável de números aleatórios?
2. há um algoritmo para converter esses números aleatórios em variáveis aleatórias associadas ao modelo de entrada para direcionar a simulação?
3. os dados estão disponíveis no contexto da simulação de interesse?

Nessas condições, devem ser selecionados os modelos probabilísticos adequados para os componentes aleatórios no modelo de simulação.

Leemis (1994) apresenta uma taxonomia para auxiliar o projetista na escolha do modelo dos dados de entrada para simulação, apresentada na figura 3.

Na simulação estocástica são sempre exigidas cópias múltiplas e independentes do modelo. A análise estatística da saída gerada pelas cópias é um pré-requisito para tirar conclusões válidas, de acordo com Gogg; Mott (1993).

Dados gerados de simulações estocásticas são por si estocásticos. As saídas geradas das múltiplas cópias do modelo devem ser analisadas com princípios de inferência estatística, para se chegar a conclusões válidas.

Construir um modelo de simulação é muito trabalhoso. Entretanto, deve-se pensar cuidadosamente em como exatamente vai ser usado o modelo, a fim de saber interpretar suas saídas, como diz Kelton (1994).

Os modelos de simulação estocásticos são mais realistas para sistemas que envolvem algum tipo de entrada aleatória, incerta. Devem ser gerados valores adequados e confiáveis para essas entradas.

Os sistemas com entradas aleatórias certamente têm saídas aleatórias. Portanto, deve-se ter cuidado na execução dos tipos certos de simulação, com relação ao modelo dos dados de entrada, e fazer o tipo certo de análise estatística sobre os dados de saída gerados pela simulação.

Para o planejamento da execução da simulação, deve-se pensar a priori quais as situações o modelo deverá avaliar: quais os parâmetros de entrada devem ser

considerados e quais as combinações serão feitas, quais os efeitos das mudanças em algum parâmetro de entrada, e se há relação com algum outro parâmetro modificado.

Figura 3 - Uma Taxonomia para Modelos de Dados de Entrada

Grandes simulações geralmente envolvem vários fatores de entrada, devendo-se fazer algum tipo de seleção desses fatores para se obter uma análise viável dos dados de saída. Para a análise estatística, tem-se usado representação gráfica. O uso de animação, através de ferramentas gráficas está se tornando popular e às vezes eficaz, mas deve sempre estar associada à análise estatística inferencial.

2.6 Características de um Software de Simulação

A seleção de um software de simulação adequado dentro do grande número de pacotes disponíveis é uma tarefa difícil. Embora um software adequado seja muito útil para completar o projeto de simulação, ele não fará todo o trabalho sozinho. Deve-se definir o modelo, e o projetista deve conhecer metodologia de simulação. Banks (1991) sugere que o projetista faça cursos especializados no assunto.

A escolha do software de simulação tem grande impacto sobre o sucesso do projeto (nível de detalhamento, validade do modelo, tempo de execução do modelo, e tempo de projeto). Há duas grandes classes de softwares usados na modelagem de simulação: linguagens de programação de propósito geral e softwares de simulação.

Linguagens de programação de propósito geral geralmente já são conhecidas do analista e estão disponíveis em qualquer computador. Além disso, essas linguagens são de aquisição mais barata.

Os softwares de simulação, por outro lado, reduzem o tempo de programação significativamente, fornecem uma estrutura natural para modelagem de simulação e, tipicamente, reduzem o custo geral do projeto. Podem ser linguagens de simulação, simuladores orientados a aplicações, ferramentas de modelagem rápida, ou planilhas de cálculo.

As linguagens de simulação surgiram para facilitar a programação de modelos de simulação, de acordo com Law; Kelton (1982), pois algumas características são comuns a quase todos os modelos, tais como:

- geração de números aleatórios;
- geração de variáveis aleatórias para uma distribuição probabilística específica;
- controle do tempo de simulação;
- geração de relatórios dos resultados da simulação.

Junto com a necessidade de se programar um modelo, vem a decisão de qual linguagem deve ser utilizada. Algumas vantagens que as linguagens de simulação têm em relação às linguagens de propósito geral são:

- as linguagens de simulação fornecem a maioria das características necessárias para programação de um modelo de simulação, possibilitando a diminuição do tempo de programação;
- a tradução do modelo para tal linguagem é mais natural;
- as alterações no modelo são feitas com mais facilidade;
- essas linguagens fornecem alocação dinâmica de memória;
- os erros de programação são minimizados e são detectados com mais facilidade.

Em contrapartida, as vantagens da utilização de linguagem de propósito geral são:

- a maioria dos usuários já conhece uma linguagem de propósito geral;
- tais linguagens estão disponíveis na maioria dos computadores;
- um programa escrito em uma linguagem de propósito geral pode ser mais rápido que um programa escrito em uma linguagem de simulação;
- linguagens de propósito geral são mais flexíveis.

A escolha de qual linguagem usar para simulação depende de uma série de fatores, tais como:

- portabilidade da linguagem;
- custo de instalação e manutenção;
- qualidade da documentação da linguagem;
- facilidade de aprendizagem.

Esses fatores podem determinar qual ou quais linguagens para simulação deve-se adquirir e qual utilizar para um determinado modelo. Um software para simulação deve suportar:

- análise estocástica dos dados de entrada: em simulação discreta, provavelmente será necessário modelar certas características do sistema para obter amostragens das distribuições de probabilidade do modelo. O projetista precisa saber quais as distribuições são representativas para o sistema modelado. Isso exige que o projetista colete dados e tente ajustar esses dados a uma certa distribuição. Esse ajuste pode e deve ser feito com a ajuda do software de simulação;
- modelagem rápida: os softwares devem seguir a filosofia dos sistemas de modelagem visual interativa (VIMS - *Visual Interactive Modelling Systems*), de acordo com Pidd (1994). Tais softwares permitem que o projetista desenvolva a lógica do modelo de forma visual, através de uma interface gráfica interativa. Além disso, permite que o projetista controle a execução do modelo;
- programação do modelo de simulação: às vezes é necessário escrever seu próprio programa, podendo ser usado, para isso, uma linguagem de simulação ou uma linguagem de propósito geral, ou ainda planilhas de cálculo ou uma base de dados;

- análise estatística das saídas: nem sempre é fácil interpretar os dados resultantes da simulação. Especialmente aqueles que incluem um grande número de elementos estocásticos e muitas saídas resultantes exigem uma cuidadosa análise estatística. O software de simulação deve viabilizar essa tarefa.

Banks (1991) apresenta um estudo comparativo de quatro classes de softwares de simulação, conforme tabela 1.

Tabela 1 - Classes de Softwares de Simulação

Classes Consideração	1 Planilhas de Cálculo (Instantâneos)	2 Ferramentas de Modelagem Rápida	3 Simuladores	4 Linguagens de Simulação
Tempo de desenvolvimento	Mínimo	Mínimo	Moderado	Moderado a Substancial
Controle do modelo e complexidade do sistema	Pobre para sistemas dinâmicos	Vago/ Moderado	Deve obedecer as restrições do software	Excelente/ Virtualmente qualquer sistema complexo
Saída	Definida e desenvolvida pelo usuário	Estatisticamente adequada. Orientada a relatório	Varia de pacote para pacote. Tipicamente orientada a relatório	Relatórios definidos pelo usuário. Personalizável
Acuracidade/ Fidelidade	Geralmente não acurada para sistemas dinâmicos	Bom para planejamento grosseiro	Varia com o nível de suposições	Excelente
Treinamento	Mínimo	Moderado, geralmente usa interface do tipo planilha de cálculo	Moderado	Moderado a substancial
Ambiente mais condizente com	Operações determinísticas de sistemas estáticos	Operações probabilísticas de baixa complexidade/ Múltiplas alternativas	Complexidade média/ operações probabilísticas/ disponibilidade de dados/ aplicações específicas	Alta complexidade/ operações probabilísticas/ disponibilidade de dados/ aplicações gerais

A primeira classe consiste de planilhas de cálculo, ou planilhas eletrônicas, tais como LOTUS 1-2-3 e QUATRO-PRO. Embora elas não sejam geralmente

conhecidos por suas habilidades de simulação, é possível fazer simulação usando a função para geração de números randômicos, ou algum comando similar. Tal função gera um número aleatório, uniformemente distribuído entre zero e um. Desenvolver modelos de simulação usando planilhas de cálculo leva uma quantidade mínima de tempo, principalmente porque os sistemas são pequenos ou as aplicações limitadas. A saída é definida pelo usuário, em forma de gráfico, com resultados bem ilustrativos. As saídas se tornam atrativas devido aos diversos tipos de gráficos administrativos oferecidos pela maioria das planilhas.

A classe número dois inclui ferramentas de modelagem rápida. O objetivo dessas ferramentas é dar uma idéia sobre algumas medidas de desempenho como *throughput* e *bottlenecks* (gargalos). O sistema é modelado em termos bem gerais, omitindo vários detalhes para conseguir apenas uma idéia sobre as medidas de desempenho. Em vários exemplos, esse nível de saída é suficiente para responder a questões feitas em uma determinada fase de desenvolvimento do sistema.

A classe número três é onde se encontram os simuladores. São simulações dirigidas a dados que não necessitam programação para certos tipos de modelos. Os simuladores têm como objetivo permitir a construção do programa de simulação através de menus e gráficos, sem necessidade de programação. Entretanto, todo simulador permite programação. O modelo deve estar dentro da estrutura (*framework*) do software, pois é muito difícil modelar além do escopo fornecido. Nesse aspecto, alguns simuladores são bem mais robustos que outros. O tempo necessário para modelagem, usando um simulador, é moderado. Os problemas reais são quase sempre mais complexos que os exemplos usados para ensino.

As saídas dos simuladores são bem elaboradas. Eles produzem gráficos ricos visualmente, e criam arquivos que podem ser lidos dentro de uma planilha ou pacote gráfico. São mais fáceis para aprender e têm construções de modelagem mais afinadas com o sistema de interesse. O problema da maioria dos simuladores é que são limitados à modelagem apenas daquelas configurações de sistemas permitidas pelas suas características padrão. Um treinamento de 2 ou 3 dias é geralmente necessário para usar os softwares. Como exemplo, podem ser citados: COMNET II.5, LANNET II.5, NETWORK II.5, ProModel, SIMFACTORY II.5, WITNESS e XCELL+.

A classe número quatro se refere às linguagens de simulação. Quase todo problema real pode ser modelado usando uma linguagem de simulação, embora seja necessário um grande investimento de tempo para garantir que todos os detalhes sejam capturados. No entanto, essa é uma vantagem das linguagens de simulação, na medida em que permitem modelagem bem detalhada. As linguagens de simulação oferecem flexibilidade ilimitada de modelagem, mas, de acordo com Law; McComas (1991), exigem um especialista em programação.

A saída de uma linguagem de simulação varia de padronizada a personalizada. Mesmo os produtos populares com saída padronizada oferecem a possibilidade de personalizar a saída. São necessários treinamentos de 3 a 4 dias e, depois de trabalhar com o software por um certo tempo até se conseguir atacar um problema real, um curso adicional de 1 ou 2 dias. Exemplos de linguagens de simulação são: AutoMod II, ARENA, GPSS H ou GPSS PC, MODSIM II, SIMAN IV, SIMSCRIPT II.5, e SLAM II.

As características mais importantes em um software de simulação, de acordo com Banks (1991), estão relacionadas com as entradas, o processamento, a saída, o ambiente e o custo. Como características da entrada, o software deve:

- permitir o interfaceamento com outro software;
- ter capacidade de análise dos dados de entrada: determinar a distribuição dos dados de entrada e seus parâmetros;
- ser portátil;
- ter sintaxe clara e de fácil compreensão;
- ter flexibilidade das entradas, permitindo entrada de dados interativamente ou por arquivo;
- ter um depurador (*debugger*) interativo, permitindo o controle da simulação pelo projetista e o acesso aos dados importantes que estão sendo coletados;
- ter flexibilidade de modelagem (para linguagens de simulação) incluindo interação de processos, orientação a eventos e simulação contínua. Idealmente, uma linguagem deve permitir a seleção de qualquer uma dessas perspectivas ou combinação delas para incluir no modelo;
- ter concisão de modelagem (para linguagens de simulação), através de blocos ou nós poderosos com relação à interação de processo de uma linguagem de simulação ou comandos poderosos com relação à orientação a eventos, o que permite o desenvolvimento de modelos compactos.

Como características do processamento, o software deve:

- ter velocidade de execução;

- ser adequado ao tipo de plataforma disponível;
- permitir a manipulação de material adequado (no contexto de sistemas de manufatura - AGV, manipuladores, etc). Isso pode não ser importante na simulação genérica, onde não é exigido manipulação de material;
- possuir geradores aleatórios variados, o que inclui as distribuições básicas: exponencial, uniforme, triangular, e normal, mais distribuições empíricas;
- permitir reiniciar simulação, descarregando observações gravadas anteriormente para alcançar situação em regime;
- fazer execuções independentes, com diferentes conjuntos de números aleatórios;
- suportar o uso de atributos, que são valores locais atribuídos às transações se movendo pelo sistema;
- suportar o uso de variáveis globais, que são valores atribuídos a todas as transações se movendo pelo sistema;
- permitir programação (para simuladores), onde o usuário pode entrar com código para incorporar características especiais ao modelo;
- permitir roteamento (*routing*) condicional (para simuladores), através do envio de transições para diferentes lugares, dependendo de uma certa condição ocorrer.

Como características da saída, o software deve:

- gerar relatórios padronizados;
- gerar relatórios personalizados;

- permitir o estabelecimento de intervalos de confiança, para as medidas de desempenho desejadas, para garantir acuracidade;
- gerar gráficos analíticos;
- permitir criação de arquivo, contendo as saídas que são entradas para outro software;
- ter opção de rastreamento (*trace*), mostrando cada evento e o status da transição ativa no tempo do evento;
- fazer manutenção de base de dados, contendo os dados de saída.

Como características do ambiente, o software deve:

- ser fácil de usar;
- ser fácil de aprender;
- ter qualidade de documentação;
- possuir características de animação:
 - fácil de desenvolver;
 - qualidade da figura;
 - suavidade de movimento;
 - portabilidade para visualização remota;
 - interface CAD.
- ter ajuda on-line;
- ter tutorial on-line.

Além disso, o fornecedor do software em questão deve:

- fornecer apoio ao usuário:
 - treinamento;
 - suporte técnico;
 - atualizações e incremento.

2.7 Uma Alternativa para Análise de Dados de Simulação Utilizando AG

A simulação de sistemas, através de um modelo computacional, tem sido usada para auxiliar no projeto e operação de sistemas complexos, pois é vista como uma ferramenta poderosa de análise que, realizando experimentos sobre o modelo, permite que várias políticas sejam testadas para a seleção da melhor delas a ser adotada na prática.

O conceito básico de simulação é fácil de ser compreendido, o que é uma vantagem sobre modelos analíticos, de acordo com Shannon (1992).

Um modelo de simulação pode ter mais credibilidade quando comparado a um modelo matemático, pois seu comportamento é comparável ao do sistema real, com características reais. Entretanto, os modelos de simulação baseados em métodos tradicionais são incapazes de gerar uma solução ótima por si só. Tais modelos têm como características:

- servir como uma ferramenta para análise do comportamento do sistema, sob condições especificadas pelo usuário que faz os experimentos;
- poder ser usado para encontrar valores ótimos para um conjunto de variáveis de controle dentro de um certo conjunto de entradas.

As principais vantagens no uso de simulação são:

- pode ser usada para explorar novas políticas, procedimentos, regras de decisão, estrutura organizacional, fluxo de informação, etc., sem interromper as operações em andamento;
- podem ser testados novos projetos, programas e sistemas antes de serem liberados recursos para implementação;
- podem ser feitos testes de viabilidade;
- permite o controle de tempo, acelerando quando se quer ver eventos mais distantes, e desacelerando para focalizar um determinado fenômeno;
- permite a identificação e análise de quais são as variáveis mais importantes;
- permite a identificação de gargalos no fluxo de informações, materiais, produtos;
- permite conhecer como o sistema realmente opera;
- permite explorar questões “*what if*”, aonde reside a grande força da simulação.

As desvantagens que podem surgir no uso de simulação são:

- a construção do modelo ainda é uma arte e exige treinamento especial;
- os resultados da simulação às vezes são difíceis de serem interpretados;
- a análise da simulação pode consumir muito tempo e ser um processo caro.

As falhas mais comuns no processo de simulação, citadas por Thesen; Travis (1991), são:

- falha em estabelecer um objetivo claro;
- falha em formular uma questão possível de ser respondida;
- usar simulação quando uma abordagem mais simples seria suficiente;
- nível impróprio de complexidade;
- más suposições sobre o modelo;
- resultados da simulação mal interpretados.

Na busca de alternativas para analisar dados de simulação, principalmente em casos de sistemas complexos, surge o Algoritmo Genético (AG).

Algoritmo Genético aparece aqui como uma técnica de Inteligência Artificial, para geração e análise dos dados de simulação, que utiliza simulações e regras probabilísticas, necessitando apenas de uma forma de avaliar as soluções candidatas produzidas pelo sistema. Assim, o sistema baseado em AG, trabalha na base da tentativa e erro, utilizando alguma forma de retroalimentação do erro, e gradativamente “aprende” a produzir soluções cada vez melhores, como relata Freitas et al. (1993). AG, por suas características de simplicidade e generalidade, é usado como alternativa às técnicas convencionais de geração e análise de dados de simulação.

Capítulo 3

Princípios Básicos de Utilização de Algoritmos Genéticos

Nos últimos anos, o aparecimento crescente de problemas complexos, decorrentes do avanço tecnológico, tem impulsionado a pesquisa e utilização de novas abordagens de processamento, com o objetivo de obter solução para certas classes de problemas até então intratáveis, ou tratados de forma ineficiente por sistemas “tradicionais” de computação. Uma das áreas da Ciência da Computação que tem grande interesse na resolução desse tipo de problema é a área de Inteligência Artificial. As técnicas de Inteligência Artificial podem ser divididas em duas grandes abordagens: a simbólica e a não-simbólica.

Na abordagem simbólica o conhecimento é armazenado na máquina de maneira explícita utilizando uma linguagem simbólica. O programador deve especificar, para cada problema a ser resolvido, a forma de representação do

conhecimento (determinando as estruturas de dados, suas propriedades e inter-relacionamentos) e as regras de inferência. Isso exige um estudo, às vezes bastante profundo, sobre as peculiaridades do problema a ser resolvido.

Na abordagem não-simbólica o conhecimento é armazenado na máquina implicitamente, e o programador não precisa elaborar um programa diferente para cada problema. As técnicas usadas para essa abordagem se baseiam em simulações e regras probabilísticas, de aplicação geral, que independem do domínio do problema. Deve-se apenas dispor de uma forma de avaliar (numericamente) as soluções candidatas produzidas pelo sistema. Assim, o sistema, trabalhando na base da tentativa e erro, e utilizando alguma forma de retroalimentação do erro, gradativamente “aprende” a produzir soluções cada vez melhores, analogamente ao conceito de “experiência” associado aos seres humanos. Nesse contexto, Algoritmos Genéticos (AG) podem ser vistos como uma técnica da abordagem não-simbólica de Inteligência Artificial, sendo aplicados na resolução de problemas em diversas áreas.

A técnica de AG é um dos paradigmas de uma nova área de pesquisa dentro da Ciência da Computação, chamada Computação Evolucionária, descrita brevemente na seção 3.1. Na seção 3.2 é feita a conceituação de AG. Os princípios básicos na aplicação do AG são discutidos na seção 3.3. O processo de reprodução é apresentado na seção 3.4, onde também são descritos os principais operadores genéticos e seus parâmetros genéticos. Na seção 3.5 são apresentados os fundamentos teóricos da técnica de AG, tais como o Teorema dos Esquemas e a Hipótese dos *Buildings Blocks*, além de discutir como o AG, de acordo com Goldberg (1989), combina investigação e exploração ao mesmo tempo, e de maneira ótima.

Os aspectos práticos na utilização do AG são discutidos na seção 3.6, aspectos esse relacionados com a questão da definição da Função Objetivo (F.O.), e com a resolução dos problemas de limites do desempenho, durante o processo evolutivo, tais como a convergência prematura e a finalização lenta. São apresentadas e analisadas as técnicas de seleção mais utilizadas, bem como os métodos de substituição de indivíduos a cada geração, tais como falhas de gerações e reposição equilibrada.

Um levantamento de um conjunto de exemplos de aplicação da técnica nas mais diversas áreas é apresentado da seção 3.7, tais como otimização de funções numéricas, processamento de imagens, otimização combinatorial (problema do empacotamento, escalonamento de chão de fábrica, escalonamento de fluxo de fábrica), projeto, aprendizado de máquina (sistemas classificadores e controle) e outras aplicações (Redes Neurais, Banco de Dados, Realidade Virtual). Na seção 3.8, são apresentadas as características principais de um processo de simulação apoiada por AG e, por fim, na seção 3.9 são traçadas algumas considerações sobre essa questão.

3.1 Computação Evolucionária

A Computação Evolucionária, como definida por Heitkoetter (1993), é uma área de pesquisa e aplicação, dentro da Ciência da Computação, que está crescendo de forma acelerada. Até recentemente, essa área podia ser caracterizada como uma

coleção de grupos de pesquisa independentes, cada um usando uma determinada abordagem, e quase nenhuma interação entre si para troca de idéias.

Ultimamente, esse quadro está mudando. Houve um aumento significativo da interação entre os grupos de pesquisa, e novas idéias têm surgido, modificando antigos conceitos. O efeito dessas discussões, conforme relatado por Jong; Spears (1993), tem sido enriquecedor.

Existem três paradigmas bem definidos, de acordo com Jong; Spears (1993), que servem de base para as pesquisas sendo conduzidas na área de Computação Evolucionária: **Algoritmos Genéticos (AG)**, **Estratégias de Evolução (EE)**, e **Programação Evolucionária (PE)**. As características básicas de cada um desses paradigmas são descritas a seguir.

O **Algoritmo Genético** trabalha com uma população de indivíduos, onde cada um constitui uma lista de valores que representa uma possível solução do problema. A cada geração, esses indivíduos são avaliados quanto ao seu desempenho, ou seja, quanto à sua adaptação ao meio. Considerando essa avaliação, eles sofrem a ação de operadores genéticos, tais como seleção, cruzamento e mutação, formando uma nova população. Após um grande número de gerações, o melhor indivíduo da última população representa a solução encontrada para o problema, como explica Heitkoetter (1993).

AG se caracteriza pela ênfase na utilização de uma representação genotípica, ou seja, entendendo o cromossomo como o próprio indivíduo, sem considerar a representação fenotípica. Utiliza, tipicamente, uma representação do indivíduo através de *strings* binárias de comprimento fixo. Esses indivíduos sofrem a ação de operadores genéticos, sem a necessidade de conhecer o domínio do problema, o que

dá a característica de generalidade à técnica. Tal generalidade se reflete na ênfase dada ao fato do AG ser um sistema adaptativo e robusto, podendo ser aplicado na resolução de muitos tipos de problema. Atualmente, as aplicações de AG não se restringem apenas a representações de indivíduos através de *strings* binárias de comprimento fixo.

Estratégias de Evolução enfocam, principalmente, a construção de sistemas capazes de resolver problemas complexos de otimização baseados em valores reais. A representação do indivíduo é feita através de um vetor de números reais, que é manipulado, basicamente, por operadores de mutação. Esses operadores são projetados para provocar modificações nos valores reais dos genes, de maneira a conservar a validade desses valores como representação real dos parâmetros do problema. Atualmente, as Estratégias de Evolução também utilizam operadores de recombinação em seus sistemas.

A **Programação Evolucionária** faz a representação dos indivíduos, com ênfase no seu fenótipo, através de máquinas de estados finitos (MEF), que são capazes de responder a estímulos do ambiente. Essa técnica baseia-se em operadores de mutação, desenvolvidos para provocar mudanças comportamentais e estruturais nos indivíduos, ao longo do processo evolutivo. Essa técnica tem sido aplicada, basicamente, em problemas de prognóstico, otimização e aprendizado de máquina. Atualmente, a Programação Evolucionária utiliza outras representações, além das MEF.

Dentro da área de Computação Evolucionária, têm surgido novos paradigmas, como, por exemplo, Programação Genética, apresentada por Koza (1992), e AG “*messy*”, visto em Goldberg et al. (1993).

Todas essas técnicas, baseadas em princípios evolucionários, são classificadas como Algoritmos Evolucionários (AE). Entretanto, a técnica mais conhecida, e talvez a mais utilizada até agora, é o AG e suas variantes. Por esse motivo, muitas vezes, os termos AG e AE são usados como sinônimos, como faz Goldberg (1994).

3.2 Conceituação de Algoritmos Genéticos

Algoritmos Genéticos (AG) são métodos adaptativos que podem ser usados para resolver problemas de busca e otimização. São baseados nos processos genéticos de organismos biológicos. Considerando várias gerações, as populações naturais evoluem de acordo com o princípio da seleção natural e sobrevivência do mais apto, primeiramente estudado por Charles Darwin no livro “A Origem das Espécies”. Imitando esse processo, o AG permite a evolução das soluções de problemas do mundo real, se estes forem devidamente codificados. Por exemplo, AG pode ser usado para projetar estruturas de pontes, calcular a taxa máxima da relação entre resistência e peso, ou para determinar o menor desperdício no corte de peças de tecidos na fabricação de roupas. Podem também ser usados no controle de processos on-line, tais como em instalações químicas, ou balanceamento de carga em um sistema computacional multiprocessado.

Os princípios básicos de AG foram estabelecidos primeiro por Holland (1975), e foram discutidos por vários pesquisadores, tais como Grefenstette (1986), Goldberg (1989), Davis (1991), Beasley et al. (1993a), etc. O AG simula os processos das

populações de seres vivos, os quais são essenciais para evolução, e que aparentemente não são controlados por nenhuma regra cujos fundamentos são entendidos.

Na natureza os indivíduos em uma população competem entre si por recursos, tais como comida, água e abrigo. Além disso, os membros de uma mesma espécie geralmente competem por um parceiro. Aqueles indivíduos que são mais bem sucedidos na luta pela sobrevivência, e atraem mais parceiros, terão provavelmente um número maior de descendentes. Os indivíduos que têm um desempenho fraco vão produzir pouco ou nenhum descendente. Isso significa que os genes dos indivíduos mais aptos vão se espalhar para um número cada vez maior de indivíduos a cada geração. A combinação de boas características de diferentes ancestrais pode às vezes produzir um descendente “super-apto”, cujo desempenho é melhor do que o de seus pais. Dessa forma, as espécies evoluem para se tornar melhor e mais bem adaptadas ao ambiente.

Os AGs usam uma analogia direta do comportamento natural. Eles trabalham com uma população de estruturas, chamadas indivíduos ou *strings*, que representam possíveis soluções para um determinado problema. A cada indivíduo é atribuído um valor de desempenho ou aptidão (*fitness*), de acordo com a sua qualidade como solução para o problema, como explica Grefenstette (1986). Com base nessa avaliação, uma nova população de soluções é formada, através do uso de operadores genéticos específicos.

Basicamente, esses operadores combinam a sobrevivência dos mais aptos dentre a população de indivíduos com uma troca aleatória, mas estruturada, de informações entre esses indivíduos de modo que, a cada geração, um novo conjunto de indivíduos é criado, usando partes dos indivíduos mais aptos da geração anterior.

Assim, serão produzidos novos indivíduos como descendentes, que herdam algumas características de cada um dos pais. Os elementos menos aptos não são escolhidos para reprodução e são descartados.

Uma nova população de possíveis soluções é formada, selecionando os melhores indivíduos da geração atual, e emparelhando-os para produzir um novo conjunto de indivíduos. Essa nova geração contém uma proporção maior das boas características dos melhores elementos da geração anterior. Dessa forma, depois de várias gerações, as boas características são espalhadas pela população, sendo misturadas e trocadas com outras boas características, e assim por diante. Favorecendo o emparelhamento dos indivíduos mais aptos, as áreas mais promissoras do espaço de busca são pesquisadas. Se o AG for bem projetado, a população vai convergir para a solução ótima do problema. Após um grande número de gerações, que é o critério de parada do AG, o indivíduo mais apto da última geração é a solução encontrada pelo algoritmo para o problema, de acordo com Goldberg (1989).

A potencialidade do AG vem do fato da técnica ser robusta, segundo Goldberg (1989), podendo ser aplicada em uma vasta gama de problemas, inclusive àqueles problemas que dificilmente conseguem ser resolvidos através de outros métodos de resolução. AG não garante encontrar a melhor solução para o problema, mas é geralmente bom para encontrar uma solução razoavelmente boa para o problema, de maneira razoavelmente rápida, como afirma Beasley et al. (1993a). Quando existem técnicas especializadas para resolução de um determinado problema, elas quase sempre apresentam melhor desempenho que o AG, tanto relacionado à velocidade quanto à precisão do resultado final. Sendo assim, a proposta é aplicar AG em áreas difíceis, onde não existem tais técnicas. Mesmo em áreas onde existem

técnicas, que podem ser aplicadas de maneira satisfatória, o AG tem sido aplicado em conjunto com essas técnicas, aumentando seu desempenho.

3.3 Princípios Básicos na Aplicação do Algoritmo Genético

O AG padrão pode ser descrito pelo algoritmo apresentado na figura 4.

```

Início /* algoritmo genético */
    gerar população atual
    calcular desempenho de cada indivíduo
    nova população := vazia
    converge := falso
    Enquanto não converge faça
        Início /* ciclo de reprodução */
            Enquanto população atual não está vazia faça
                Início
                    selecione dois indivíduos i1 e i2 da população atual para emparelhamento
                    /* priorizando os indivíduos mais aptos */
                    faça cruzamento de i1 e i2 para gerar dois descendentes i'1 e i'2
                    verifique a possibilidade de ocorrer mutação sobre i'1 ou i'2
                    calcule o desempenho dos dois descendentes
                    nova população ← nova população ∪ {i'1, i'2}
                Fim
            Se a população convergiu para a solução ótima
                então converge := verdade
        Fim
    Fim

```

Figura 4 - Algoritmo Genético Tradicional

Esquemáticamente, a execução do AG é representada na figura 5.

Figura 5 - Esquema de Aplicação do Algoritmo Genético

Para resolver um problema através de AG, deve-se considerar que:

- Cada solução candidata é representada por um indivíduo, que é uma *string* definida usando um alfabeto finito, de modo que cada *string* representa um conjunto de valores para o conjunto de parâmetros do problema. Cada posição da *string* representa um gene, como visto em Beasley et al. (1993a). Essa *string*, ou indivíduo, também é chamada de cromossomo.
- A aptidão do indivíduo, ou seja, sua capacidade de adaptação ao meio, é um valor não negativo da qualidade da solução representada pelo indivíduo. Quanto maior o valor de sua aptidão, melhor é a solução representada por esse indivíduo. Assim, a aptidão na terminologia de AG corresponde ao valor da função objetivo na terminologia de Pesquisa Operacional (referente a problemas de maximização). A aptidão de um indivíduo pode ser vista como o

desempenho que esse indivíduo apresenta na resolução do problema. O sucesso do AG depende da escolha adequada da medida de aptidão, ou seja, da função objetivo, conforme estudo de Tam (1992). Essa tarefa nem sempre é simples e fácil, de acordo com Beasley et al. (1993a).

Se o AG foi implementado corretamente, a população vai evoluir durante as sucessivas gerações, de tal maneira que o desempenho do melhor indivíduo e a média de desempenho da população cresce em direção ao valor ótimo global. A convergência significa a progressão na direção do aumento da uniformidade. Um gene tem convergência quando 95% da população têm esse mesmo valor de gene. A população converge quando todos os genes têm convergência. Em uma aplicação típica de AG, o desempenho varia como mostra a figura 6. Nessa figura, cada valor do eixo “geração” corresponde à geração que está sendo analisada. O par de barras representado em cada intervalo do mesmo eixo corresponde, respectivamente, ao valor do desempenho médio daquela geração e ao valor do desempenho do melhor indivíduo daquela geração. À medida que a população converge, o desempenho médio cresce e se aproxima do desempenho do melhor indivíduo.

Figura 6 - Um Exemplo de Convergência de Desempenho em Aplicação de AG

3.4 Reprodução

Durante a fase reprodutiva do AG, os indivíduos são selecionados na população e recombinados, produzindo descendentes que formarão a próxima geração. Os pais são selecionados aleatoriamente, utilizando um esquema que favorece os indivíduos mais aptos. Os bons indivíduos provavelmente serão selecionados várias vezes em uma geração, e os indivíduos mais fracos provavelmente não serão selecionados.

Para garantir a monotonicidade no valor do desempenho do melhor indivíduo durante o processo evolutivo, é aplicada a reprodução elitista. Esse procedimento implica na preservação dos melhores indivíduos da população, que passam inalterados para a próxima população, garantindo que o valor do desempenho do melhor indivíduo da população seguinte nunca será menor que o desempenho dos melhores indivíduos das gerações anteriores. De acordo com Davis (1991), essa estratégia aumenta o desempenho do AG.

Para a realização dessas tarefas na reprodução, são aplicados os operadores genéticos, sendo que cada um deles é responsável por uma das etapas do ciclo reprodutivo. Os operadores genéticos básicos são: seleção, cruzamento e mutação. Esses operadores são descritos a seguir.

3.4.1 Operadores Genéticos

Os operadores *seleção* e *cruzamento* podem ser considerados como os principais operadores genéticos, devido à sua ampla utilização e importância para os fundamentos teóricos do AG. A *mutação* tem uma importância menor em relação aos outros operadores, mas também é bastante utilizada. Esses três operadores são descritos a seguir.

Seleção: é um operador no qual os indivíduos são copiados para formar a população da geração seguinte, em quantidades proporcionais à aptidão de cada um, ou seja, ao valor da função objetivo associado a cada um. As cópias produzidas são submetidas à ação de outros operadores genéticos, para a geração de novos

indivíduos. Uma das formas de seleção mais simples e mais usada é a seleção pela regra da roleta. Essa seleção simula o procedimento de girar uma roleta viciada, onde cada indivíduo da geração atual tem uma fatia da roleta, sendo o tamanho dessa fatia proporcional à aptidão do indivíduo. Assim, quanto mais apto o indivíduo, maior a probabilidade de ser selecionado. A figura 7 ilustra a criação dessa roleta. Entretanto, se não for feito nada além da seleção, a trajetória da população seria constituída apenas de modificações das proporções das *strings* contidas na população inicial. O algoritmo precisa ter meios para explorar diferentes estruturas, de acordo com Koza (1992) e Goldberg (1994).

Figura 7 - Indivíduos de uma População e sua Correspondente Roleta de Seleção

Cruzamento: é um operador genético primário de exploração de estruturas. Objetiva proporcionar a troca de *substrings* de maneira satisfatória. A aplicação do operador cruzamento básico acontece em três etapas. Primeiro, dois indivíduos, que se constituem um par, são escolhidos na população selecionada. Um ponto de cruzamento ao longo da *string* é escolhido aleatoriamente e os valores dos genes, a

partir dessa posição, são trocados entre as duas *strings*. Um exemplo é visto na figura 8.

Figura 8 - Exemplo de Aplicação do Operador Cruzamento

Essas duas novas *strings* são inseridas na população. O processo continua, emparelhando os indivíduos e gerando seus filhos até a nova população estar completa, preenchida com os filhos, que são construídos a partir dos pedaços e fatias dos "bons" pais (selecionados). O cruzamento pode ser realizado sobre alguns indivíduos, quando se especifica uma taxa de cruzamento, de acordo com Goldberg (1989).

Mutação: É um operador que modifica o valor de algum gene, escolhido aleatoriamente, de um indivíduo, como ilustrado na figura 9. Tal operador pode ser visto como uma política de proteção contra a perda de diversidade genética, ou seja, contra a não exploração de alguma região do espaço de busca. Além disso, esse operador dá ao AG a característica de ser uma técnica *hill-climbing*, ou seja, permite o reconhecimento de ótimos locais, como afirma Goldberg (1994). Assim a mutação tende a aumentar a diversidade genética da população.

Pode-se observar que nem a seleção e nem o cruzamento criam novos valores de genes. Assim, se uma população não possui um valor em uma determinada

posição, e esse valor é essencial para se obter uma boa solução, somente o operador mutação pode produzir esse valor de gene nessa posição, destaca Goldberg (1989). Tipicamente, a taxa de mutação é muito baixa, na ordem de uma mutação a cada mil genes.

Antes da Mutação:	1 1 0 0 0
Depois da Mutação:	1 1 0 0 0

Figura 9 - Exemplo de Aplicação do Operador Mutação

Estudos indicam que, no início do processo evolutivo, o operador de cruzamento é muito importante e, portanto, a taxa de cruzamento deve ser alta. No decorrer do processo, devido a homogeneidade genética entre os indivíduos, esse operador perde sua importância gradativamente. Pela mesma razão, ou seja, pela diversidade genética da população no início do processo evolutivo e pela homogeneidade genética crescente dos indivíduos no decorrer do processo, o operador de mutação ganha importância gradativamente. Nesse contexto, sugere-se que à medida que taxa de cruzamento vá decrescendo durante a evolução das populações, a taxa de mutação vá crescendo, conforme estudos de Grefenstette (1986), Goldberg (1989), Davis (1991), Koza (1992), Goldberg et al. (1992), Beasley et al. (1993b), Heitkoetter (1993), e Fang et al. (1993).

3.4.2 Parâmetros Genéticos

A representação de sistemas complexos pelos AGs pode ser formalizada através de alguns parâmetros genéticos, de acordo com Goldberg (1989), tais como:

- **Tamanho da população:** este parâmetro afeta a eficiência do AG. Uma população pequena geralmente faz com que o AG tenha um desempenho fraco, pois a população muito provavelmente não irá cobrir suficientemente o espaço de busca da solução do problema. Quanto maior a população, maior é a sua representatividade do domínio do problema, conforme destaca Goldberg et al. (1992). Além disso, uma população grande o suficiente impede uma convergência prematura para soluções locais, ao invés de soluções ótimas.
- **Taxa de Cruzamento:** a frequência da aplicação do operador cruzamento é controlada pela taxa de cruzamento. Quando a taxa de cruzamento tende ao valor máximo, ou seja, taxa de cruzamento igual a 1, mais rapidamente novos indivíduos são introduzidos na população. Uma taxa de cruzamento baixa pode estagnar a busca da solução ótima. Tipicamente, o valor dessa taxa varia de 0.6 a 1, de acordo com Beasley et al. (1993a). Se o cruzamento não é aplicado, os pais são duplicados para a geração seguinte.
- **Taxa de Mutação:** para aumentar a diversidade da população, o operador mutação é muito usado, mas sempre com uma taxa pequena, pois, do contrário, a aleatoriedade da população aumenta muito, impedindo a sua convergência para uma solução ótima.

- **Taxa de Elitização:** define a porcentagem de indivíduos que passarão inalterados através das gerações. Geralmente, o valor dessa taxa é de 0.01, ou seja, se a população tem cem indivíduos, a cada geração um indivíduo (que apresenta o melhor desempenho) passa inalterado para a próxima geração.

3.5 Fundamentos Teóricos de AG

Muitas vezes a viabilidade da aplicação dos AGs esbarra no fato de que esses algoritmos também são sistemas complexos, conforme discute Davis (1991), que se comportam de maneira diferente nas diferentes fases de sua aplicação, sendo que cada operador tem sua importância enfatizada em cada uma dessas fases, como destaca Beasley et al. (1993a). Os AGs mais simples são discretos, não lineares, estocásticos, altamente dimensionais, com características de paralelismo implícito, que operam sobre uma grande variedade de tipos de problemas. Isso dificulta a compreensão de sua semântica, como explica Goldberg (1993). Os AGs são difíceis de projetar e analisar, e os desenvolvedores de software têm que tomar muitas decisões de projeto antes de conseguir uma aplicação bem sucedida. Isso acontece porque ainda não existe uma teoria bem fundamentada, consolidada e consagrada de computação genética. A implementação de um AG que seja altamente eficiente ainda é um desafio de projeto, como discutido em Goldberg (1994).

A maioria das pesquisas com AG se concentra na busca de regras empíricas para entender porque o AG tem um bom desempenho. Não há uma “teoria geral” aceita, que explique exatamente porque o AG tem as propriedades que tem.

Entretanto, várias hipóteses têm sido levantadas, explicando parcialmente o sucesso do AG. Essas hipóteses podem ajudar na implementação bem sucedida do AG.

3.5.1 Uma Abordagem Informal a Esquemas e ao Teorema dos Esquemas

O teorema dos esquemas de Holland (1975) foi a primeira explicação sobre o funcionamento do AG. Um esquema é um padrão de valores de genes, que pode ser representado, considerando o alfabeto binário, por uma *string* de caracteres no alfabeto $\{0,1,\#\}$. Um indivíduo tem um determinado esquema se ele combina aquele esquema, onde o símbolo “#” representa qualquer um dos outros símbolos do alfabeto. Por exemplo, o indivíduo “1010” contém, entre outros, os esquemas “10##”, “#0#0”, “##1#”, e “101#”. A ordem de um esquema é o número de símbolos não-# que ele contém (no exemplo: 2, 2, 1, 3, respectivamente). O comprimento de um esquema é definido como o número de símbolos entre os símbolos não-# mais distantes, incluindo aqueles (no exemplo: 2, 3, 1, 3, respectivamente).

O teorema dos esquemas explica a potencialidade do AG em termos da forma de processamento dos esquemas. Os indivíduos na população têm oportunidades de se reproduzir, gerando descendentes. O número de oportunidades que um indivíduo tem é proporcional à sua aptidão. Assim, o melhor indivíduo contribui mais com seus genes para a próxima geração. Assume-se que um indivíduo tem um alto desempenho pelo fato de seus esquemas serem bons. Passando mais desses bons esquemas para a próxima geração, aumenta a probabilidade de encontrar melhores soluções.

Holland mostrou que o melhor caminho para explorar o espaço de busca é dar oportunidade de cruzamento aos indivíduos, proporcionalmente ao seus desempenhos com relação ao resto da população. Dessa forma, os bons esquemas receberiam um número exponencialmente crescente de cópias nas sucessivas gerações. É isso o que afirma o teorema dos esquemas. Holland também mostrou que, em virtude dos indivíduos terem um grande número de esquemas diferentes, o número de esquemas que são processados a cada geração é da ordem de n^3 , onde n é o tamanho da população. Essa propriedade é chamada de paralelismo implícito, sendo uma das explicações para o bom desempenho do AG.

3.5.2 Hipótese dos *Building Blocks*

Goldberg (1989) diz que o poder do AG está no fato da técnica ser capaz de encontrar bons *building blocks*, que são esquemas de comprimento pequeno. Esses *building blocks* são bits que trabalham bem juntos, e tendem a gerar uma melhora no desempenho do indivíduo que o contém. Um esquema de boa codificação é aquele que estimula a formação de *building blocks*, garantindo que:

1. os genes relacionados estão bem próximos no cromossomo, e
2. há pouca interação entre genes.

A interação (também chamada de epistasia) entre os genes significa que a contribuição de um gene para o desempenho do indivíduo depende dos valores de outros genes. Se essas duas regras são observadas na aplicação da técnica, então o AG será tão eficaz quanto o previsto pelo teorema dos esquemas.

Essas condições nem sempre são fáceis de serem encontradas. Na maioria das vezes, a forma exata do relacionamento entre os genes não é conhecida. Assim, mesmo que haja apenas relacionamentos simples, pode ser impossível organizar a codificação para refletir isso.

A condição 2 é uma pré-condição para 1. Se a contribuição de cada gene para o desempenho geral for independente de todos os outros genes, então é possível resolver o problema otimizando um gene de cada vez, o que, geralmente, não é possível. Se for possível garantir que cada gene interage com apenas alguns outros genes, e que eles podem ser colocados juntos na codificação, então as condições 1 e 2 são satisfeitas. Por outro lado, se há muita interação entre os genes, então as condições não são satisfeitas.

Como essas recomendações, feitas por Goldberg, garantem que o AG vai trabalhar da melhor maneira possível, deve-se tentar projetar os esquemas de acordo com elas. Entretanto, duas questões pertinentes podem ser feitas:

1. É possível encontrar esquemas que atendem as recomendações das hipóteses de *building blocks*? (e se for possível, como eles podem ser encontrados?)
2. Se não for possível encontrar tais esquemas ideais, o AG pode ser modificado para aumentar seu desempenho nessas circunstâncias? (e se for possível, como fazê-lo?)

A busca por respostas a essas questões tem se evidenciado como importantes tópicos de pesquisa.

3.5.3 Investigação e Exploração

Qualquer algoritmo de otimização eficiente deve usar duas técnicas para encontrar o máximo global: investigação, para pesquisar áreas novas e desconhecidas no espaço de busca, e exploração, para usar o conhecimento obtido em pontos previamente visitados, na busca de pontos melhores. Esses dois requisitos são contraditórios, e um bom algoritmo de busca deve encontrar um equilíbrio entre eles.

Uma busca puramente aleatória é boa para investigação, enquanto que um método puramente *hill-climbing* é bom na exploração, mas faz pouca investigação. Combinações dessas duas estratégias podem ser muito eficazes, mas é difícil saber qual é o melhor equilíbrio entre as duas.

Goldberg (1989) mostra que o AG combina investigação e exploração ao mesmo tempo, e de maneira ótima. Entretanto, embora isso seja teoricamente verdade, existem problemas na prática, pois algumas condições devem ser impostas:

1. uma população infinita,
2. a função objetivo deve refletir exatamente a utilidade da solução, e
3. os genes não devem interagir de maneira significativa.

A condição 1 não pode ser satisfeita, na prática. Sendo assim, o desempenho do AG sempre estará sujeito a erros estocásticos. Esse problema, que também existe na natureza, é chamado de declínio genético.

Mesmo que não haja qualquer pressão através da seleção, ainda assim os membros da população vão convergir para algum ponto do espaço de busca. Isso acontece por causa do acúmulo de erros estocásticos. Se, por acaso, um gene começa a predominar na população, é bem provável que se torne mais predominante ainda na próxima geração. Se esse aumento de predominância se mantém ao longo de várias gerações, e a população é finita, esse gene se espalha por todos os membros da população. Quando isso acontece, ele se torna fixo, e o cruzamento não pode introduzir novos valores de genes. No decorrer das gerações, como um efeito “bola de neve”, todos os genes passam a ter valores fixos, o que garante a convergência.

A taxa de declínio genético é uma medida do limite inferior da taxa com que o AG pode convergir para a solução correta. Essa taxa pode ser reduzida aumentando-se a taxa de mutação. Entretanto, se a taxa de mutação é muito alta, a pesquisa se torna muito aleatória, o que também prejudica o desempenho do AG.

As condições 2 e 3 podem ser satisfeitas em testes de laboratórios, mas são difíceis de serem satisfeitas em problemas do mundo real.

3.6 Aspectos Práticos na Utilização do AG

Existem várias técnicas de seleção de indivíduos para cruzamento. Basicamente, essas técnicas visam superar problemas relacionados à função objetivo.

3.6.1 Função Objetivo

A função objetivo, juntamente com o esquema de codificação, são o aspecto mais importante para a aplicação do AG. De acordo com Grefenstette (1986), esses dois parâmetros, função objetivo e esquema de codificação, são críticos com relação ao desempenho do AG.

Para muitos problemas, não é possível definir uma função objetivo ideal. Se o AG (ou qualquer outra técnica de busca) vai ter um bom desempenho, isso dependerá da construção da função objetivo, que não deve ter muitos máximos locais, nem um máximo global muito isolado.

A regra geral é que a função objetivo deve refletir o valor do indivíduo de alguma maneira “real”. Em muitos problemas, essa tarefa de construção da função objetivo é óbvia. Entretanto, o valor “real” de um indivíduo pode não ser uma medida útil na busca genética, como, por exemplo, em problemas que apresentam restrições com relação aos valores de alguns resultados, ou seja, problemas onde um indivíduo deve ter um desempenho dentro de um certo conjunto de valores de desempenho. Qualquer outro valor fora desse conjunto torna o indivíduo inválido como solução para o problema. Nesses casos, devem-se ter meios de medir o quanto o valor do desempenho de um indivíduo é bom para guiar a busca até um valor de desempenho dentro do conjunto de valores estabelecido como válido.

Uma abordagem usada com esse fim é a adoção de uma **política de penalização**, que representa o quanto um indivíduo é ruim, e seu desempenho é

definido em função dessa medida, como pode ser visto em Goldberg (1989). Richardson et al. (1989) ensina a definir uma política de penalização. Basicamente, as políticas de penalização baseadas no quanto as restrições foram violadas são melhores do que aquelas políticas baseadas em quantas restrições foram violadas. Podem ser definidas boas políticas de penalização tendo como base o custo de transformação esperado, ou seja, se um indivíduo é inválido, quanto custará torná-lo válido? De acordo com Beasley et al. (1993a), ainda há muita pesquisa a ser feita nessa área.

Quando a função objetivo é muito complexa, pode-se usar a técnica de **avaliação aproximada de função**. Se uma outra função, mais rápida e simples, pode ser aplicada e resulta em um valor aproximado do valor da verdadeira função objetivo, o AG pode encontrar indivíduos melhores, em um tempo de CPU menor do que aplicando a função objetivo real, afirma Goldberg (1989). De acordo com Beasley et al. (1993a), o AG é robusto o suficiente para convergir, mesmo com o “ruído” decorrente do uso de uma função aproximada de avaliação.

3.6.2 Problemas de Limites do Desempenho

Na execução do AG, os valores dos genes dos indivíduos da população inicial são gerados aleatoriamente. Conseqüentemente, existe uma grande variação de desempenho entre esses indivíduos. No decorrer do processo evolutivo, certos valores, para cada gene, começam a predominar. À medida que a população converge, os limites dos valores de desempenho da população são reduzidos. Essa

variação dos limites do desempenho, durante o processo evolutivo, pode levar a problemas de convergência prematura e finalização lenta.

3.6.2.1 Convergência Prematura

Um problema clássico que ocorre em algumas aplicações de AG é que os genes de um grupo de indivíduos de desempenho comparativamente alto (mas não ótimo) podem dominar a população rapidamente, provocando a convergência para um ótimo local. Essa situação pode eliminar a capacidade do AG de buscar melhores soluções, pois o cruzamento entre indivíduos quase idênticos é sempre ineficiente. Nesse caso, somente com o operador mutação pode-se conseguir explorar novas áreas do espaço de busca, mas a pesquisa se torna lenta e aleatória, com ressalta Beasley et al. (1993a).

O teorema dos esquemas diz que os melhores indivíduos devem ter mais oportunidades de gerar descendentes, proporcionalmente aos valores de seus desempenhos. Entretanto, quando aplicado, tal teorema provoca convergência prematura, pois a população é finita. Para resolver esse problema, Beasley et al. (1993a) sugere que o processo de seleção dos indivíduos para cruzamento deve ser modificado, controlando o número de oportunidades dada a cada indivíduo, de tal forma que esse número não seja nem muito grande e nem muito pequeno. Assim, diminui-se os limites dos valores de desempenho, prevenindo a dominância de um indivíduo muito apto já no início do processo evolutivo.

3.6.2.2 Finalização Lenta

Esse problema é o inverso do problema da convergência prematura. Depois de várias gerações, a população converge, mas pode não localizar de forma precisa o máximo global. O desempenho médio é alto, e a diferença entre os desempenhos dos indivíduos é pequena. Quando isso acontece, implica que há um gradiente insuficiente na função objetivo para que o AG chegue ao resultado ótimo.

As mesmas técnicas de seleção, usadas para combater convergência prematura, são usadas para resolver esse problema, de acordo com Beasley et al. (1993a), através da expansão dos limites dos valores do desempenho na população. Como na convergência prematura, o escalonamento dos valores de desempenho pode levar a uma supercompressão (ou subexpansão) por causa de apenas um indivíduo ruim.

3.6.3 Técnicas de Seleção

A seleção implica dar oportunidades reprodutivas para cada indivíduo. Em princípio, cada indivíduo da população é copiado para um “*pool* de pais”, onde os indivíduos mais aptos têm maior probabilidade de receber mais de uma cópia, e os indivíduos menos aptos podem não receber nenhuma cópia. Através de um esquema de reposição geracional, o tamanho do *pool* de pais é igual ao tamanho da população. Em seguida, os pares de indivíduos são formados de forma aleatória, e sofrem cruzamento. Essa operação se repete até que o *pool* esteja vazio.

O comportamento do AG depende muito da forma como os indivíduos são escolhidos para serem colocados no *pool* de pais. Existem duas classes de métodos para se fazer isso: remapeamento explícito de desempenho e remapeamento implícito de desempenho.

- remapeamento explícito de desempenho: faz-se uma escala de desempenhos da população, e mapeia-se essa escala em uma nova escala. Esses valores remapeados são usados como o número de cópias que o indivíduo terá no *pool* de pais (número de tentativas reprodutivas), considerando o mesmo tamanho de população. Como exemplos desses métodos, podem ser citados o escalonamento de desempenhos, o enquadramento (*windowing*) de desempenhos e a classificação de desempenhos, vistos em Davis (1991) e Beasley et al. (1993a);
- remapeamento implícito de desempenho: de efeito similar ao remapeamento explícito, mas sem o passo intermediário de calcular um desempenho modificado. Como exemplo, pode ser citado o método de seleção por sorteio (regra da roleta), visto em Goldberg (1989) e Beasley et al. (1993a).

De acordo com Beasley et al. (1993a), todos esses métodos, com o ajuste adequado de parâmetros, podem apresentar desempenhos similares. Portanto, não existe um método “melhor”.

3.6.4 Falhas de Gerações e Reposição Equilibrada

A falha de geração é definida como a proporção de indivíduos na população que são trocados a cada geração. A maioria dos trabalhos usa uma falha de geração

igual a 1, ou seja, toda a população é trocada a cada geração, devido aos resultados da pesquisa de Grefenstette (1986).

Entretanto, tendências mais recentes favorecem a reposição equilibrada, como Davis (1991), que operam no outro extremo onde, a cada geração, apenas alguns indivíduos são trocados (geralmente dois). Os esquemas mais conhecidos, para a escolha dos indivíduos que serão descartados e dos indivíduos que serão cruzados, são:

- seleção de pais de acordo com o melhor desempenho e seleção aleatória para reposição;
- seleção aleatória de pais e seleção para reposição de acordo com o pior desempenho;
- seleção de pais e seleção para reposição de acordo com o melhor e o pior desempenho.

A principal diferença entre o AG convencional de reposição geracional e o AG de reposição equilibrada é que, no AG de reposição equilibrada, as estatísticas da população (tal como o desempenho médio) são recalculadas depois de cada cruzamento (o que não é computacionalmente dispendioso), e os descendentes estão imediatamente disponíveis para reprodução. Isso pode acelerar, pelo menos inicialmente, a busca do melhor indivíduo no espaço de busca.

Estudos citados por Beasley et al. (1993a) não encontraram evidências de que o AG de reposição equilibrada é melhor que o AG convencional de reposição geracional. Entretanto, Davis (1991) comenta que alguns pesquisadores tentaram usar

reposição equilibrada e concluíram que essa forma de seleção tem desempenho inferior comparado ao AG de reposição geracional.

Embora Beasley et al. (1993a) associe o tempo de convergência do AG, para a solução ótima, com a técnica de seleção empregada, Goldberg (1993) diz que a conexão entre o algoritmo de seleção e o tempo de convergência ainda não está bem compreendida.

3.7 Aplicações com AG

Goldberg (1994) defende o uso de AG para uma vasta maioria de aplicações por ser uma técnica muito interessante, apresentando diversos atrativos como:

1. Podem resolver rapidamente, e de maneira confiável, problemas difíceis.
2. São fáceis de serem integrados a simulações e modelos.
3. Podem ser estendidos.
4. São fáceis de hibridar.

Uma das principais razões para usar AG é que eles são algoritmos altamente eficientes. Os resultados de trabalhos empíricos há tempos sugerem isso, mas só agora estudos teóricos estão evidenciando esse fato. O AG pode resolver problemas que tenham muitas soluções ótimas difíceis de encontrar. Além disso, como AG trabalha com amostragem, a população pode ser dimensionada para conter uma certa diversidade genética, o que diminuiria a quantidade de erros possíveis de acontecer, ressalta Goldberg et al. (1992). Isso torna o AG excepcionalmente tolerante a falhas.

Como o AG exige muito pouca informação específica do problema, esses algoritmos se tornam extremamente fáceis de serem integrados ao código de uma aplicação existente. Além disso, como o AG é tolerante a ruído, as simulações por eventos discretos e outros avaliadores "ruidosos" podem ser usados diretamente, se o dimensionamento da população é feito para levar em conta as variações estocásticas do processo de avaliação.

Mesmo quando os princípios básicos do AG não se aplicam totalmente à resolução de um determinado problema, é sempre possível fazer adaptações, de acordo com a natureza do problema em questão, através da definição de operadores genéticos específicos, baseados no conhecimento sobre o domínio desse problema. Isso vai ajudar a formar um AG híbrido ou baseado em conhecimento. Por exemplo, muitos domínios de busca têm heurísticas de busca local mais eficientes que seleção e mutação. Pegar a melhor solução em menor tempo implica geralmente combinação da abordagem do AG em uma primeira instância e, a seguir, uma técnica local específica do problema. Existem várias formas de inserir informações sobre o problema nos operadores ou na codificação, de acordo com Goldberg (1989) e Davis (1991).

3.7.1 Exemplos de Aplicações

Alguns exemplos de aplicações de AG já foram mencionados no início desse capítulo. Entretanto, para ilustrar a flexibilidade do AG, são listados aqui alguns trabalhos, a maioria de natureza prática, e outros ainda como tópicos de pesquisa.

- **Otimização de funções numéricas.** As pesquisas tradicionais com AG se concentram nessa área. Os AGs têm se mostrado adequados e, inclusive, obtém melhores resultados que técnicas convencionais de otimização em funções “ruidosas”, descontínuas, multimodais, e difíceis. Como exemplo, podem ser citados trabalhos de Tam (1992), Goldberg et al.(1993), Dávalos; Stange (1994), McKinney; Lin (1994), Reid (1994 a), Reid (1994 b), e Ritzel et al. (1994).
- **Processamento de Imagens.** Como acontece com raios-X médicos ou imagens de satélites, é comum a necessidade de alinhar duas imagens, captadas em tempos diferentes. Através de comparações de sobreposições aleatórias de pontos sobre as duas imagens, o AG pode descobrir, de forma eficiente, um conjunto de equações para ajustar uma imagem com relação à outra, como descrito em Goldberg (1989). Outro caso é a análise de imagens de satélites para detectar alvos bélicos, como cita Goldberg (1994). Uma outra aplicação ainda é o uso de AG no reconhecimento de criminosos através de fotografias de suspeitos, como descrito em Caldwell; Johnston (1991). O AG gera um conjunto de faces aleatoriamente, e a testemunha seleciona as imagens que são mais parecidas com o suspeito. Essas imagens são usadas para gerar mais faces para a próxima geração. A testemunha atua como a “função objetivo” do AG, e controla a convergência para a imagem correta.
- **Otimização Combinatorial.** Essas aplicações exigem soluções para problemas que envolvem combinações de objetos discretos. Elas são muito parecidas com otimização de funções, onde são exigidas técnicas diferentes de codificação, recombinação, e função objetivo. Provavelmente, o problema mais estudado é o **problema do caixeiro viajante** e suas variantes, como visto em Goldberg; Smith

(1986), Goldberg; Kuo (1987), Liepins et al. (1987), Glover; Greenberg (1989), Gorges-Schleuter (1989), Horn et al. (1992), Levitin; Rubinovitz (1993), e Simpson et al. (1994). O objetivo é encontrar a menor rota para visitar um determinado grupo de cidades ou lugares, determinando-se uma rota ótima entre os vários lugares. O **problema do empacotamento** também se caracteriza como de otimização combinatorial e consiste na determinação da organização de um conjunto de objetos em um determinado espaço, sendo um problema típico da indústria. Esse problema tem sido muito estudado, como visto em Goldberg (1986), Goldberg; Smith (1986), Kröger et al. (1991), Freitas et al. (1993), e Carneiro et al. (1994). Outro problema correlato é o **escalonamento de chão de fábrica**, onde o propósito é alocar, de maneira eficiente, um conjunto de recursos (máquinas, pessoal, salas, instalações) para executar um conjunto de tarefas, como, por exemplo, a produção de vários lotes de componentes de máquinas. Existe, também, o **escalonamento de fluxo de fábrica**, que consiste na alocação do programa de execução das tarefas na produção buscando a maneira ideal. Existem restrições óbvias: a mesma máquina não pode ser usada para fazer duas coisas diferentes ao mesmo tempo. A alocação ótima considera o tempo mínimo para execução total ou o tempo mínimo de espera por recursos. Como exemplo, podem ser citados os estudos feitos por Cleveland; Smith (1989), Whitley et al. (1989), Bagchi et al. (1991), Davis (1991), Lawton (1992), Bruns (1993), Fang et al. (1993), Freitas et al. (1994), e Fraser (1995).

- **Projeto.** As tarefas de projeto podem ser uma mistura de otimização combinatorial e de funções. Como exemplo, pode ser citados aplicações CAD (*Computer Aided Design*) ou projeto de redes de computadores. AG para projeto pode ser

hibridado com sistemas especialistas ou de otimização mais tradicionais, com o objetivo de permitir uma gama maior de projetos que um projetista pode avaliar. Goldberg (1994) cita alguns exemplos de pesquisas nessa área, além de Goldberg (1991).

- **Aprendizado de Máquina.** Existem várias aplicações de AG em aprendizado de máquina e **sistemas classificadores**. O AG tenta evoluir um conjunto de regras *if...then* para alcançar uma certa situação. Essa técnica tem sido aplicada em teoria de jogos, modelagem política e econômica, como cita Davis (1991). Outros exemplos de trabalhos na área são Booker et al. (1989), McCallum; Spackman (1990), Oliver (1994), e Twardowski (1994). O uso principal de técnicas de aprendizado de máquina tem sido no campo de **controle**. É um sistema complexo, como uma indústria química, existem muitos parâmetros de controle a serem ajustados, a fim de que o sistema trabalhe de maneira ótima. Geralmente é aplicada a abordagem de sistemas classificadores, sendo que as regras para controlar o sistema devem ser criadas a cada aplicação. O desempenho de um conjunto de regras pode ser avaliado através da análise do desempenho do sistema real ou da simulação de um modelo computacional. Alguns trabalhos nessa área são Forgaty (1988), Goldberg (1989), Karr (1991), Sikora (1992), Carse; Forgaty (1994), Pipe et al. (1994), Porter; Passino (1994), e Then; Chong (1994).
- **Outras Aplicações.** A cada dia aparecem novas áreas de aplicação de AG. Como exemplo, podem ser citados os trabalhos de treinamento de **Redes Neurais** com AG, desenvolvidos por Garis (1990) e Murray (1994), a aplicação de AG no processo de indexação de **Banco de Dados**, visto em Celko (1993), ou uso da AG

na criação da morfologia de criaturas e controle de sua força muscular em ambiente de **Realidade Virtual** feito por Sims (1994).

3.7.2 Ambientes de Programação com AG

Ribeiro Filho; Treleaving (1994) apresentam uma taxonomia para ambientes de programação com AG. Nesse trabalho, são definidas três classes de ambientes de programação: sistemas orientados a aplicações, sistemas orientados a algoritmos e conjunto de ferramentas (*tool kits*).

- **Sistemas orientados a aplicações** são do tipo “caixa preta”, ocultando os detalhes de implementação do AG. Esses sistemas são dirigidos aos executivos. Alguns desses sistemas suportam um conjunto de aplicações, como escalonamento, telecomunicações, etc. (por exemplo, PC/BEAGLE); outros enfocam um domínio específico de aplicação, como finanças (por exemplo OMEGA).
- **Sistemas orientados a algoritmos** suportam AG específicos. Esses sistemas são subdivididos em:
 - **sistemas de algoritmo específico** - contém um único AG, como, por exemplo, GENESIS.
 - **bibliotecas de algoritmos** - contém várias versões de AG e operadores agrupados em bibliotecas, como, por exemplo, OOGA.

- **Conjunto de ferramentas** são sistemas com muitas utilidades. Os algoritmos e operadores genéticos podem ser usados em uma grande variedade de aplicações e são subdivididos em:

- **sistemas tutores** - para auxiliar novatos a compreenderem os conceitos de AG. Tais sistemas oferecem um pequeno conjunto de opções para configurar o AG.
- **sistemas de propósito geral** - oferecem um conjunto de ferramentas para programar as aplicações de AG. Esses sistemas podem também ajudar o usuário a personalizar o software, como o Splicer.

A tabela 2 lista alguns dos ambientes de programação com AG, classificados de acordo com a taxonomia apresentada.

Tabela 2 - Ambientes de Programação com AG e suas Categorias

Orientados a Aplicações	Orientados a Algoritmos		Conjunto de Ferramentas	
	algoritmo específico	biblioteca de algoritmos	tutores	propósito geral
EVOLVER	ESC ^A P _A DE	<i>EM</i>	GA Workbench	EnGENEer
OMEGA	GAGA			GAME
PC/BEAGLE	GAUCSD			<i>MicroGA</i>
XpertRule	GENESIS	OOGA		PeGAsuS
GenAsys	GENITOR			Splicer

Detalhes sobre cada um desses ambientes podem ser vistos em Ribeiro Filho; Treleaving (1994). Entretanto, nenhum desses ambientes enfoca o uso de AG como apoio à simulação de sistemas.

3.8 Simulação Apoiada por Algoritmo Genético

Como foi visto no capítulo anterior, em um processo de simulação utilizando métodos tradicionais de modelagem e análise dos dados de entrada e saída, deve-se saber qual o tipo de distribuição estocástica dos dados de entrada, a fim de saber qual é a análise estatística mais adequada dos dados de saída. Essas tarefas não são triviais, sendo que o sucesso na escolha da análise dos dados de saída depende totalmente da escolha do modelo dos dados de entrada. Essas escolhas ficam a cargo do projetista, que deve conhecer a fundo o comportamento do sistema modelado, a fim de conseguir entender os resultados da simulação através da escolha correta do método analítico dos dados de saída. Essa é a tarefa mais difícil do processo de simulação.

Utilizando AG para apoiar a simulação têm-se grandes vantagens em relação à simulação tradicional. A primeira vantagem é que o projetista fica livre da tarefa de analisar o comportamento dos dados de entrada do sistema, para conhecer a sua distribuição estocástica. O AG gera um conjunto inicial de valores, chamado de população inicial, de maneira aleatória. Essa população é então utilizada para dar início ao processo evolutivo, sem necessidade de um maior conhecimento do comportamento do sistema.

Outra vantagem, talvez a principal, é que os resultados gerados são avaliados automaticamente pelo próprio AG, que verifica a qualidade de cada indivíduo como solução para o problema, independentemente de qualquer análise estatística. Assim o

projetista não precisa entender os dados de saída, analisando-os estatisticamente. O AG o libera de tal tarefa, a mais complicada do processo de simulação.

Também o processo de realimentação dos dados de entrada, para a simulação prosseguir até que se consiga uma configuração ideal ou satisfatória para o sistema modelado, é feita pelo AG, em função da análise da qualidade das soluções geradas por cada indivíduo, quando executado o modelo.

Esquemáticamente, a simulação apoiada pelo AG funciona conforme a representação da figura 10. Aqui, o AG dá início ao processo de evolução da solução, envia dados de entrada para o simulador, que executa o modelo, e gera dados de saída que serão analisado pelo AG. Continuando, a partir da avaliação feita e da aplicação dos operadores genéticos, o AG gera novos dados de entrada para o simulador. Esse processo se repete até que uma solução satisfatória seja encontrada pelo AG.

Figura 10 - Esquema de um Simulador Apoiado por Algoritmo Genético

Nesse estudo, como será visto no próximo capítulo, a simulação é baseada em modelos representados por grafos. Entretanto, o comportamento do AG seria o mesmo com outras estruturas, pois, para aplicar a técnica de AG, é necessário apenas

ter uma forma de representação das soluções, ou seja, um alfabeto definido para representação dos dados de entrada, e uma forma de avaliação da qualidade (aptidão ou desempenho) das soluções apresentadas, ou seja, conhecer a função objetivo.

Para a aplicação de AG no apoio à simulação, é interessante ter uma ferramenta computacional, que forneça ao projetista de sistemas complexos um ambiente adequado para desenvolvimento do modelo do sistema e a simulação desse modelo.

3.9 Considerações Finais

Como visto anteriormente, uma das áreas da Ciência da Computação que tem grande interesse na resolução de problemas complexos através de simulação é a Inteligência Artificial, com a utilização da técnica de Algoritmo Genético (AG).

AG trabalha na base da tentativa e erro que, utilizando análise de desempenho e operadores genéticos para gerar soluções, usa o conceito de retroalimentação do erro, aprendendo a produzir soluções cada vez melhores. Sendo assim, o AG libera o projetista da tarefa de análise dos dados de entrada e dos dados de saída da simulação, que corresponde à tarefa mais difícil, quando são aplicadas técnicas convencionais de análise de dados de simulação.

Apesar do AG nem sempre encontrar a solução ótima para o problema, na maioria das vezes ele encontra uma solução quase ótima, o que é aceitável quando se considera um problema complexo. Com AG, a cada restrição do problema, apenas o

cálculo do desempenho de cada indivíduo é modificado, sem que a implementação da técnica se torne mais complexa.

Utilizando a técnica de AG pode-se caminhar para a solução ótima, sem necessidade de análise mais detalhada dos dados de simulação. O AG faz a realimentação dos dados de entrada a partir da análise de desempenho com os dados de saída, sem precisar de maiores conhecimentos sobre o domínio do problema.

Portanto, pode-se verificar a generalidade e a aplicabilidade da técnica de AG para modelar sistemas complexos, e ainda obter uma solução ótima ou quase ótima para o problema, o que nem sempre é possível com métodos exatos convencionais.

Para a aplicação de AG no apoio à simulação, será apresentada uma ferramenta computacional, que fornece ao projetista de sistemas complexos um ambiente adequado para desenvolvimento do modelo do sistema e a simulação desse modelo.

Capítulo 4

SimAG - Ferramenta para Simulação de Sistemas Baseada em AG

A utilização de técnicas de apoio à simulação é, na maioria das vezes, muito dependente de ferramentas computacionais adequadas, que facilitem o tratamento dos detalhes da aplicação e do processo de simulação. O uso prático de AG para apoiar a simulação não foge dessas considerações. Nesse sentido, há a necessidade de um ambiente computacional que suporte a modelagem do sistema a ser simulado, a geração de dados para simulação, a simulação do sistema em si, a aplicação da função objetivo (F.O.), e a aplicação dos operadores genéticos. Como destaca Pidd (1994), um software para simulação deve suportar:

1. análise estocástica dos dados de entrada;
2. modelagem gráfica;

3. programação do modelo de simulação;
4. análise estatística das saídas.

O AG, no contexto de uma ferramenta computacional, seria utilizado nas tarefas 1 e 4, que, como visto anteriormente, são as tarefas mais difíceis do processo de simulação.

Algumas vantagens de se usar ferramentas baseadas em AG são: obtenção de resultados melhores, em menos tempo; possibilidade de uma representação interativa dos resultados; possibilidade de automatizar os aspectos mais técnicos do AG; e auxílio na estruturação do processo para a obtenção dos resultados desejados.

Um certo número de ferramentas tem sido desenvolvido, utilizando a técnica de AG, como mostra Ribeiro Filho (1994). Entretanto, o uso de AG como suporte à simulação de sistemas complexos ainda não foi devidamente explorado. Nesse contexto, vê-se a necessidade de aprimoramento de ferramentas e tratamento de aspectos específicos para apoio ao processo de simulação.

Nesse capítulo é descrito, na seção 4.1, o desenvolvimento da ferramenta proposta, juntamente com seus módulos. A seção 4.1.1 apresenta o módulo de geração. Na seção 4.1.2, discute-se o módulo de execução, os operadores genéticos implementados, a política de penalização adotada, e o processo de simulação apoiado por AG implementado. Nas seções 4.1.3 e 4.1.4 são descritos o módulo de visualização da evolução e o módulo de estatísticas. Por fim, na seção 4.2 são discutidas e reafirmadas as justificativas para o desenvolvimento da ferramenta e sua utilização.

4.1 Desenvolvimento da Ferramenta

Com o objetivo de explorar a idéia de apoio ao processo de simulação, através do uso de AG, foi desenvolvida uma versão protótipo de uma ferramenta, baseada em AG, para apoio à simulação de sistemas complexos, apresentada por Silva et al. (1996).

Essa ferramenta foi desenvolvida, utilizando plataforma SUN com sistema operacional UNIX, e os recursos oferecidos pelos softwares X-Windows, X-View, e DevGuide. Interfaces amigáveis são oferecidas, através de janelas e caixas de diálogo, facilitando e auxiliando o usuário na manipulação da ferramenta, simplificando assim seu uso. Para implementação da ferramenta, optou-se pela utilização da linguagem C padrão, com o objetivo de dar a maior portabilidade possível à ferramenta.

A estrutura da ferramenta é composta por quatro módulos principais: geração do sistema, execução da simulação apoiada pelo AG, visualização da evolução, e estatísticas da evolução. O diagrama de blocos, apresentado na figura 11, mostra a estruturação da ferramenta. A figura está associada ao menu principal da ferramenta, ou seja, os módulos citados são as opções oferecidas pelo menu principal.

O menu principal pode ser visto na figura 12. A seguir, são apresentadas: a descrição das operações que cada módulo executa; a forma de implementação e desenvolvimento desses módulos; e as interações com o usuário.

Figura 11 - Estrutura da Ferramenta para Simulação de Sistemas Baseada em AG

Figura 12 - Tela do Menu Principal

4.1.1 Módulo de Geração

O Módulo de Geração é responsável pela descrição do sistema modelado sob a forma de um grafo dirigido, conforme ilustra, de maneira genérica, a figura 13.

Figura 13 - Exemplo de um Grafo Modelado pela Ferramenta.

No contexto de AG, o conjunto de valores iniciais (valores das entradas E_i), que alimenta esse grafo, representa um indivíduo. Após a execução do grafo, o conjunto de valores resultantes (valores das saídas S_j) será analisado, aplicando a função objetivo, para avaliar seu desempenho como solução para o problema. A partir dessa avaliação, uma nova população é gerada, aplicando os operadores genéticos básicos.

Para geração do sistema, o módulo oferece, através de um editor gráfico, opções de edição e recuperação de um grafo anteriormente editado, grafo esse que reflete a estrutura do sistema em questão. A figura 14 mostra o menu de opções do editor.

Na edição, o usuário determina as coordenadas de cada nó, as funções associadas a cada arco de saída, e o destino desses arcos, conforme visto na figura 15.

Figura 14 - Menu de Opções do Editor Gráfico

Após a edição textual de todos os nós componentes do grafo, o usuário deve identificar os nós com entradas diretas, conforme ilustra a figura 16. A estrutura de cada nó componente do grafo é composta pelos elementos vistos na figura 15 e 16, ou seja, identificação, posição, identificação da origem dos arcos de entrada 1 e 2, função 1 e 2, identificação dos destinos dos arcos de saída 1 e 2. O grafo como um todo é armazenado em uma estrutura de lista, onde cada elemento da lista corresponde a um nó. Feito isso, o grafo é gerado na tela, como pode ser visto na figura 17.

Editar Grafo

Identificação do nó: 1

Posição: Linha: 1 Coluna: 1

Função 1 ($ax + by$): a: b: Destino: 0

Função 2 ($ax + by$): a: b: Destino: 0

Próximo nó

Cancelar

OK

AVISO: Destino 0, representa saída direta!

Figura 15 - Tela de Edição de um Grafo

Entradas

Nós com entradas de E1: 1

Próxima Entrada

Próximo Nó

Cancelar

OK

Figura 16 - Edição das Entradas Diretas do Grafo

Figura 17 - Grafo Gerado pela Ferramenta

Nesse ponto, o usuário deve editar a função objetivo através da tela de edição oferecida, conforme mostra a figura 18. A estrutura dessa função é interpretada pelo programa, e o usuário apenas indica os valores dos coeficientes. Além disso, deve indicar qual é o objetivo: maximizar, minimizar ou ignorar.

Função Objetivo

$FO = aS1 + bS2 + cS3 + dS4$

Valor de a: _____

Tipo Saída:

Figura 18 - Edição da Função Objetivo

A função objetivo é armazenada também em uma estrutura de lista, o que facilita seu cálculo quando da execução do AG e simulador.

É muito comum, considerando sistemas complexos, que algumas saídas, ou todas, tenham restrições de valores. Para isso, o usuário deve informar quais saídas têm restrições e quais seus valores de referência, como ilustra a figura 19.

Restrições

Restrição de S1: _____

Tipo Restrição:

Figura 19 - Estabelecimento das Restrições da Entradas de um Grafo

Esses valores são utilizados para verificação da necessidade de aplicação de penalização sobre o indivíduo que violou as restrições, corrigindo o valor de seu desempenho, proporcionalmente ao valor de violação das restrições, conforme será discutido no item seguinte, referente ao módulo de execução.

Todos esses dados sobre o grafo, a função objetivo, bem como as restrições, podem ser armazenados em um arquivo, de extensão “.gr”, para posterior recuperação e execução.

Na operação de recuperação o usuário deve fornecer o nome do arquivo a ser recuperado, como indica a figura 20. Esse grafo, então, é visto na tela do editor.

No módulo de geração existe também a possibilidade de visualizar a função objetivo e as restrições estabelecidas, como mostra a figura 21. Nesse módulo, são realizados alguns testes de consistência e completeza, para evitar problemas no grafo editado ou na função objetivo. Por exemplo, é feita a verificação da existência de pelo menos um arco de entrada e um arco de saída em cada nó. Também é verificado se existe pelo menos um arco dirigido para o vetor de saídas. O usuário é impedido de editar os arcos vindos do vetor de entrada e destinados aos nós do grafo. Esses testes auxiliam o projetista na tarefa de especificação do sistema.

Figura 20 - Definição do Nome do Arquivo do Sistema a ser Recuperado

The screenshot shows a window titled "Função Objetivo e Restrições". Inside, the objective function is displayed as $FO = 1.00 S1 + 1.00 S2 + 1.00 S3 + 1.00 S4$. Below this, the constraints are listed as $S1: \leq 1000.00$, $S2: \leq 1000.00$, $S3: \leq 1000.00$, and $S4: \leq 1000.00$. An "OK" button is located in the bottom right corner of the window.

Figura 21 - Visualização da Função Objetivo e Restrições

4.1.2 Módulo de Execução

No Módulo de Execução deve-se definir o tamanho da população e o número de gerações antes da execução do AG, como mostra a figura 22.

Dependendo do problema em questão, se o projetista tiver algum conhecimento específico do domínio do problema, ele poderá ter a possibilidade de determinar uma faixa de valores, dentro da qual os valores do vetor de entrada do grafo do sistema modelado estarão variando, como mostra a figura 23.

Figura 22 - Determinação do Tamanho da População e Número de Gerações

Figura 23 - Limitação dos Valores do Vetor de Entradas

Quando a opção de executar o AG é feita, o usuário deve definir o nome do arquivo de resultados, com extensão “.res”, que será usado para visualização do processo evolutivo e análise estatística dos resultados, como ilustra a figura 24. Durante a execução do AG e da simulação, o usuário é informado que o processo evolutivo está em andamento, conforme mostra a figura 25.

Figura 24 - Definição do Nome do Arquivo de Resultados

Figura 25 - Tela de Execução do AG e Simulador

4.1.2.1 Operadores Genéticos Implementados

Na implementação do AG, os operadores genéticos foram implementados considerando alguns fatores, que são descritos a seguir.

O processo de reprodução implementado considera o conceito de **reprodução elitista**, ou seja, alguns indivíduos da população passam inalterados para a geração seguinte, garantindo a monotonicidade do valor do desempenho do melhor indivíduo

no decorrer do processo evolutivo. Na implementação, após a realização de um conjunto de testes para determinar a taxa de elitização mais apropriada, foi estabelecida uma taxa de elitização igual a 1%, ou seja, os indivíduos da população, escolhidos na taxa de um a cada cem, apresentando os melhores desempenhos, passam inalterados para a próxima geração. Essa taxa foi estabelecida considerando a necessidade de controlar e manter a diversidade genética da população por mais tempo, no sentido de impedir a convergência prematura da solução para uma solução sub-ótima.

O operador seleção implementado segue a filosofia da **seleção pela regra da roleta**, descrita no capítulo anterior, o que garante que todos os indivíduos, mesmo aqueles que não apresentam um bom desempenho, têm chances de gerar descendentes, sendo que essas chances são proporcionais ao desempenho de cada um.

Esse tipo de operador segue o princípio de que um indivíduo com um desempenho ruim pode ser um bom pai, ou seja, gerar filhos com bom desempenho, e sua combinação genética pode ser fundamental para chegar a uma solução satisfatória para o problema em questão. Considerando essa possibilidade, tal indivíduo deve ter garantida sua chance de gerar descendentes.

Na implementação do operador cruzamento foi considerada uma **taxa de cruzamento variável**. Entretanto, o conceito de taxa de cruzamento variável utilizado aqui não corresponde exatamente ao conceito estabelecido, descrito no capítulo anterior. O conceito descrito anteriormente define que deve ser estabelecido, previamente, um valor inicial de taxa de cruzamento e um índice de variação desse valor inicial, ou seja, de quantas em quantas gerações essa taxa varia, e qual o valor a ser decrementado do valor atual. No entanto, o princípio que rege sua aplicação é o mesmo, ou seja, à medida que a população se torna homogênea, no decorrer do

processo evolutivo, o operador cruzamento perde sua força e importância, pois o cruzamento de indivíduos iguais não gera novos descendentes. Por esse motivo, ele deve ser aplicado um número menor de vezes. Na ferramenta desenvolvida neste trabalho, a cada par de indivíduos a serem cruzados, é feita uma análise para verificar se esses indivíduos são idênticos. Se tais indivíduos são idênticos, eles simplesmente são copiados para a geração seguinte, levando consigo seu valor de desempenho. Portanto, o valor da taxa de cruzamento varia a cada execução do AG, dependendo de quão homogênea se torna a população. A definição desse operador foi uma das consequências deste trabalho de pesquisa e foi chamado de “**cruzamento analítico**”.

O objetivo, considerando uma taxa de cruzamento variável, é agilizar o processo evolutivo, evitando que sejam feitos cruzamentos desnecessários, e com isso tornando o AG mais rápido. Essa é uma característica de qualquer operador cruzamento que utiliza uma taxa variável. Nesta implementação, entretanto, o indivíduo que passa inalterado para a próxima população leva consigo o seu valor de desempenho.

Considerando os sistemas modelados com grafos dirigidos, a simulação pode ser entendida como sendo uma parte do processo de cálculo do valor da função objetivo do indivíduo, na aplicação do AG. O cálculo da função objetivo implica percorrer o grafo, verificar suas saídas, e aplicar sobre elas a função objetivo. Além disso, deve ser feita a verificação da violação de restrições e, se for o caso, aplicar penalizações. Todo esse processo leva um tempo de processamento significativo. À medida que um indivíduo herda de seus pais o valor do seu desempenho, todo esse processo é evitado, o que agiliza muito a execução do AG. A figura 26 mostra como é esse processo de aplicação do cruzamento analítico, considerando todo o contexto de simulação de sistemas apoiada por AG proposto.

O ganho de tempo, com esse operador cruzamento analítico, com taxa de aplicação variável de acordo com testes realizados é, em média, 25% do tempo de processamento do módulo de execução.

Na implementação do operador mutação, foi definida uma **taxa de mutação variável exponencial**, que se comporta como uma função exponencial. Tal função foi estabelecida de seguinte maneira: inicialmente, o valor da taxa de mutação é fixado em 0,001, ou seja, a cada mil genes, um gene sofre mutação, como sugere Goldberg (1989); quando o AG processa metade do número de gerações determinado, essa taxa aumenta para 0,002, ou seja, a cada quinhentos genes, um gene sofre mutação; quando o AG processa três quartos do número de gerações determinado, a taxa de mutação passa a valer 0,004, ou seja, um a cada duzentos e cinquenta genes sofre mutação; por fim, quando o AG processa sete oitavos do número de gerações determinado, essa taxa passa a valer 0,008, ou seja, um a cada cento e vinte e cinco genes sofre mutação. A figura 27 ilustra a variação da taxa de mutação exponencial no decorrer do processo evolutivo.

Esses valores de taxa de mutação foram determinados através de testes, considerando que o operador mutação tem sua importância ressaltada, à medida que a população se torna homogênea, garantindo a exploração de todo o espaço de busca e evitando soluções sub-ótimas. Entretanto, deve-se considerar que o valor estipulado não pode ser muito alto, pois isso tornaria a busca muito aleatória, desconsiderando o direcionamento já determinado pelo AG até então.

Figura 26 - O Processo de Simulação Apoiada por AG Proposto

Figura 27 - Variação da Taxa de Mutação Exponencial

4.1.2.2 Política de Penalização

Considerando sistemas complexos, é muito comum haver um conjunto de restrições relacionado com os resultados, ou seja, um indivíduo deve ter um desempenho dentro de um certo conjunto de valores, como discutido no capítulo anterior. Qualquer outro valor fora desse conjunto torna o indivíduo inválido como solução para o problema.

Para resolver a questão com os indivíduos inválidos, deve-se definir uma política de penalização, baseada na determinação numérica de quanto um indivíduo é ruim. Assim, o desempenho do indivíduo é definido, considerando esse valor.

Usando as idéias de Richardson et al. (1989), foi utilizada neste trabalho uma política de penalização baseada no quanto um indivíduo violou cada uma das restrições existentes, e não em quantas restrições foram violadas. Essa penalização é aplicada após o cálculo do desempenho do indivíduo. A política de penalização adotada é expressa no algoritmo descrito na figura 26.

O objetivo da aplicação da penalização é corrigir as distorções no valor do desempenho do indivíduo e viabiliza que um indivíduo, mesmo inválido com um desempenho penalizado, possa gerar bons filhos, podendo levar à uma boa solução para o problema. Esse indivíduo, então, ganha uma chance de gerar descendentes, mesmo que ele seja uma solução inválida.

Entretanto, deve-se tomar o cuidado de definir uma política de penalização robusta o suficiente, de tal maneira que um indivíduo inválido, que já teve seu valor de desempenho corrigido, não tenha um desempenho muito alto, que possa, em alguma etapa do processo evolutivo, ser o melhor indivíduo, ou um dos melhores indivíduos, da população. Isso faria com que tal indivíduo, através da reprodução elitista adotada, participasse da elite e fosse sendo passado para as gerações seguintes. Tal situação poderia levar o AG a encontrar uma solução não apenas ruim, mas também inviável.

Durante os estudos para a determinação da melhor política de penalização foram testadas algumas possibilidades. A primeira delas foi baseada na idéia de penalizar o indivíduo de acordo com o valor médio de suas porcentagens de violação. Por exemplo, se o problema considerado apresentava restrições relativas a três saídas, verificava-se qual a porcentagem de violação da primeira restrição (v_1), qual a porcentagem de violação da segunda restrição (v_2) e qual a porcentagem de violação da terceira restrição (v_3). Com esses valores, tirava-se a média de violação

(v_m). Esse valor era descontado do desempenho do indivíduo em questão. Entretanto, essa política não se mostrou robusta o suficiente para evitar que um indivíduo inválido dominasse a população.

Uma segunda política de penalização foi definida, considerando o valor integral de cada violação. Para cada restrição violada, a porcentagem de violação calculada era descontada do desempenho do indivíduo em questão. Essa política, assim como a anterior, também não foi suficientemente forte.

Figura 28 - Política de Penalização Adotada

Por fim, a terceira política de penalização adotada aplica a penalização por faixas de valores, entre 0% e 25%, entre 25% e 50%, e acima de 50%. Essa penalização, como pode ser visto no algoritmo da figura 28, é aplicada a cada violação de restrição. Um conjunto de testes mostrou que essas três faixas de valores eram suficientes para dar a característica de robustez necessária à política de penalização.

Nesse conjunto de testes, o exemplo de sistema utilizado foi um modelo simplificado de uma refinaria de petróleo, descrito no próximo capítulo. Foram executadas, para cada uma das três políticas de penalização descritas anteriormente, dez simulações considerando uma população de dez indivíduos e cinquenta gerações, dez simulações considerando uma população de cinquenta indivíduos e cinquenta gerações, dez simulações considerando uma população de cem indivíduos e cem gerações, e dez simulações considerando uma população de duzentos indivíduos e quinhentas gerações.

Para a primeira política de penalização adotada, observou-se que aproximadamente 45% das simulações apresentou um indivíduo inválido como solução. Nos experimentos com a segunda política de penalização, observou-se que 13% das simulações teve como resultado um indivíduo inválido. Já com a terceira política adotada, não foi observada nenhuma solução inválida após a realização dos testes.

4.1.2.3 O Processo de Simulação

Para o cálculo do desempenho de um indivíduo, é necessário percorrer o grafo do sistema modelado, ou seja, simular o sistema em questão. Nesse ponto, percebe-se a conexão entre o processo de aplicação do AG e o processo de simulação, como ilustrado na figura 10, que mostra, esquematicamente, um simulador apoiado por AG.

Pode-se entender que, no processo interativo entre AG e simulador, do ponto de vista do AG, a simulação acontece como parte do processo de cálculo do valor do desempenho do indivíduo.

Neste trabalho, optou-se pelo desenvolvimento de um simulador por questões de praticidade e rapidez. A técnica de AG poderia ser aplicada a qualquer simulador disponível comercialmente. Entretanto, a dificuldade de recursos financeiros para aquisição de um software de simulação foi um fator determinante na estratégia adotada.

Foi desenvolvido, então, um protótipo, tendo como objetivo mostrar a viabilidade do uso da técnica de AG para apoiar a tarefa de simulação de sistemas, especialmente de sistemas complexos.

O sistema é modelado como um grafo dirigido, como ilustrado na figura 13. Tendo em vista que essa é uma versão protótipo da ferramenta, algumas restrições foram impostas. Os nós do grafo têm dois arcos de entrada, e um ou dois arcos de saída. A estrutura do nó do grafo é vista, esquematicamente, na figura 29.

Os campos “e1” e “e2” recebem os valores calculados nos nós que têm arcos com destino nesse nó em questão, ou os valores do vetor de entrada. O campo

“identificação do nó” indica qual o nó em questão. Os campos “função f1” e “função f2” contêm a descrição das respectivas funções, armazenadas em estrutura de lista. A função f2 pode não existir.

Figura 29 - Estrutura de um Nó do Grafo

Essas funções, nessa versão protótipo da ferramenta, são funções lineares, no formato $ax+by$, visto na figura 15, onde x corresponde ao valor do arco de entrada “e1”, e y corresponde ao valor do arco de entrada “e2”. Em versões futuras, as funções poderão ser definidas de acordo com as necessidades da especificação do sistema.

Os campos “valor saída s1” e “valor saída s2” recebem os valores resultantes do cálculo das funções.

Por fim, os campos “destino 1” e “destino 2” identificam os nós, ou posições do vetor de saída, que irão receber os valores de saída calculados.

Os nós do grafo são armazenados em um arquivo de extensão “.gr”, criado no módulo de edição. O arquivo está ordenado por ordem de edição, ou seja, o primeiro nó editado é o último registro do arquivo. Quando esse arquivo é recuperado para execução, esses nós são colocados em uma lista encadeada, denominada “gr”. Cada nó recuperado é inserido no fim da lista. Assim, a lista reflete a ordem dos nós estabelecida na edição.

Cada elemento da lista “gr” é cabeça de uma outra lista, onde estão contidos os nós destinos dos arcos de saída desse elemento, ou seja, seus nós adjacentes. Essa lista é chamada “adja”. Esquematicamente, essa estrutura de grafo pode ser entendida como ilustra a figura 30, considerando o grafo da figura 13.

Figura 30 - Estrutura do Grafo Manipulada pela Ferramenta

O vetor de entradas e o vetor de saídas são, cada um, uma estrutura de lista encadeada que, considerando o grafo exemplo, podem ser representados como

mostram as figuras 31 e 32. Nelas, o campo “valor” corresponde ao valor do gene em questão, no caso do vetor de entradas, ou ao valor resultante, no caso do vetor de saídas.

No vetor de entradas, o campo “dest” é cabeça de uma lista de nós, ou posições no vetor de saídas, destino dos arcos oriundos de cada posição desse vetor. As posições no vetor de saída são determinadas automaticamente, na ordem de edição dos nós.

Para percorrer o grafo, inicialmente, o vetor de entradas é preenchido com os valores dos genes do indivíduo em questão. Após o preenchimento dessa lista de valores de entrada, os nós do grafo com entrada direta têm os campos “e1” e “e2” preenchidos. Em seguida, o grafo começa a ser percorrido. A lógica desse processo é descrita no algoritmo da figura 33.

Figura 31 - Estrutura do Vetor de Entradas

Ao final da execução desse procedimento, ou seja, quando a lista de execução estiver vazia, o vetor de saídas já estará preenchido. Assim, o cálculo do desempenho do indivíduo em questão já pode ser feito.

Figura 32 - Estrutura do Vetor de Saída

4.1.3 Módulo de Visualização da Evolução

O Módulo de Visualização da Evolução foi elaborado para que o usuário tenha uma noção do processo evolutivo. Para isso, o módulo mostra a variação dos pontos soluções de cada saída, ou seja, de cada posição do vetor de saídas. Nesse caso particular, o módulo está mostrando soluções planares pois, como citado anteriormente, essa versão protótipo da ferramenta suporta apenas funções lineares, no formato $ax+by$.

Eventualmente, o usuário (quando de interesse) pode fornecer o nome do arquivo de resultados, gerado no módulo de execução, outro que não o corrente. Entretanto, quando esse módulo é chamado, após a execução de um grafo, os resultados dessa última execução aparecem na tela. Se o usuário estiver interessado em visualizar outros resultados, a opção de recuperar outro arquivo pode ser feita, e o

usuário deve informar o nome do arquivo de resultados em questão, como mostra a figura 34.

Início

Verifique todos os nós do grafo que estão prontos para ser executados, ou seja, que têm suas entradas “e1” e “e2” preenchidas. Esses nós são colocados em uma lista de execução.

Enquanto a lista de execução não estiver vazia ***faça***:

Início

Considere o nó cabeça da lista de execução.

Calcule o valor da função f_1 e coloque em “s1”.

Se existir, calcule o valor da função f_2 e coloque em “s2”.

Encontre o nó, ou posição do vetor de saída, que deve receber o valor “s1”, percorrendo a lista de adjacentes.

Encontre o nó, ou posição do vetor de saída, que deve receber o valor “s2”, se existir esse valor, percorrendo a lista de adjacentes.

Percorra a lista de adjacentes de cada nó, para encontrar os nós que estão prontos para execução e insira tais nós no fim da lista de execução.

Retire o nó cabeça da lista de execução.

Fim***Fim***

Figura 33 - Algoritmo de Percorrimento do Grafo

Figura 34 - Escolha do Arquivo de Resultados a Ser Analisado

Para cada posição do vetor de saídas, que corresponde ao que é processado pelo sistema modelado, é gerado um gráfico, correspondente à função relacionada com o cálculo desse valor.

A partir da geração desse gráfico, o usuário pode visualizar a evolução desse valor, correspondente a uma posição do vetor de saída, ou seja, um dos produtos gerados pelo sistema. Esses valores são mostrados como pontos plotados sobre o gráfico e unidos por uma linha, indicando qual a direção da evolução. O último valor, ou seja, a solução encontrada pelo AG, é indicado por uma circunferência, conforme ilustra a figura 35.

Os valores vistos no gráfico correspondem aos valores relacionados com o melhor indivíduo em cada geração. Nesse contexto, como um indivíduo pode permanecer como o melhor indivíduo por várias gerações, até que outro indivíduo de melhor desempenho seja encontrado, apenas alguns pontos são plotados no gráfico.

Figura 35 - Visualização da Evolução dos Valores Encontrados pelo AG

Nesse módulo o usuário pode ver cada uma das saídas geradas, podendo também escolher a ordem de visualização desses resultados. Cabe ressaltar aqui que o gráfico é dimensionado de acordo com os valores encontrados pelo AG, podendo não corresponder ao valor ótimo possível para o problema, pois o AG não garante a otimalidade de resultados.

Nas versões futuras, a evolução das saídas poderá ser visualizada sobre curvas tridimensionais, uma vez que poderão ser usadas funções mais complexas de segundo grau.

4.1.4 Módulo de Estatísticas

O Módulo de Estatísticas fornece ao usuário uma análise estatística dos resultados obtidos com a aplicação do AG para resolução do sistema modelado.

Da mesma forma que o módulo de visualização da evolução, inicialmente, o usuário deve fornecer o nome do arquivo de resultados, gerado no módulo de execução. Quando esse módulo é chamado, após a execução de um grafo, as estatísticas dessa última execução aparecem na tela. Para a visualização de outros resultados armazenados será necessário fornecer o nome de outro arquivo, através da janela vista na figura 36.

Figura 36 - Recuperação do Arquivo de Resultados para Análise Estatística

A análise estatística realizada nesse módulo consiste, basicamente, na geração de um histograma, onde pode ser vista a evolução do valor da solução representado pelo melhor indivíduo em cada população. Além disso, é mostrada também a evolução do valor médio da qualidade da solução da população como um todo. Esses dois resultados são colocados no mesmo histograma, com o objetivo de permitir que o usuário verifique o comportamento do AG e como foi o processo de evolução da qualidade da solução. Um exemplo, genérico, dessa análise estatística pode ser visto na figura 37.

As curvas apresentadas no gráfico ilustram o comportamento típico de um processo de evolução, quando aplicado o AG. Como indicado, cada barra escura representa o desempenho do melhor indivíduo na geração correspondente, e cada barra clara representa o desempenho médio da população na correspondente geração. Inicialmente, o desempenho médio da população é baixo, quando comparado ao desempenho do melhor indivíduo, devido à diversidade genética da população, ou seja, à heterogeneidade dos indivíduos. À medida que a evolução ocorre, a média do valor de desempenho da população tende a se igualar ao valor do desempenho do melhor indivíduo, pois os piores indivíduos vão sendo descartados, e os melhores indivíduos vão sendo preservados, gerando mais descendentes. Esse procedimento torna a população mais homogênea. Mesmo que outro indivíduo, com desempenho superior ao melhor indivíduo da geração anterior, seja encontrado, essa tendência persiste, como pode ser observado no gráfico da figura 37.

Nesse módulo, o dimensionamento do gráfico à tela, com relação ao número de gerações e ao valor de desempenhos, é feito automaticamente. Nesse módulo, também, podem ser verificados o número de mutações ocorridas e o número de simulações executadas. Esse registro se torna interessante, pois tanto a taxa de

mutação quanto a taxa de cruzamento são variáveis, sendo que, a cada execução do grafo, tais valores podem ser modificados, dependendo do número de gerações e do crescimento da homogeneidade entre os indivíduos da população. O tempo de processamento para encontrar a solução para o problema em questão também é registrado.

Figura 37 - Exemplo da Análise Estatística Feita pela Ferramenta

4.2 Justificativas para o Desenvolvimento da Ferramenta e sua Utilização

Considerando que a simulação de sistemas complexos se torna inviável sem um apoio computacional efetivo e sem um suporte de avaliação de resultados para selecionar os dados de entrada e analisar os dados de saída gerados no processo de simulação, foi desenvolvida uma ferramenta computacional para simulação de sistemas complexos apoiada por AG.

Tal ferramenta foi desenvolvida e apresentada nesse capítulo, como uma versão protótipo, com o objetivo de mostrar a viabilidade da aplicação da técnica de AG no apoio à tarefa de simulação de sistemas complexos.

Esse trabalho poderia ter sido desenvolvido utilizando um simulador comercial genérico. Entretanto, por razões já citadas anteriormente, optou-se pelo desenvolvimento de um simulador simples, que permitisse a realização de experimentos, visando testar e mostrar a potencialidade do AG como suporte efetivo na tarefa de simulação.

No desenvolvimento de tal ferramenta, foram elaboradas interfaces amigáveis, em um ambiente computacional avançado (estações SUN - Sparc 20, sistema operacional UNIX, X-Windows, X-View, etc.), e utilizadas estruturas de dados dinâmicas, visando, futuramente, uma expansão do escopo dessa ferramenta.

Nesse contexto, um exemplo de aplicação da ferramenta, através da modelagem e simulação de um sistema complexo, torna-se importante e interessante.

Será usado para isso um exemplo de um sistema complexo simplificado, com solução ótima conhecida através do método de programação linear. Dessa maneira, os resultados obtidos com a aplicação do método proposto poderão ser comparados com a solução ótima, gerando análises de desempenho e eficácia do processo.

Capítulo 5

Estudo de Caso: Um Exemplo de Aplicação da Técnica Proposta

Neste trabalho está sendo proposta a utilização de AG como técnica de geração e análise de dados no processo de simulação de sistemas complexos. Para isso, foi desenvolvida uma ferramenta computacional de apoio à utilização dessa técnica. Para ilustrar a aplicabilidade e potencialidade da técnica de simulação apoiada por AG proposta, será apresentado aqui um exemplo de aplicação da ferramenta, através da modelagem e simulação de um sistema complexo simplificado, com solução ótima conhecida através de um método matemático. A opção por um exemplo de sistema complexo simplificado se deve ao fato da necessidade de um exemplo com solução ótima conhecida, para efeito de comparação com a solução obtida através da aplicação da técnica proposta. Os resultados obtidos serão utilizados em um estudo analítico de desempenho e eficácia do processo.

Este capítulo está organizado da maneira descrita a seguir. Na seção 5.1 apresenta-se uma descrição do problema estudado. Na seção 5.2 é apresentada a resolução do problema utilizando um método matemático. A resolução do problema através de simulação apoiada por AG é vista na seção 5.3. Na seção 5.4 é feita uma análise comparativa de resultados computacionais, quanto à qualidade da solução e quanto ao tempo de execução, considerando várias implementações de AG. A seção 5.5 traz uma argumentação no sentido de comprovar a aplicabilidade da técnica de simulação apoiada por AG proposta.

5.1 Descrição do Problema

Será considerada aqui uma versão simplificada do problema do balanceamento e produção de uma refinaria de petróleo, visto em Edgar e Himmelblau (1988). A figura 38 mostra um esquema dos recursos necessários e produtos gerados pela refinaria, onde custos e preços de venda são dados entre parêntesis.

A tabela 3 lista as informações sobre as porcentagens de cada tipo de petróleo necessárias no processo de refino dos produtos. A tabela também lista as quantidades estabelecidas pelo mercado para cada produto em termos da produção diária máxima. Também é informado o custo do processamento.

Figura 38 - Esquema de Entradas e Saídas da Refinaria

Tabela 3 - Dados para Entradas e Produtos da Refinaria

	Volume de petróleo em porcentagem		Produção máxima permitida por dia (l)
	petróleo #1	petróleo #2	
Gasolina	80	44	24.000
Querosene	5	10	2.000
Óleo Combustível	10	36	6.000
Resíduo	5	10	
Custo de Processamento (\$/barril)	0,5	1	

Considerando o seguinte conjunto de variáveis:

- E1 = quantidade diária do petróleo #1
- E2 = quantidade diária do petróleo #2
- S1 = quantidade diária de gasolina
- S2 = quantidade diária de querosene
- S3 = quantidade diária de óleo combustível
- S4 = quantidade diária de resíduo
- S5 = custo de material
- S6 = custo de processamento

Deseja-se maximizar o lucro desse sistema. A Função Objetivo (F.O.) desse sistema, então, é a diferença entre o valor da produção e custos, ou seja:

$$F.O. = \text{lucro} = \max (\text{produção} - \text{custo de material} - \text{custo de processamento})$$

Todos os valores são diários, onde:

$$\text{produção} = 36 S1 + 24 S2 + 21 S3 + 10 S4 \quad (1)$$

$$\text{custo de material} = 24 E1 + 15 E2 \quad (2)$$

$$\text{custo de processamento} = 0,5 E1 + 1 E2 \quad (3)$$

Pela figura 38 e tabela 3 tem-se:

$$\text{gasolina} = 0,80 E1 + 0,44 E2 = S1 \quad (4)$$

$$\text{querosene} = 0,05 E1 + 0,10 E2 = S2 \quad (5)$$

$$\text{óleo combustível} = 0,10 E1 + 0,36 E2 = S3 \quad (6)$$

$$\text{resíduo} = 0,05 E1 + 0,10 E2 = S4 \quad (7)$$

$$\text{custo material} = 24,00 E1 + 15,00 E2 = S5 \quad (8)$$

$$\text{custo processamento} = 0,50 E1 + 1,00 E2 = S6 \quad (9)$$

Portanto, a Função Objetivo, fazendo as devidas substituições, pode ser expressa da seguinte maneira:

$$F.O. = \max(36S1 + 24S2 + 21S3 + 10S4 - S5 - S6) \quad (10)$$

Considerando o contexto de sistemas complexos, é comum existir um conjunto de restrições impostas ao problema, devido à natureza do problema. Nesse caso, as restrições impostas pelo mercado são:

$$\text{gasolina:} \quad S1 \leq 24.000 \Rightarrow 0,80 E1 + 0,44 E2 \leq 24.000 \quad (11)$$

$$\text{querosene:} \quad S2 \leq 2.000 \Rightarrow 0,05 E1 + 0,10 E2 \leq 2.000 \quad (12)$$

$$\text{óleo combustível:} \quad S3 \leq 6.000 \Rightarrow 0,10 E1 + 0,36 E2 \leq 6.000 \quad (13)$$

Um outro conjunto de restrições, embora não esteja explicitado na formulação do problema, diz respeito aos valores de E1 e E2, que devem ser valores não negativos:

$$\text{petróleo \#1:} \quad E1 \geq 0 \quad (14)$$

$$\text{petróleo \#2:} \quad E2 \geq 0 \quad (15)$$

Tais variáveis devem ser positivas, pois não tem sentido ter uma taxa de produção negativa.

5.2 Resolução do Problema Utilizando um Método Matemático

O problema proposto pode ser resolvido através do uso de Programação Linear, como apresentado por Edgar e Himmelblau (1988).

A dimensão do problema pode ser reduzida, eliminando-se as variáveis S1, S2, S3 e S4 através de restrições de igualdade, deixando apenas duas variáveis independentes E1 e E2. A função de produção pode ser reduzida da seguinte maneira:

$$\begin{aligned}
 \text{Produção total} &= (36) (0,80 E1 + 0,44 E2) \\
 &+ (24) (0,05 E1 + 0,10 E2) \\
 &+ (21) (0,10 E1 + 0,36 E2) \\
 &+ (10) (0,05 E1 + 0,10 E2) \\
 &= (32,6 E1 + 26,8 E2)/\text{dia}
 \end{aligned}$$

A Função Objetivo, após subtrair o custo de material e custo de processamento, é:

$$f(E) = \max(8,1E1 + 10,8E2) \quad (16)$$

No caso das restrições, se por alguma razão fossem permitidos valores negativos, deveria ser aplicado um procedimento de transformação linear para converter esses valores em novos valores não negativos. Isso é necessário para resolver o problema através do método de Programação Linear. Assim, a declaração

formal do problema de Programação Linear está completa, consistindo das Eqs (11) a (16).

Será feita uma ilustração gráfica do procedimento, consistindo de três passos:

- Traçar as restrições no plano E1-E2.
- Determinar a região de factibilidade para E1 e E2 (aqueles valores de E1 e E2 que satisfazem todas as restrições).
- Encontrar o ponto sobre a borda da região de factibilidade que maximiza a função objetivo, através do exame das interseções das restrições; a inclinação da função objetivo $f(E) = e$, onde e é um parâmetro que indica diferentes valores de f , pode indicar quais as interseções são mais favoráveis.

A figura 39 define a região de factibilidade para o problema exemplo através dos limites das restrições A, B e C. A área hachurada, que é a região factível que satisfaz todas as restrições, é convexa.

A figura 40 é uma ampliação da região factível, com sobreposição da F.O. sobre seus contornos. À medida que E1 e E2 se tornam maiores, a F.O. aumenta; entretanto, não é possível ter uma produção ilimitada, devido ao conjunto de restrições do problema. Nesse contexto, deve-se encontrar o nível máximo de produção, que também satisfaça as restrições. Começando com $E1=E2=0$, deslizando ao longo da restrição $E2=0$ (ao longo do eixo E1). Pode-se aumentar a F.O. até a interseção da restrição A com $E2=0$. Agora, percorrendo a restrição A, há um aumento de F.O. ao longo desse caminho.

Figura 39 - Delimitação da Região Factível para o Problema da Refinaria

Figura 40 - Ampliação da Região Factível com Parametrização da F.O.

Edgar e Himmelblau (1988) destacam que a análise da F.O., para ver quais as direções levam a um aumento do seu valor, é muito útil na resolução de problemas através de Programação Linear. A figura 40 identifica três linhas tracejadas, correspondendo a três níveis diferentes de crescimento desse valor. Em cada caso, a equação para o contorno do desempenho é $f=8,1E1+10,8E2$. Cada linha identifica os possíveis valores de E1 e E2 que dão esse desempenho, identificados pelos círculos numerados de 1 a 4. A visualização dos valores crescentes de desempenho é fácil a partir do gráfico. A figura 40 mostra que o valor ótimo da F.O. ocorre no vértice da região factível, com desempenho = f3.

Pode-se verificar que o valor ótimo, conforme indica o círculo número 1, ocorre aproximadamente em:

$$E1 \approx 26.000$$

$$E2 \approx 7.000$$

Os valores exatos para E1 e E2 são encontrados a partir da resolução das inequações (11) e (12) como equações:

$$0,80E1 + 0,44E2 = 24.000$$

$$0,05E1 + 0,10E2 = 2.000$$

O resultado desse cálculo é:

$$E1 = 26.200$$

$$E2 = 6.900$$

$$F.O. = 8.1E1 + 10.8E2 = \$286.700$$

Portanto, o valor ótimo para a F.O., ou seja, o desempenho ótimo do sistema é \$286.700. Esse valor será usado para comparação de resultados obtidos através do método de simulação apoiada por AG proposto.

Aqui, a utilização de Programação Linear foi possível devido à simplicidade do exemplo apresentado. Com um exemplo mais complexo, a Programação Linear se torna inviável, sendo necessária a aplicação de outros métodos de resolução de problemas.

5.3 Resolução do Problema Através de Simulação Apoiada por AG

Considerando a ferramenta proposta no capítulo anterior, em Silva; Kirner (1996) é apresentado o exemplo do sistema da refinaria, modelado da maneira indicada na figura 41.

Figura 41 - Modelo Gráfico da Refinaria de Petróleo Proposto pela Ferramenta

A figura 42 mostra a tela do editor gráfico da ferramenta com o modelo da refinaria desenhado. Na figura 43 pode ser vista a F.O. editada no ambiente da

ferramenta, bem como as restrições impostas ao problema em questão. Na figura 44 são apresentados resultados computacionais, com 50 indivíduos na população e 10 gerações.

Também foram determinados como valores limite de validade para as entradas E1 e E2 os intervalos $[0, 30.000]$ e $[0, 10.000]$, respectivamente.

Esses resultados foram obtidos em dois segundos, e a qualidade da solução encontrada pelo AG foi muito satisfatória, considerando o contexto de sistemas complexos. A solução encontrada pelo AG foi $E1=26.190,17$ e $E2=6.898,87$. O valor da F.O. considerando essas entradas foi \$286.648,09.

Figura 42 - Modelo Gráfico da Refinaria de Petróleo Utilizando a Ferramenta

Função Objetivo e Restrições		
FO = 36.00 S1 + 24.00 S2 + 21.00 S3 + 10.00 S4 - 1.00 S5 - 1.00 S6		
Restrições:	S1: <= 24000.00	S2: <= 2000.00
	S3: <= 6000.00	S4: Sem Restrição
	S5: Sem Restrição	S6: Sem Restrição
		OK

Figura 43 - Determinação da Função Objetivo e Restrições do Problema da Refinaria

Resultado
Entradas: 26190.17 6898.87
Saídas: 23987.64 1999.40 5102.61 1999.40 732047.12 19993.96
Desempenho: 286648.09
Tempo de execução : 2 segundos
OK

Figura 44 - Resultados Computacionais para o Problema da Refinaria

Na figura 45, que mostra a evolução do melhor indivíduo durante o processo evolutivo no que se refere à produção de querosene (S2).

Figura 45 - Evolução do Resultado Referente à Produção de Querosene

A figura 46 mostra a análise estatística da execução do SimAG. O melhor indivíduo da última geração, que apresentou um desempenho igual a \$286.648,09, teve qualidade igual a 99,98% do valor ótimo. O desempenho médio da última geração foi \$280.043,08, representando 97,68% do valor ótimo.

Figura 46 - Análise dos Resultados Computacionais Apresentados pela Ferramenta

5.4 Análise Comparativa de Resultados Computacionais

Para efeito de comparação de qualidade de resultados, foi feito um estudo comparativo dos resultados apresentados para o problema da refinaria descrito, considerando várias implementações de AG. Na tabela 4 são descritos os quatro tipos de implementação de AG e as legendas correspondentes a cada tipo, sendo que o método proposto utiliza cruzamento analítico e taxa de mutação exponencial. Além disso, a tabela apresenta as legendas referentes à variação no tamanho da população.

Tabela 4 - Legenda usada nos Gráficos de Análise de Resultados

AG com taxa de cruzamento igual a 1 e taxa de mutação igual a 0,001
AG com taxa de cruzamento igual a 1 e taxa de mutação exponencial
AG com cruzamento analítico e taxa de mutação igual a 0,001
AG com cruzamento analítico e taxa de mutação exponencial
Resultados considerando população de 10 indivíduos
Resultados considerando população de 50 indivíduos
Resultados considerando população de 100 indivíduos
Resultados considerando população de 200 indivíduos

A tabela 5 apresenta um resumo dos testes realizados, considerando o valor do desempenho, ou da F.O., e o tempo de processamento em cada execução. A partir desse quadro, foram feitos análises e gráficos, comparando os resultados por partes, considerando a qualidade da solução e o tempo de processamento. O objetivo dessas comparações é mostrar que a proposta do Simulador apoiado pelo AG proposto apresenta melhores resultados na maioria das vezes, tanto em relação à qualidade da solução quanto em relação ao tempo de execução.

Todos os testes feitos seguiram a configuração do valor dos genes E1 e E2 variando entre 0 e 10.000 e 0 e 30.000, respectivamente. Para cada configuração foram executados dez testes, sendo que foi considerado o valor mais próximo da média dos resultados.

Tabela 5 - Resultados da Simulação Apoiada por AG variando as Implementações dos Operadores Genéticos

			10 gerações	valor	\$278.004	\$276.462	\$270.764	\$278.636
		tempo	1s	1s	0s	0s
	50 gerações	valor	\$283.942	\$277.763	\$276.793	\$283.954
		tempo	2s	1s	1s	2s
	100 gerações	valor	\$278.147	\$277.383	\$275.280	\$285.349
		tempo	3s	3s	3s	3s
10 gerações	valor	\$283.003	\$280.218	\$279.296	\$284.360
		tempo	2s	2s	2s	2s
	50 gerações	valor	\$282.549	\$284.612	\$284.666	\$286.404
		tempo	8s	7s	9s	7s
	100 gerações	valor	\$281.996	\$282.962	\$280.841	\$286.175
		tempo	15s	15s	14s	14s
10 gerações	valor	\$284.852	\$284.457	\$284.358	\$285.499
		tempo	4s	4s	4s	4s
	50 gerações	valor	\$284.294	\$285.219	\$285.735	\$286.112
		tempo	26s	15s	15s	15s
	100 gerações	valor	\$285.393	\$284.196	\$284.556	\$286.354
		tempo	31s	30s	29s	29s
	500 gerações	valor	\$285.110	\$284.855	\$285.984	\$285.494
		tempo	146s	148s	134s	132s
	1000 gerações	valor	\$285.185	\$285.240	\$285.471	\$285.752
		tempo	528s	432s	260s	263s
500 gerações	valor	\$285.594	\$286.315	\$285.854	\$285.716
		tempo	584s	621s	338s	292s
	1000 gerações	valor	\$285.891	\$286.521	\$286.378	\$286.358
		tempo	1032s	1095s	577s	566s

Também pode-se observar que, à medida em que o tamanho da população e o número de gerações aumentavam, o desvio padrão das soluções encontradas em cada uma das baterias de testes realizadas diminuiu, o que vem enfatizar a tendência do AG em buscar a solução ótima para o problema.

5.4.1 Análise da Qualidade da Solução

Inicialmente foi testada a implementação do AG com taxa de mutação fixa em 0,001, seguindo a sugestão de Goldberg (1989) para o valor dessa taxa. Além disso, o operador cruzamento é o operador padrão, com taxa de cruzamento igual a 1, ou seja, todos os elementos da população de pais sofrem cruzamento. Os resultados obtidos são apresentados na tabela 6.

Tabela 6 - Resultados Computacionais do AG Padrão

10 gerações	\$278.004	\$283.003	\$284.852
50 gerações	\$283.942	\$282.549	\$284.294
100 gerações	\$278.147	\$281.996	\$285.393

A análise desses resultados pode ser vista na figura 47. Pode-se observar que, independente do número de gerações, sempre os resultados com população de 100 indivíduos são melhores que os resultados com população de 50 indivíduos, que por sua vez são melhores que os resultados com população de 10 indivíduos. O aumento

no número de gerações não implicou em uma melhora na qualidade da solução, sendo que houve até uma piora na qualidade da solução encontrada, de maneira geral. Isso porque a taxa de mutação é fixa e muito pequena, ocorrendo o mesmo número de mutações nas três situações, com dez gerações, com cinquenta gerações e com cem gerações. A diversidade genética, ligada ao tamanho da população, foi mais determinante na qualidade da solução encontrada .

Figura 47 - Qualidade das Soluções com AG com Taxa de Cruzamento Fixa e Taxa de Mutação Fixa

Utilizando a taxa de mutação exponencial proposta, mas o operador cruzamento tradicional, os resultados obtidos são listados na tabela 7.

Tabela 7 - Resultados Computacionais do AG com Taxa de Mutação Exponencial e Cruzamento Padrão

10 gerações	\$276.462	\$280.218	\$284.457
50 gerações	\$277.763	\$284.612	\$285.219
100 gerações	\$277.383	\$282.962	\$284.196

Na figura 48 pode ser vista a análise dos resultados computacionais referentes a execução dessa variação do AG. Nessa implementação, pode-se perceber que a qualidade da solução tende a aumentar também com o aumento do número de gerações. Isso pode estar associado ao maior número de vezes em que o operador mutação é aplicado, aumentando a probabilidade de encontrar uma solução cada vez melhor.

Figura 48 - Qualidade das Soluções com AG com Taxa de Cruzamento Fixa e Taxa de Mutação Exponencial

A tabela 8 apresenta o conjunto de soluções encontrado aplicando AG com operador cruzamento analítico e taxa de mutação fixa em 0,001.

A análise gráfica desses resultados é apresentada na figura 49. Nessa implementação, a qualidade da solução encontrada volta a cair, considerando a análise de resultados da implementação anterior. Tal comportamento reforça a idéia de que a taxa de mutação exponencial propicia a descoberta de melhores soluções.

Tabela 8 - Resultados Computacionais do AG com Taxa de Mutação Fixa e Cruzamento Analítico

10 gerações	\$270.764	\$279.296	\$284.358
50 gerações	\$276.793	\$284.666	\$285.735
100 gerações	\$275.280	\$280.841	\$284.556

Figura 49 - Qualidade das Soluções com AG com Cruzamento Analítico e Taxa de Mutação Fixa

A tabela 9 traz os resultados da implementação do AG proposto, ou seja, taxa de mutação exponencial e operador cruzamento analítico.

Tabela 9 - Resultados Computacionais do AG com Taxa de Mutação Exponencial e Cruzamento Analítico

10 gerações	\$278.636	\$284.360	\$285.499
50 gerações	\$283.954	\$286.404	\$286.112
100 gerações	\$285.349	\$286.175	\$286.354

O gráfico da figura 50 mostra a análise feita desses resultados. Nele pode ser observado que, tanto o número de indivíduos na população quanto o número de gerações estão influenciando na qualidade da solução encontrada pelo AG. Esse comportamento reforça novamente a idéia de que a taxa de mutação exponencial contribui para a melhora da qualidade da solução proposta pelo AG.

Figura 50 - Qualidade das Soluções com AG com Cruzamento Analítico e Taxa de Mutação Fixa

A tabela 10 mostra os resultados de cada uma das quatro implementações, considerando uma população com 10 indivíduos. A análise gráfica de tais resultados pode ser vista na figura 51. O gráfico mostra que, mesmo em uma população pequena, o que significa pouca diversidade genética, o AG proposto, com taxa de mutação exponencial e cruzamento analítico, apresenta qualidade de solução superior a qualquer outra implementação analisada.

Tabela 10 - Resultados Computacionais de Cada Implementação do AG com População com 10 Indivíduos

10 gerações	\$278.004	\$276.462	\$270.764	\$278.636
50 gerações	\$283.942	\$277.763	\$276.793	\$283.954
100 gerações	\$278.147	\$277.383	\$275.280	\$285.349

Figura 51 - Qualidade das Soluções com 10 Indivíduos na População

Na tabela 11 são apresentados os resultados computacionais, considerando uma população com 50 indivíduos. No gráfico da figura 52 pode-se perceber a mesma tendência dos resultados da análise anterior, entretanto, a qualidade das soluções encontradas pelo AG proposto é maior, considerando esse tamanho de população. Isso nem sempre ocorre nas outras implementações.

Tabela 11 - Resultados Computacionais de Cada Implementação do AG com População com 50 Indivíduos

10 gerações	\$283.003	\$280.218	\$279.296	\$284.360
50 gerações	\$282.549	\$284.612	\$284.666	\$286.404
100 gerações	\$281.996	\$282.962	\$280.841	\$286.175

Figura 52 - Qualidade das Soluções com 50 Indivíduos na População

Por fim, na tabela 12 e na figura 53 são apresentados, respectivamente, os resultados e a análise desses resultados, considerando uma população com 100 indivíduos.

Tabela 12 - Resultados Computacionais de Cada Implementação do AG com População com 100 Indivíduos

10 gerações	\$284.852	\$284.457	\$284.358	\$285.499
50 gerações	\$284.294	\$285.219	\$285.735	\$286.112
100 gerações	\$285.393	\$284.196	\$284.556	\$286.354

Figura 53 - Qualidade das Soluções com 100 Indivíduos na População

Percebe-se aqui a mesma tendência anterior, ou seja, a implementação do AG tradicional, com taxa de mutação fixa e taxa de cruzamento fixa, apresenta uma comportamento um tanto aleatório, quando aumenta o número de gerações. As implementações intermediárias apresentam uma pequena tendência em direção ao aumento da qualidade da solução à medida que cresce o número de gerações. Por

outro lado, o AG proposto apresenta uma tendência mais acentuada para alcançar um conjunto melhor de soluções, à medida que aumenta o tamanho da população e o número de gerações. Esses resultados vêm confirmar a validade da proposta de AG apresentada.

Entretanto, alguns fatores influenciam negativamente a busca pela melhor solução, a partir de uma certa configuração. Tais fatores estão relacionados à homogeneidade da população a partir de um certo momento do processo evolutivo, que, de certa maneira, causam uma saturação no processo, impedindo a melhora na qualidade da solução. Esse problema aparece em todas as configurações de AG estudadas, variando apenas o momento em que começam a acontecer. Essa variação depende dos operadores usados, principalmente o operador mutação. Pode-se perceber que tal problema, de queda na qualidade da solução, é bem menor, além de aparecer bem mais tarde, no AG proposto, com mutação exponencial. Isso pode ser entendido como sendo a característica da mutação de provocar a diversidade genética, evitando que o processo evolutivo rapidamente convirja para soluções subótimas.

5.4.2 Análise do Tempo de Execução

Para análise de tempo de processamento, foram comparados os resultados do AG proposto (com mutação exponencial e cruzamento analítico) e os resultados de cada uma das implementações feitas.

A primeira comparação feita foi do AG proposto, com taxa de mutação exponencial e cruzamento analítico, e do AG tradicional, com taxa de mutação e de

cruzamento fixas. Inicialmente, o que se esperava era que o AG proposto tivesse um menor tempo de processamento, visto que o operador mutação é aplicado um número maior de vezes.

A tabela 13 mostra os resultados das duas implementações, considerando populações com 10 indivíduos, 50 indivíduos e 100 indivíduos.

As figura 54, 55 e 56 mostram, respectivamente, a análise do tempo de processamento dessas implementações, considerando vários tamanhos de população. Quando o tempo de execução de uma determinada configuração é igual a zero, ele não é desenhado no gráfico.

Tabela 13 - Tempos de Processamento do AG tradicional e do AG proposto

		10 gerações	1s	0s
	50 gerações	2s	2s
	100 gerações	3s	3s
10 gerações	2s	2s
	50 gerações	8s	7s
	100 gerações	15s	14s
10 gerações	4s	4s
	50 gerações	26s	15s
	100 gerações	31s	29s

Figura 54 - Tempo de Processamento do AG Tradicional e do AG Proposto com População de 10 Indivíduos

Figura 55 - Tempo de Processamento do AG Tradicional e do AG Proposto com População de 50 Indivíduos

Figura 56 - Tempo de Processamento do AG Tradicional e do AG Proposto com População de 100 Indivíduos

Pode-se perceber que, em qualquer configuração, o AG proposto teve um desempenho igual ou melhor que o AG tradicional. A desigualdade em favor do AG proposto se acentua à medida que a população e o número de gerações crescem. Isso acontece, pois no decorrer do processo evolutivo, com a população cada vez mais homogênea, o cruzamento analítico permite que poucos cruzamentos ocorram de fato, o que não ocorre na implementação do AG tradicional, onde mesmo os pais iguais sofrem cruzamento, o que acarreta maior tempo de processamento. À medida que a população cresce, bem como o número de gerações considerado, o tempo gasto pelo operador cruzamento analítico, na análise dos pares de pais, é compensada pela não aplicação do cruzamento em si.

A comparação de tempo de processamento, considerando o AG proposto e o AG com taxa de mutação exponencial e taxa de cruzamento fixa, é apresentada na

tabela 14. Os gráficos das figuras 57, 58 e 59 mostram a análise do desempenho dessas duas implementações.

Tabela 14 - Tempos de Processamento do AG com taxa de mutação exponencial e taxa de cruzamento fixa e do AG proposto

		10 gerações	1s	0s
	50 gerações	1s	2s
	100 gerações	3s	3s
10 gerações	2s	2s
	50 gerações	7s	7s
	100 gerações	15s	14s
10 gerações	4s	4s
	50 gerações	15s	15s
	100 gerações	30s	29s

Figura 57 - Tempo de Processamento do AG com Taxa de Mutação Exponencial e Taxa de Cruzamento Fixa e do AG Proposto com População de 10 Indivíduos

Figura 58 - Tempo de Processamento do AG com Taxa de Mutação Exponencial e Taxa de Cruzamento Fixa e do AG Proposto com População de 50 Indivíduos

Figura 59 - Tempo de Processamento do AG com Taxa de Mutação Exponencial e Taxa de Cruzamento Fixa e do AG Proposto com População de 100 Indivíduos

Nessa análise, pode-se perceber que a tendência da análise anterior, de que o AG proposto tem um tempo de processamento menor que a implementação do AG com taxa de mutação exponencial e taxa de cruzamento fixa, permanece.

Entretanto, comparando os resultados do AG proposto com o AG com taxa de mutação fixa e cruzamento analítico, vistos na tabela 15, essa análise muda, como mostram os gráficos das figuras 60, 61 e 62.

Tabela 15 - Tempos de Processamento do AG com taxa de mutação fixa e cruzamento analítico e do AG proposto

		10 gerações	0s	0s
	50 gerações	1s	2s
	100 gerações	3s	3s
10 gerações	2s	2s
	50 gerações	9s	7s
	100 gerações	14s	14s
10 gerações	4s	4s
	50 gerações	15s	15s
	100 gerações	29s	29s

Figura 60 - Tempo de Processamento do AG com Taxa de Mutação Fixa e Cruzamento Analítico e do AG Proposto com População de 10 Indivíduos

Figura 61 - Tempo de Processamento do AG com Taxa de Mutação Fixa e Cruzamento Analítico e do AG Proposto com População de 50 Indivíduos

Figura 62 - Tempo de Processamento do AG com Taxa de Mutação Fixa e Cruzamento Analítico e do AG Proposto com População de 100 Indivíduos

Nessa análise percebe-se que o tempo de processamento é praticamente igual para as duas implementações, pois as duas implementações utilizam o operador cruzamento analítico. Entretanto, com taxa de mutação fixa, pode ocorrer um tempo de processamento menor, pois o operador mutação será aplicado um número menor de vezes favorecendo, na questão relacionada ao tempo de execução, a implementação do AG com cruzamento analítico e taxa de mutação fixa.

Com os resultados obtidos até aqui, o AG proposto apresenta vantagens sobre pelo menos duas das implementações estudadas. Entretanto, à medida que a população aumenta e o número de gerações considerado também aumenta, essa vantagem se torna mais evidente, conforme mostra a tabela 16.

Tabela 16 - Tempos de Processamento considerando populações de 100 e 500 indivíduos, e 500 e 1000 gerações

		500 gerações	146s	148s	134s	132s
	1000 gerações	528s	432s	260s	263s
500 gerações	584s	621s	338s	292s
	1000 gerações	1032s	1095s	577s	566s

Os gráficos das figura 63 e 64 mostram a análise de desempenho de cada implementação.

Figura 63 - Tempo de Processamento Considerando 100 Indivíduos na População

Figura 64 - Tempo de Processamento Considerando 500 Indivíduos na População

Esses resultados evidenciam que o AG proposto apresenta vantagens sobre qualquer outra implementação estudada, principalmente sobre o AG tradicional, o que valida sua proposta como apoio à simulação de sistemas complexos.

Considerando a simplicidade do exemplo estudado, fica claro que não é necessário uma população maior que 50 indivíduos e mais do que 10 gerações para obter uma boa solução, como apresentado anteriormente. Entretanto, o sistema aqui apresentado foi simplificado, para permitir a comparação dos resultados da aplicação da técnica de simulação apoiada por AG proposta com os resultados ótimos obtidos através da técnica de Programação Linear. Em sistemas reais, cuja complexidade se torna um problema de difícil equacionamento, Jones (1993) sugere que sejam definidos tamanho da população igual a 200 indivíduos e número de gerações igual a 1000. Com essa configuração, e considerando os resultados das análises aqui apresentadas, o método de simulação apoiada por AG aqui proposto se mostra a melhor opção para resolução do problema considerado, tanto em termos de qualidade da solução quanto em termos de tempo de processamento.

5.5 Análise da Aplicabilidade da Técnica

Neste capítulo foi apresentado um exemplo simplificado da modelagem do problema do balanceamento e produção de uma refinaria de petróleo, baseada em um grafo dirigido, onde pôde ser constatada a aplicabilidade da técnica de simulação apoiada por AG na resolução de problemas complexos.

Para a aplicação de AG no apoio à simulação, foram apresentados e analisados resultados computacionais, considerando o exemplo de modelagem descrito e variações na implementação do AG.

A partir das análises feitas, pode-se constatar que o Simulador apoiado pelo AG proposto apresentou melhores resultados na maioria das vezes, tanto em relação à qualidade da solução quanto em relação ao tempo de execução.

Para cada configuração foram executados dez testes, sendo que foi considerado o valor mais próximo da média dos resultados.

Também pode-se observar que, à medida em que o tamanho da população e o número de gerações aumentavam, o desvio padrão das soluções encontradas em cada uma das baterias de testes realizadas diminuiu, o que enfatiza a tendência do AG em buscar a solução ótima para o problema.

A implementação do AG proposto apresentou uma tendência mais acentuada para alcançar um conjunto melhor de soluções, à medida que o tamanho da população e o número de gerações aumentaram. Esses resultados confirmam a validade da proposta de AG apresentada.

Além disso, como o tempo de processamento é crescente, à medida em que cresce o tamanho da população e o número de gerações, pode-se fazer uma composição de tempo versus qualidade da solução que, nesse estudo de caso, pode ser de 50 indivíduos na população e 50 gerações.

Apesar do AG nem sempre encontrar a solução ótima para o problema, ele sempre encontra uma solução satisfatória, o que é aceitável quando se considera um problema complexo.

Existem técnicas de busca exatas, que dão a solução ótima. Entretanto, esses métodos convencionais muitas vezes não se aplicam a problemas complexos. O tratamento das restrições é dificultado, aumentando a complexidade, e até inviabilizando a implementação. Em AG, a cada restrição do problema, apenas o cálculo do desempenho de cada indivíduo é modificado, sem que a implementação da técnica se torne mais complexa.

Portanto, uma ferramenta computacional baseada em AG para simulação de sistemas complexos é um apoio extremamente importante para os projetistas de tais sistemas, uma vez que AG pode contornar o problema relacionado à complexidade do sistema modelado, e analisar os dados de entrada e de saída automaticamente, liberando o projetista de tal tarefa.

Capítulo 6

Conclusões

Uma das grandes dificuldades na utilização da simulação como uma ferramenta para auxiliar no projeto e operação de sistemas complexos é que os modelos de simulação baseados em métodos tradicionais são incapazes de gerar uma solução ótima por si só. Esses modelos devem ser suportados por um módulo gerador dos dados de entrada e um módulo de análise dos dados gerados pela simulação. Tanto os dados de entrada quanto os resultados da simulação, como destacou Pidd (1994), são difíceis de serem gerados e interpretados.

Algoritmo Genético (AG) foi apresentado aqui como uma alternativa para geração e análise dos dados de simulação. AG, sendo uma técnica de Inteligência Artificial, que trabalha na base da tentativa e erro e, utilizando análise de desempenho e operadores genéticos para gerar soluções, usa o conceito de retroalimentação do erro, aprendendo a produzir soluções cada vez melhores.

Sendo assim, o AG pode ser usado para liberar o projetista da tarefa de análise dos dados de entrada e dos dados de saída da simulação, que corresponde à tarefa mais difícil, quando são aplicadas técnicas convencionais de análise de dados de simulação.

6.1 Atividades Realizadas

Considerando que a simulação de sistemas complexos se torna inviável sem um suporte de avaliação de resultados para selecionar os dados de entrada e analisar os dados de saída gerados no processo de simulação, além de exigir um apoio computacional efetivo, foi apresentada uma proposta de utilização de AG para apoiar a simulação de sistemas complexos, juntamente com uma ferramenta computacional aplicando a técnica proposta na simulação de sistemas.

A utilização do AG, dentro do processo proposto, é exatamente na geração de dados para simulação e na análise dos dados gerados pela simulação, cuidando da retroalimentação do simulador com novos dados de entrada, de acordo com a análise feita dos dados gerados pela simulação.

A ferramenta computacional foi desenvolvida e apresentada como uma versão protótipo, com o objetivo de mostrar a viabilidade da aplicação da técnica de AG no apoio à tarefa de geração e análise de dados de simulação de sistemas complexos.

Esse trabalho poderia ter sido desenvolvido utilizando um simulador comercial genérico, tais como ARENA ou AutoMode. Entretanto, por razões de tempo e recursos financeiros escassos, optou-se pelo desenvolvimento de um simulador

simples, que permitisse a realização de experimentos, visando testar e mostrar a potencialidade do AG como suporte efetivo na tarefa de simulação.

Para a implementação da técnica proposta, através do desenvolvimento da ferramenta, foram elaboradas interfaces amigáveis, em um ambiente computacional avançado, e utilizadas estruturas de dados dinâmicas, visando, futuramente, uma expansão do escopo dessa ferramenta.

Também foram desenvolvidas variações dos operadores genéticos, como a mutação exponencial e o cruzamento analítico, visando aumentar a eficiência do AG.

Uma política de penalização robusta foi apresentada, garantindo a factibilidade da solução encontrada pelo AG proposto.

Foi apresentado um exemplo de aplicação do AG no processo de simulação, através da modelagem e simulação de um sistema complexo simplificado, com solução ótima conhecida através do método de programação linear. Dessa maneira, os resultados obtidos com a aplicação do método proposto puderam ser comparados com a solução ótima, gerando análises de desempenho e eficácia do processo.

Foi apresentado um exemplo simplificado da modelagem do problema do balanceamento e produção de uma refinaria de petróleo, baseada em um grafo dirigido, onde pôde ser constatada a aplicabilidade da técnica de simulação apoiada por AG na resolução de problemas complexos, através da geração dos dados de entrada para simulação e análise dos dados gerados pela simulação.

As atividades desenvolvidas durante a realização deste trabalho serviram de base para a publicação de quatro artigos: um estudo da aplicação de AG ao problema do corte de barras, em Freitas et al. (1993); proposta de utilização de um AG para resolução do problema do balanceamento de cargas, em Freitas et al. (1994); estudo da aplicabilidade de AG na simulação de sistemas complexos, em Silva; Kirner

(1996); e a apresentação do protótipo da ferramenta baseada em AG para simulação de sistemas complexos, em Silva et al. (1996). Além dos artigos, foram publicados um relatório técnico, em Santos et al. (1995), e os seguintes resumos em congressos de iniciação científica: Carpinetti; Anacleto (1993), Santos; Anacleto (1994), Carpinetti; Anacleto (1994), Aluizio; Silva (1995), Miranda; Silva (1995), Miranda et al. (1995), Aluizio; Silva (1996), Miranda; Silva (1996), Aluizio et al. (1996).

6.2 Principais Contribuições

Este trabalho apresentou e avaliou a aplicabilidade da técnica de AG no apoio à simulação de sistemas complexos. Tal estudo contribuiu para a especificação e proposta de utilização de uma certa configuração de AG na geração e análise de dados de simulação. A partir dessa proposta, foram desenvolvidas e apresentadas as seguintes contribuições:

- Um esquema de utilização de AG acoplado a um simulador, que permite, de forma clara e simples, o entendimento com relação à contribuição que o AG pode dar ao processo de simulação, gerando os dados para o simulador e analisando os dados gerados pela simulação.
- Um ambiente computacional gráfico baseado em AG para simulação de sistemas complexos.

6.3 Análise de Resultados

Para a aplicação de AG no apoio à simulação, foram apresentados e analisados resultados computacionais, considerando o exemplo de modelagem descrito, sua solução através de um método matemático exato, e variações na implementação do AG.

A partir das análises feitas, pôde-se constatar que o SimAG - simulador apoiado por AG - apresentou melhores resultados na maioria das vezes, tanto em relação à qualidade da solução quanto em relação ao tempo de execução, quando comparado com as outras implementações de AG estudadas.

Para cada configuração foram executados dez testes, sendo que foi considerado o valor mais próximo da média dos resultados.

Também pôde-se observar que, à medida em que o tamanho da população e o número de gerações aumentavam, o desvio padrão das soluções encontradas em cada uma das baterias de testes realizadas diminuiu, o que enfatiza a tendência do AG em buscar a solução ótima para o problema.

A implementação do AG proposto apresentou uma tendência mais acentuada para alcançar conjunto melhor de soluções, à medida que o tamanho da população e o número de gerações aumentaram. Esses resultados confirmam a validade da proposta de AG apresentada.

Entretanto, alguns fatores influenciaram negativamente na busca pela melhor solução, a partir de uma certa configuração, gerando uma queda na qualidade da solução. Tais fatores estão relacionados ao problema da homogeneidade da

população, a partir de um certo momento do processo evolutivo, causando uma saturação no processo, impedindo a melhora na qualidade da solução. Esse problema aparece em todas as configurações de AG estudadas, estando relacionado com problemas de limites do desempenho, mais especificamente com o problema de finalização lenta, abordado no capítulo de revisão do AG.

Entretanto, tal problema, de queda na qualidade da solução, foi bem menor, além de ter aparecido bem mais tarde, no AG proposto, com mutação exponencial e cruzamento analítico. Isso pode ser entendido como sendo a característica da mutação de provocar a diversidade genética, evitando que o processo evolutivo rapidamente convirja para soluções subótimas.

Para solucionar o problema de finalização lenta, os operadores aplicados devem ser melhor estudados e aprimorados, principalmente o operador seleção e o operador mutação, como sugere Beasley et al. (1993a).

Além disso, como o tempo de processamento é crescente, à medida em que cresce o tamanho da população e o número de gerações, pode-se fazer uma composição de tempo versus qualidade da solução, na tentativa de otimizar, não só o sistema em questão, mas também o tempo para a resolução desse problema.

Como o AG, na maioria das vezes, encontra uma solução quase ótima, tal resultado é aceitável e interessante, quando se considera um problema complexo. Para esse tipo de problema, os métodos convencionais muitas vezes não se aplicam, pois existe um grande número de restrições matemáticas. O tratamento dessas restrições é dificultado, aumentando a complexidade ou inviabilizando a implementação. Com AG, isso não acontece, ou seja, a sua implementação não se torna mais complexa.

Utilizando a técnica de AG, pode-se caminhar para a solução ótima, sem necessidade de análise mais detalhada dos dados de simulação. O AG faz a

realimentação dos dados de entrada a partir da análise de desempenho com os dados de saída, sem precisar de maiores conhecimentos sobre o domínio do problema.

Portanto, pôde-se verificar a generalidade e a aplicabilidade da técnica de AG para apoiar o processo de simulação de sistemas complexos, e ainda obter uma solução ótima ou quase ótima para o problema, o que nem sempre é possível com métodos exatos convencionais.

Assim, uma ferramenta computacional baseada em AG para simulação de sistemas complexos é um apoio extremamente importante para os projetistas de tais sistemas, uma vez que AG pode contornar o problema relacionado com a complexidade do sistema modelado, e analisar os dados de entrada e de saída automaticamente, liberando o projetista de tal tarefa e de possíveis erros nessa análise.

6.4 Trabalhos Futuros

Como parte dos trabalhos futuros, o escopo da ferramenta para simulação de sistemas complexos apoiada por AG deverá ser ampliado, o que permitirá a modelagem e simulação de sistemas maiores, com maior número de componentes. Isso se fará através do uso de estruturas de dados mais complexas para representar o sistema como um grafo, considerando um número qualquer de arcos de entrada em cada nó do grafo, e um número qualquer de arcos de saídas.

Também serão desenvolvidos novos operadores genéticos, tais como inversão e cruzamento multiponto, que deverão ficar armazenados em uma base de dados de operadores. O usuário, então, conhecendo um pouco do domínio do problema em

questão, poderá escolher os operadores genéticos mais apropriados para resolução de tal situação. Se for de seu interesse, o usuário poderá também definir novos operadores genéticos, que serão incluídos nessa base de dados.

Além disso, a função objetivo considerada poderá ser qualquer função, e não apenas funções lineares.

Conseqüentemente, o módulo de visualização deverá utilizar recursos de computação gráfica para poder mostrar ao usuário o comportamento dessas funções em gráficos tridimensionais ou n-dimensionais.

Também será estudada uma forma de explorar as características intrínsecas de paralelismo da técnica de AG, através do uso de redes de computadores, seguindo sugestão feita por Srivastava et al. (1994). Tal estudo tem por objetivo tornar mais rápido o processo de busca da solução para o problema em questão, através do uso de uma abordagem distribuída na implementação do AG.

Outro aspecto a ser explorado será o uso de Realidade Virtual no processo de simulação. Essa ferramenta, acoplada à simulação do modelo, deverá permitir a animação do processo simulado, mostrando o sistema em um ambiente tridimensional. Utilizando recursos de realidade virtual, o projetista poderá navegar nesse ambiente tridimensional referente ao exemplo modelado fisicamente.

Esse módulo permitirá a construção do ambiente virtual referente ao exemplo simulado, viabilizando o controle de variáveis, a análise de resultados, e a visualização tridimensional. Isso dará ao projetista de sistemas complexos um apoio gráfico efetivo com a possibilidade de uso de recursos de realidade virtual e multimídia.

Referências Bibliográficas

- ALUIZIO, F.; ABREU, J.C.; SILVA, J. C. A. Um Ambiente Gráfico para Simulação de Sistemas Complexos com Algoritmo Genético - Módulos de Geração e Análise Estatística. In: IV CIC, UFSCar, São Carlos, SP, 1996. *Anais*. São Carlos, SP, 1996, p. 101. /Resumo/
- ALUIZIO, F.; MIRANDA, R. Z.; ZANATA, P.; ABREU, J.C.; SILVA, J. C. A. Ferramenta Para Simulação de Sistemas Com Algoritmo Genético. In: XV CICTE, EESC-USP, São Carlos, SP, 1996. *Anais*. São Carlos, SP, 1996. /Resumo/ /No Prelo/
- ALUIZIO, F.; SILVA, J. C. A. Um Ambiente Paralelo de Software para Desenvolvimento de Sistemas com Algoritmos Genéticos. In: III CIC, UFSCar, São Carlos, SP, 1995. *Anais*. São Carlos, SP, 1995, p. 152. /Resumo/
- BAGCHI, S.; UCKUN, S.; MIYABE, Y.; KAWAMURA, K. Exploring Problem-specific Recombination Operators for Job Shop Scheduling. In: FOURTH INTERNATIONAL CONFERENCE ON GENETIC ALGORITHMS, San

- Mateo, California, 1991. *Proceedings*. San Mateo, California, 1991. p.10-17.
- BANKS, J. Selecting Simulation Software. In: WINTER SIMULATION CONFERENCE, Phoenix, Arizona, 1991. *Proceedings*. Phoenix, Arizona, 1991. p.15-20.
- BEASLEY, D.; BULL, D.R. ; MARTIN, R.R. An Overview of Genetic Algorithm: Part 1 Fundamentals. *University Computing* , v.15, n.2, p.58-69, 1993a.
- BEASLEY, D.; BULL, D.R.; MARTIN, R.R. An Overview of Genetic Algorithm: Part 2 Research Topics. *University Computing* , vol. 15, n. 4, pp. 170-181, 1993b.
- BOOKER, L.B.; GOLDBERG, D.E.; HOLLAND, J.H. Systems and Genetic Algorithms. *Artificial Intelligence*, n.40, p.235-282, 1989.
- BRUNS, R. Direct Chromosome Representation and Advanced Genetic Operators for Production Scheduling. In: FIFTH INTERNATIONAL CONFERENCE ON GENETIC ALGORITHMS, USA, 1993. *Proceedings*. USA, 1993. p.352-359.
- CALDWELL, C.; JOHNSTON, V.S. Tracking a Criminal Suspect through "Face-Space" with a Genetic Algorithm. In: FOURTH INTERNATIONAL CONFERENCE ON GENETIC ALGORITHMS, San Mateo, California, 1991. *Proceedings*. San Mateo, California, 1991. p.416-421.
- CANGUSSU, J.W.L. *Execução Programada de Statecharts*. São Carlos, 1993, 146p. Dissertação (Mestrado) - Instituto de Ciências Matemáticas de São Carlos, Universidade de São Paulo.
- CARNEIRO, S.A.; ALVARENGA, A.G.; DAZA, V.M.P. Problema do Corte, via Algoritmo Genético. In: XXVI SIMPÓSIO BRASILEIRO DE PESQUISA OPERACIONAL. Florianópolis, SC, 1994. *Anais*. Florianópolis, SC, 1994. p.734-739.

- CARPINETTI, B.; ANACLETO, J. C. Aplicação de Algoritmos Genéticos na Especificação de Sistemas Complexos. In: XII CICTE, EESC-USP, São Carlos, SP, 1993. *Anais*. São Carlos, SP, 1993, p. 175. /Resumo/
- CARPINETTI, B.; ANACLETO, J. C. Especificando Sistemas de Manufatura com Algoritmos Genéticos. In: XIII CICTE, EESC-USP, São Carlos, SP, 1994. *Anais*. São Carlos, SP, 1994, p. 231. /Resumo/
- CARSE, B.; FORGATY, T.C. A New Approach to Genetics Based Machine Learning in Fuzzy Controller Design. In: 1994 IEEE INTERNATIONAL SYMPOSIUM ON INTELLIGENT CONTROL. Columbus, Ohio, USA, 1994. *Proceedings*. Columbus, Ohio, USA, 1994. p.231-236.
- CELKO, J. Genetic Algorithms and Database Indexing. *Dr. Dobb's Journal*, p.30-34, abr. 1993.
- CLEVELAND, G.A.; SMITH, S.F. Using Genetic Algorithms to Schedule Flow Shop Release. In: THIRD INTERNATIONAL CONFERENCE ON GENETIC ALGORITHMS, Los Altos, California, 1989. *Proceedings*. Los Altos, California, USA, 1989. p.160-169.
- DÁVALOS, R.V.; STANGE, P. Uma Comparação entre Otimização Global via Algoritmos Genético e via GAMS. In: XXVI SIMPÓSIO BRASILEIRO DE PESQUISA OPERACIONAL, Florianópolis, 1994. *Anais*. Florianópolis, 1994. p.696-701.
- DAVIS, L. *Handbook of Classifier Genetic Algorithms*. New York, Van Nostrand Reinhold, 1991.
- EDGAR, T.F.; HIMMELBLAU, D.M. *Optimization of Chemical Processes*. New York, McGraw-Hill, 1988.
- EFE, K. Heuristic Models of Task Assignment Scheduling in Distributed Systems.

- IEEE Computer*, p.50-56, jun. 1982.
- FANG, H.; ROSS, P.; CORNE, D. A Promising Genetic Algorithm Approach to Job-Shop Scheduling, Rescheduling, and Open-Shop Scheduling Problems. In: FIFTH INTERNATIONAL CONFERENCE ON GENETIC ALGORITHMS, USA, 1993. *Proceedings*. USA, 1993. p.375-382.
- FISHWICK, P.A. Simulation Model Design. In: WINTER SIMULATION CONFERENCE, Lake Buena Vista, Florida, USA 1994. *Proceedings*. Lake Buena Vista, Florida, USA 1994. p.173-175.
- FORGATY, T.C. Rule-based Optimization of Combustion in Multiple Burner Furnances and Boiler Plants. *Engineering Applications of Artificial Intelligence*, v.1, n.3, p.203-209, 1988.
- FRASER, J. Finite Scheduling and Manufacturing Synchronization: Tools for Real Plant Productivity. *IIE Solutions*, p.44-53, set. 1995.
- FREITAS, A. A.; ANACLETO, J. C.; KIRNER, C. Applying Genetic Algorithms to the Load-Balancing Problem. In: BAEZA-YATES, R., coord. *Computer Science 2: Research and Applications*. New York, Plenum Press, 1994. p.7-13.
- FREITAS, A. A.; ANACLETO, J. C.; MORÁBITO NETO, R.; KIRNER, C. Algoritmos Genético e sua Aplicação ao Problema do Corte de Barras. In: I SIMPÓSIO BRASILEIRO DE AUTOMAÇÃO INTELIGENTE, Rio Claro, 1993. *Anais*. Rio Claro, UNESP, 1993. p.38-47.
- GARIS, H. Genetic Programming: Building Artificial Nervous Systems using Genetically Programmed Neural Network Modules. In: MACHINE LEARNING: SEVENTH INTERNATIONAL CONFERENCE, Austin, Texas, USA, 1990. *Proceedings*. Austin, Texas, USA, University of Texas, 1990. p.132-139.
- GLOVER, F.; GREENBERG, H.J. New Approaches for Heuristic Search: A

- Bilateral Linkage with Artificial Intelligence. *European Journal of Operational Research*, n.39, p.119-130, 1989.
- GOGG, T.J.; MOTT J.R.A. Introduction to Simulation. In: WINTER SIMULATION CONFERENCE, Los Angeles, California, USA, 1993. *Proceedings*. Los Angeles, California, USA, 1993. p.9-17.
- GOLDBERG, D.E. A Tale of Two Problems: Broad and Efficient Optimization Using Genetic Algorithms. In: 1986 SUMMER COMPUTER SIMULATION CONFERENCE, Reno, Nevada, 1986. *Proceedings*. Reno, Nevada, The Society for Computer Simulation, 1986. p.44-48.
- GOLDBERG, D.E. Genetic Algorithms as a Computational Theory of Conceptual Design. *Applications of Artificial Intelligence in Engineering*, v.6, p.3-16, 1991.
- GOLDBERG, D.E. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, Massachusetts, Addison-Wesley, 1989.
- GOLDBERG, D.E. Genetic and Evolutionary Algorithms Come of Age. *Communications of the ACM*, v.37, n.3, p.113-119, 1994.
- GOLDBERG, D.E. Making Genetic Algorithms Fly: A Lesson from the Wright Brothers. *Advanced Technology for Developers*, v.2, p.1-8, fev. 1993.
- GOLDBERG, D.E.; DEB, K.; CLARK, J.H. Genetic Algorithms, Noise, and the Sizing of Populations. *Complex Systems*, v.6, n.8, p.333-362, ago. 1992.
- GOLDBERG, D.E.; DEB, K.; KARGUPTA, H.; HARIK, G. *Rapid, Accurate Optimization of Difficult Problems Using Fast Messy Genetic Algorithms*. Urbana, Illinois, USA, Univ. of Illinois at Urbana-Champaign, 1993. (IlliGAL Report, 93004)
- GOLDBERG, D.E.; KUO, C. H. Genetic Algorithms in Pipeline Optimization. *Journal of Computing in Civil Engineering*, v.1, n.2, p.128-141, abr. 1987.

- GOLDBERG, D.E.; SMITH, R.E. *AI Meets OR: Blind Inferential Search Using Genetic Algorithms*. Tuscaloosa, USA, Univ. of Alabama, 1986. (TCGA Report, 86002)
- GORGES-SCHLEUTER, M. ASPARGOS: An Asynchronous Parallel Genetic Optimization Strategy. In: THIRD INTERNATIONAL CONFERENCE ON GENETIC ALGORITHMS, Los Altos, California, 1989. *Proceedings*. Los Altos, California, USA, 1989. p.422-427.
- GREFENSTETTE, J.J. Optimization of Control Parameters for Genetic Algorithms. *IEEE Transaction on Systems, Man, and Cybernetics*, v.SM-16, n.1, p.122-128, 1986.
- HEITKOETTER, J. *The Hitch-Hiker's Guide to Evolutionary Computation*. FAQ in comp.ai.genetic, Internet, 1993.
- HOLLAND, J.H. *Adaptation in Natural and Artificial Systems*. Michigan, Univ. of Michigan, 1975.
- HORN, J.; GOLDBERG, D.E.; DEB, K. *Research Note: LongPath Problems for Mutation-Based Algorithms*. Urbana, Illinois, USA, Univ. of Illinois at Urbana-Champaign, 1992. (IlliGAL Report, 92011)
- JONES, A. .J. Genetic Algorithms and their Applications to the Design of Neural Networks. *Neural Computing & Applications*, vol.1, n.1, p.32-45, 1993.
- JONG, K.D.; SPEARS, W. On the State of Evolutionary Computation. In: FIFTH INTERNATIONAL CONFERENCE ON GENETIC ALGORITHMS, USA, 1993. *Proceedings*. USA, 1993. p.618-623.
- KARR, C. Genetic Algorithm for Fuzzy Controlers. *AI Expert*, v.6, n.2, p.26-33, fev. 1991.
- KELTON, W.D. Analysis of Output Data. In: WINTER SIMULATION

- CONFERENCE, Lake Buena Vista, Florida, USA, 1994. *Proceedings*. Lake Buena Vista, Florida, USA 1994. p.62-68.
- KOZA, J. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, Mass., MIT Press, 1992.
- KRÖGER, B.; SCHWENDERLING, P.; VOMBERGER, O. Parallel Genetic Packing of Rectangles. In: PARALLEL PROBLEM SOLVING FROM NATURE I - L.N.C.S., USA, 1991. *Proceedings*. USA, 1991. p.160-164.
- LAW, A.; KELTON, W.D. *Simulation Modeling and Analysis*. New York, McGraw-Hill, 1982.
- LAW, A.M.; MCCOMAS, M.G. Secrets of Successful Simulation Studies. In: WINTER SIMULATION CONFERENCE, Phoenix, Arizona, USA, 1991. *Proceedings*. Phoenix, Arizona, USA, 1991. p. 21-27.
- LAWTON, G. Genetic Algorithms for Schedule Optimization. *AI Expert*, p.23-27, mai. 1992.
- LEEMIS, L.M.. Input Modeling. In: WINTER SIMULATION CONFERENCE, Lake Buena Vista, Florida, USA, 1994. *Proceedings*. Lake Buena Vista, Florida, USA 1994. p.55-61.
- LEVITIN, G.; RUBINOVITZ, J. Genetic Algorithm for Linear and Cyclic Assignment Problem. *Computers Ops. Res.*, v.20, n.6, p.575-586, 1993.
- LIEPINS, G. E.; HILLIARD, M.R.; PALMER, M.; MORROW, M. Greedy Genetics. In: SECOND INTERNATIONAL CONFERENCE ON GENETIC ALGORITHMS, USA, 1987. *Proceedings*. USA, 1987. p.90-99.
- McCALLUM, R.A.; SPACKMAN, K.A. Using Genetic Algorithms to Learn Disjunctive Rules from Examples. In: MACHINE LEARNING: SEVENTH INTERNATIONAL CONFERENCE, Austin, Texas, USA, 1990. *Proceedings*.

- Austin, Texas, USA, University of Texas, 1990. p.149-152.
- McKINNEY, D.C.; LIN, M.D. Genetic Algorithm Solution of Groundwater Management Models. *Water Resources Research*, v.30, n.6, p.1897-1906, jun. 1994.
- MIRANDA, R. Z.; ALUIZIO, F.; SILVA, J. C. A. Um Ambiente Gráfico para Modelagem e Simulação de Sistemas Complexos. In: XIV CICTE, EESC-USP, São Carlos, SP, 1995. *Anais*. São Carlos, SP, 1995, p. 234. /Resumo/
- MIRANDA, R. Z.; SILVA, J. C. A. Um Editor Gráfico para Apoio à Modelagem e Simulação de Sistemas. In: III CIC, UFSCar, São Carlos, SP, 1995. *Anais*. São Carlos, SP, 1995, p. 173. /Resumo/
- MIRANDA, R. Z.; ZANATA, P.; SILVA, J. C. A. Um Ambiente Gráfico para Simulação de Sistemas Complexos com Algoritmo Genético - Módulos de Execução e Visualização da Evolução. In: IV CIC, UFSCar, São Carlos, SP, 1996. *Anais*. São Carlos, SP, 1996, p. 84. /Resumo/
- MURRAY, D. Tuning Neural Networks with Genetic Algorithms. *AI Expert*, p.27-31, jun. 1994.
- OLIVER, J. Finding Decision Rules with Genetic Algorithms. *AI Expert*, p.33-39, mar. 1994.
- PIDD, M. An Introduction to Computer Simulation. In: WINTER SIMULATION CONFERENCE, Lake Buena Vista, Florida, USA, 1994. *Proceedings*. Lake Buena Vista, Florida, USA 1994. p.7-14.
- PIPE, A.G.; JIN, Y.; WINFIELD, A. A Hybrid Adaptive Heuristic Critic Architecture for Learning in Large Static Search Spaces. In: 1994 IEEE INTERNATIONAL SYMPOSIUM ON INTELLIGENT CONTROL. Columbus, Ohio, USA, 1994. *Proceedings*. Columbus, Ohio, USA, 1994. p.237-242.

- PORTER II, L.M.L.; PASSINO, K.M. Genetic Model Reference Adaptive Control. In: IEEE INTERNATIONAL SYMPOSIUM ON INTELLIGENT CONTROL, Columbus, Ohio, USA, 1994. *Proceedings*. Columbus, Ohio, USA, 1994. p.219-224.
- PRITSKER, A.A.; HENRIKSEN, J.O.; FISHWICK, P.A.; CLARK, G.M. Principles of Modeling. In: WINTER SIMULATION CONFERENCE, Phoenix, Arizona, USA, 1991. *Proceedings*. Phoenix, Arizona, USA, 1991. p. 1199-1207.
- REID, D.J. *Enhanced Genetic Operators for the Resolution of Discrete Optimization Problems*. Brisbane, Queensland, Australia, Univ. of Queensland, out. 1994b. (Technical Report, 312)
- REID, D.J. *Genetic Algorithms in Constrained Optimization*. Brisbane, Queensland, Australia, Univ. of Queensland, mar. 1994a. (Technical Report, 298)
- RIBEIRO FILHO, J.L.; TRELEAVING, P.C. Genetic-Algorithm Programming Environments. *Computer*, p.28-43, jun. 1994.
- RICH, E. *Inteligência Artificial*. São Paulo, McGraw-Hill, 1988.
- RICHARDSON, J.T.; PALMER, M.R.; LIEPINS, G.E.; HILLIARD, M.R. Some Guidelines for Genetic Algorithms with Penalty Functions. In: THIRD INTERNATIONAL CONFERENCE ON GENETIC ALGORITHMS, Los Altos, California, 1989. *Proceedings*. Los Altos, California, USA, 1989. p.191-197.
- RITZEL, B.J.; EHEART J.W.; RANJITHAN, S. Using Genetic Algorithms to Solve a Multiple Objective Groundwater Pollution Containment Problem. *Water Resources Research*, v.30, n.5, p.1589-1603, 1994.
- SADOWSKI, R.P. Avoiding The Problems and Pitfalls in Simulation. In: WINTER SIMULATION CONFERENCE, Phoenix, Arizona, USA, 1991. *Proceedings*. Phoenix, Arizona, USA, 1991. p. 48-55.

- SANTOS, A.; ANACLETO, J. C. Algoritmos Genéticos Aplicados à Modelagem de Sistemas. In: II CIC, UFSCar, São Carlos, SP, 1994. *Anais*. São Carlos, SP, 1994, p. 23. /Resumo/
- SANTOS, A.; CARPINETTI, B.; ANACLETO, J. C. KIRNER, C. Uma Implementação de Algoritmos Genéticos em Ambiente Computacional Paralelo. São Carlos, DC-UFSCar, 1995. (Relatório Técnico DC-UFSCar, 006/95)
- SARGENT, R.G. Verification and Validation of Simulation Models. In: WINTER SIMULATION CONFERENCE, Lake Buena Vista, Florida, USA, 1994. *Proceedings*. Lake Buena Vista, Florida, USA 1994. p.77-87.
- SHANNON, R. E. Introduction to Simulation. In: WINTER SIMULATION CONFERENCE, Arlington, Virginia, USA 1992. *Proceedings*. Arlington, Virginia, USA 1992. p.65-81.
- SIKORA, R. Learning Control Strategies for Chemical Processes. *IEEE Expert*, p.35-43, jun. 1992.
- SILVA, J. C. A.; ALUIZIO, F.; ABREU, J. C.; ZANATA, P.; MIRANDA, R. Z.; KIRNER, C. Uma Ferramenta para Simulação de Sistemas Complexos Baseada em Algoritmo Genético. In: VII Semana de Informática, UEM, Maringá, PR, 1996. *Anais*. UEM, Maringá, PR, 1996. /No prelo/
- SILVA, J. C. A.; KIRNER, C. Applying Genetic Algorithms to the Simulation of Complex Systems. In: ISCA 5th International Conference on Intelligent Systems, Reno, Nevada, 1996. *Proceedings*. Reno, Nevada, USA, 1996, p. 135-139.
- SIMPSON, A.R.; DANDY, G.C.; MURPHY, L.J. Genetic Algorithms Compared to Other Techniques for Pipe Optimization. *Journal of Water Resources Planning and Management*, v.120, n.4, p.423-443, jul./ago. 1994.
- SIMS, K. Evolving Virtual Creatures. In: COMPUTER GRAPHICS. Orlando,

- Florida, USA, 1994. *Proceedings*. Orlando, Florida, USA, 1994. p.15-22.
- SRIVASTAVA, A.; KUMAR, A.; PATHAK, R. M. Distributed Approach for Implementing Genetic Algorithms. In: 1994 INTERNATIONAL CONFERENCE ON PARALLEL PROCESSING, Pennsylvania, 1994. *Proceedings*. Pennsylvania State University, Pennsylvania, USA, 1994. vol III, p. 106-109.
- SWAI, J. Simulation Software Survey. *OR/MS Today*, v.18, n.5, p.81-102, 1991.
- TAM, K.Y. Genetic Algorithms, Function Optimization, and Facility Layout Design. *European Journal of Operational Research*, v.63, p.322-346, 1992.
- THEN, T.W.; CHONG, E.K.P. Genetic Algorithm in Noisy Environment. In: 1994 IEEE INTERNATIONAL SYMPOSIUM ON INTELLIGENT CONTROL. Columbus, Ohio, USA, 1994. *Proceedings*. Columbus, Ohio, USA, 1994. p.225-230.
- THESEN, A.; TRAVIS, L. E. Introduction to Simulation. In: WINTER SIMULATION CONFERENCE, Phoenix, Arizona, USA, 1991. *Proceedings*. Phoenix, Arizona, USA, 1991. p.5-14.
- TWARDOWSKI, K. An Associative Architecture for Genetic Algorithm-Based Machine Learning. *Computer*, p.27-38, nov. 1994.
- WHITLEY, D.; STARKWEATHER, T.; FURQUAY, D. Scheduling Problems and Traveling Salesmen: The Genetic Edge Recombination Operator. In: THIRD INTERNATIONAL CONFERENCE ON GENETIC ALGORITHMS, Los Altos, California, 1989. *Proceedings*. Los Altos, California, USA, 1989. p.133-140.
- WILLIAMS, T. Fuzzy, Neural and Genetic Methods Train to Overcome Complexity. *Computer Design*, p.59-76, mai. 1995.