EDUARDO JAQUES SPINOSA

ADAPTAÇÃO DINÂMICA DE PARÂMETROS EM COMPUTAÇÃO EVOLUCIONÁRIA: O CONTROLE DO TAMANHO DA POPULAÇÃO EM UM SISTEMA DE PROGRAMAÇÃO GENÉTICA

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre, pelo Curso de Pós-Graduação em Informática, do Setor de Ciências Exatas da Universidade Federal do Paraná.

Orientadora:
Prof.ª Dr.ª Aurora T. R. Pozo

CURITIBA
ABRIL 2002

Ministério da Educação
Universidade Federal do Paraná
Mestrado em Informática

PARECER

Nós, abaixo assinados, membros da Banca Examinadora da defesa de Dissertação de Mestrado em Informática do aluno *Eduardo Jaques Spinosa*, avaliamos o trabalho intitulado, *"Adaptação Dinâmica de Parâmetros em Computação Evolucionária: O Controle do Tamanho da População em um Sistema de Programação Genética"*, cuja defesa foi realizada no dia 06 de maio de 2002, às quatorze horas, no anfiteatro A do Setor de Ciências Exatas da Universidade Federal do Paraná. Após a avaliação, decidimos pela aprovação do candidato.

Curitiba, 06 de maio de 2002.

Profª. Drª. Aurora Trinidad Ramirez Pozo
DINF/UFPR - Orientadora

Prof. Dr. Guilherme Bittencourt
DAS/UFSC

Laura Sanchez Garcia
PGinf/UFPR

Para Iolanda e Wilson.

AGRADECIMENTOS

À professora orientadora, Drª. Aurora Trinidad Ramirez Pozo, por ter estado presente em todos os momentos.

Ao professor Dr. Guilherme Bittencourt, pela valiosa análise crítica e pelas idéias que enriqueceram este trabalho.

À professora Drª. Laura Sánchez Garcia, pela revisão criteriosa e pelo cuidado dispensado à minha dissertação.

A todos os que contribuíram para a realização deste trabalho.

SUMÁRIO

LISTA DE TABELAS ..... **vi**
LISTA DE FIGURAS. ..... **vii**
RESUMO. ..... **viii**
ABSTRACT. ..... **ix**
1. INTRODUÇÃO **1**
1.1. CONTEXTUALIZAÇÃO 1
1.2. MOTIVAÇÃO 2
1.3. OBJETIVOS E PASSOS METODOLÓGICOS 3
1.4. ORGANIZAÇÃO DO TEXTO 3
2. CONCEITOS BÁSICOS DE COMPUTAÇÃO EVOLUCIONÁRIA **5**
2.1. INTRODUÇÃO 5
2.2. VISÃO GERAL DO ALGORITMO DE AG/PG 6
2.3. FUNÇÃO DE *FITNESS* OU DE APTIDÃO 7
2.4. MÉTODOS DE SELEÇÃO 8
2.5. RECOMBINAÇÃO E OPERADORES GENÉTICOS 8
2.6. CRITÉRIO DE TÉRMINO 9
2.7. PARÂMETROS DO ALGORITMO 10
2.8. DIFERENÇAS ENTRE AG E PG 11
3. CONFIGURAÇÃO DE PARÂMETROS: CLASSIFICAÇÃO E PRINCIPAIS TRABALHOS **14**
3.1. CONTROLE ADAPTATIVO 16
3.1.1. Julstrom 1995 16
3.1.2. Barbosa 2000 19
3.1.3. Lobo 2000: "AG sem parâmetros" 21
3.2. CONTROLE AUTO-ADAPTATIVO 21
3.3. CONTROLE POR META-ALGORITMO 22
3.4. DISCUSSÃO DOS MÉTODOS 23
4. O MÉTODO "AG SEM PARÂMETROS" **25**
4.1. ESCOLHA DA TAXA DE SELEÇÃO E TAXA DE CRUZAMENTO 25
4.2. CONTROLE DO TAMANHO DA POPULAÇÃO 28
4.3. RESULTADOS OBTIDOS POR LOBO 30
5. IMPLEMENTAÇÃO E EXPERIMENTOS **32**
5.1. A FERRAMENTA *LIL-GP* 32
5.2. MÓDULO DE CONTROLE DE POPULAÇÕES 33
5.3. EXPERIMENTOS 35
5.3.1. O Problema da Trilha da Formiga 35
5.3.2. O Problema da Regressão Simbólica 38
5.4. METODOLOGIA E PARÂMETROS UTILIZADOS 39
6. RESULTADOS **42**
6.1. RESULTADOS SEM CONTROLE DO TAMANHO DA POPULAÇÃO 43
6.1.1. Como Interpretar os Gráficos 43
6.1.2. Análise dos Resultados sem Adaptação 51
6.1.3. A Inércia das Grandes Massas 52
6.2. RESULTADOS COM CONTROLE DO TAMANHO DA POPULAÇÃO 55
6.2.1. Como Interpretar os Gráficos 56
6.2.2. Análise dos Resultados com Adaptação 63
7. CONCLUSÕES **67**
7.1. PERSPECTIVAS E TRABALHOS FUTUROS 69
GLOSSÁRIO **70**
REFERÊNCIAS **71**

LISTA DE TABELAS

TABELA 1 - SEQÜÊNCIA NORMAL DE EXECUÇÃO DAS POPULAÇÕES 29
TABELA 2 - VALORES NORMAIS DA *FITNESS* MÉDIA DE CADA POPULAÇÃO 30
TABELA 3 - *OVERTAKING* DA POPULAÇÃO 2 SOBRE A 1 30
TABELA 4 - PARÂMETROS UTILIZADOS NOS EXPERIMENTOS 40
TABELA 5 - ALTERAÇÃO RELATIVA NO TAMANHO DO TORNEIO E NA TAXA DE CRUZAMENTO 65

LISTA DE FIGURAS

FIGURA 1 - ESTRUTURA DOS INDIVÍDUOS EM AG E PG 12
FIGURA 2 - TIPOS DE CONFIGURAÇÃO DE PARÂMETROS 15
FIGURA 3 - HISTÓRICO DE OPERADORES DE UM CROMOSSOMO 17
FIGURA 4 - RESULTADOS DE BARBOSA 20
FIGURA 5 - ADAPTAÇÃO DE PARÂMETROS USANDO UM META-ALGORITMO 23
FIGURA 6 - FUNCIONAMENTO DO MÓDULO DE CONTROLE 34
FIGURA 7 - EXEMPLO DE SOLUÇÃO PARA O PROBLEMA DA FORMIGA DE SANTA FÉ 37
FIGURA 8 - TRILHA PERCORRIDA PELO PROGRAMA DA FIGURA 7 37
FIGURA 9 - GRÁFICOS DO EXPERIMENTO DA TRILHA DA FORMIGA, SEM ADAPTAÇÃO, USANDO PARÂMETROS DO KOZA 2 45
FIGURA 10 - GRÁFICOS DO EXPERIMENTO DA TRILHA DA FORMIGA, SEM ADAPTAÇÃO, USANDO PARÂMETROS DO LOBO 46
FIGURA 11 - GRÁFICOS DO EXPERIMENTO DA REGRESSÃO DE ORDEM 3 (X^3), SEM ADAPTAÇÃO, USANDO PARÂMETROS DO KOZA 2 47
FIGURA 12 - GRÁFICOS DO EXPERIMENTO DA REGRESSÃO DE ORDEM 3 (X^3), SEM ADAPTAÇÃO, USANDO PARÂMETROS DO LOBO 48
FIGURA 13 - GRÁFICOS DO EXPERIMENTO DA REGRESSÃO DE ORDEM 4 (X^4), SEM ADAPTAÇÃO, USANDO PARÂMETROS DO KOZA 2 49
FIGURA 14 - GRÁFICOS DO EXPERIMENTO DA REGRESSÃO DE ORDEM 4 (X^4), SEM ADAPTAÇÃO, USANDO PARÂMETROS DO LOBO 50
FIGURA 15 - A INÉRCIA DAS GRANDES MASSAS - TRILHA DA FORMIGA, SEM ADAPTAÇÃO, USANDO PARÂMETROS DO KOZA 2 54
FIGURA 16 - GRÁFICOS DO EXPERIMENTO DA TRILHA DA FORMIGA, COM ADAPTAÇÃO, USANDO PARÂMETROS DO KOZA 2 57
FIGURA 17 - GRÁFICOS DO EXPERIMENTO DA TRILHA DA FORMIGA, COM ADAPTAÇÃO, USANDO PARÂMETROS DO LOBO 58
FIGURA 18 - GRÁFICOS DO EXPERIMENTO DA REGRESSÃO DE ORDEM 3 (X^3), COM ADAPTAÇÃO, USANDO PARÂMETROS DO KOZA 2 59
FIGURA 19 - GRÁFICOS DO EXPERIMENTO DA REGRESSÃO DE ORDEM 3 (X^3), COM ADAPTAÇÃO, USANDO PARÂMETROS DO LOBO 60
FIGURA 20 - GRÁFICOS DO EXPERIMENTO DA REGRESSÃO DE ORDEM 4 (X^4), COM ADAPTAÇÃO, USANDO PARÂMETROS DO KOZA 2 61
FIGURA 21 - GRÁFICOS DO EXPERIMENTO DA REGRESSÃO DE ORDEM 4 (X^4), COM ADAPTAÇÃO, USANDO PARÂMETROS DO LOBO 62

RESUMO

A Computação Evolucionária (CE) introduz um novo paradigma para resolver problemas em Inteligência Artificial, representando candidatos à solução como indivíduos e evoluindo-os com base na Teoria da Seleção Natural de Darwin. Algoritmos Genéticos (AG) e Programação Genética (PG), duas importantes técnicas de CE, têm sido aplicadas com sucesso tanto em cenários teóricos quanto em situações práticas.

Este trabalho discute o ajuste automático dos parâmetros que controlam o processo de busca neste tipo de algoritmo. Baseado em uma pesquisa recente, um método que controla o tamanho da população em AG é adaptado e implementado em PG.

Uma série de experimentos clássicos foi realizada, antes e depois das modificações, mostrando que este método pode aumentar a robustez e a confiabilidade no algoritmo. Os dados permitem uma discussão sobre o método e a importância da adaptação de parâmetros em algoritmos de CE.

Palavras-chave: Programação Genética, adaptação de parâmetros, tamanho da população, Computação Evolucionária, Algoritmos Genéticos.

ABSTRACT

Evolutionary Computation (EC) introduces a new paradigm for solving problems in Artificial Intelligence, representing solution candidates as individuals and evolving them based on Darwin's Theory of Natural Selection. Genetic Algorithms (GA) and Genetic Programming (GP), two important EC techniques, have been successfully applied both in theoretical scenarios and practical situations.
This work discusses the automatic adjustment of the parameters that control the search process on this type of algorithm. Based on a recent research, a method that controls the population size in a GA is adapted and implemented in GP.
A series of classic experiments has been performed before and after the modifications, showing that this method can improve the algorithms' robustness and reliability. The data allow a discussion about the method and the importance of the adaptation of parameters in EC algorithms.

Keywords: Genetic Programming, parameters adaptation, population size, Evolutionary Computation, Genetic Algorithms.

1. INTRODUÇÃO

1.1. CONTEXTUALIZAÇÃO

Dentro da Inteligência Artificial, uma das áreas que tem se destacado nos últimos anos tanto em trabalhos teóricos quanto em novas aplicações é a Computação Evolucionária (CE). Este novo paradigma, que recentemente tem despertado grande interesse por parte de pesquisadores em todo o mundo, engloba um conjunto de técnicas de busca baseadas nos conceitos da Teoria da Evolução das Espécies de Charles Darwin [Darwin 1859].

Nos algoritmos de CE, cada característica é representada sob a forma de um gene, que pode adquirir diferentes estruturas dependendo do tipo de problema e da técnica utilizada. O conjunto destas características que forma uma solução em potencial é chamado de "indivíduo". A busca pela solução do problema é feita por meio da recombinação genética entre indivíduos, com base na Seleção Natural de Darwin, até que uma solução válida seja encontrada; ou seja, até que um dos indivíduos atenda às condições pré-definidas pelo problema.

Dentro da CE, duas técnicas importantes se destacam: os Algoritmos Genéticos (AG) e a Programação Genética (PG). A diferença entre elas está, principalmente, na estrutura utilizada para a representação do indivíduo, como veremos no capítulo seguinte.

Vários trabalhos vêm demonstrando que estas técnicas podem ser aplicadas com sucesso em situações reais de busca em grandes espaços, tais como a mineração de dados (*Data Mining*), ou a descoberta de novas estruturas de proteínas, entre outras [Spector 2001] [IEEE 2002]. Da mesma forma, este novo tipo de algoritmo de busca também tem apresentado bons resultados em aplicações que envolvam a otimização de algum cenário, como no projeto automático de uma rede de distribuição de energia elétrica, ou em processos industriais em que o custo (tempo consumido, espaço físico

ocupado etc) precise ser minimizado.

Porém, como se trata de uma área relativamente recente, cujos avanços mais relevantes ocorreram na última década, muito há ainda que se pesquisar. Novos experimentos devem ajudar a definir com mais clareza as aplicações mais indicadas para este novo paradigma.

1.2. MOTIVAÇÃO

Apesar dos avanços proporcionados pelos recentes trabalhos no campo da CE, mais especialmente em PG [Spector 2001], o bom funcionamento destes sistemas ainda está fortemente relacionado a uma configuração adequada dos parâmetros que controlam o algoritmo central. Para cada novo problema proposto é preciso escolher os valores mais apropriados para cada parâmetro.

Experimentalmente verifica-se que os valores iniciais dos parâmetros afetam diretamente a eficiência do algoritmo e a utilização dos recursos da máquina (processamento e memória). Valores ótimos fazem com que o algoritmo encontre rapidamente uma solução, enquanto escolhas inadequadas podem até mesmo torná-lo incapaz de resolver o problema.

Para uma boa configuração destes parâmetros, o usuário deve possuir experiência prévia na solução de outros problemas semelhantes [Harik 1999]. Ele deve ser capaz de entender e prever a interferência de cada parâmetro no funcionamento do algoritmo, o que não é uma tarefa trivial. Muitas vezes pode ser preciso fazer diversas execuções do problema com diferentes conjuntos de parâmetros, até que se chegue a resultados satisfatórios. Este tipo de conhecimento não deveria fazer parte dos requisitos para um usuário do sistema [Lobo 2000], cuja preocupação principal é, simplesmente, aplicar o algoritmo para resolver um determinado problema.

Na grande maioria dos sistemas, os parâmetros escolhidos no início de uma execução permanecem constantes durante todo o processo, fazendo com que seu comportamento seja totalmente estático e limitado, não havendo a possibilidade de

adaptação do algoritmo às condições específicas de cada momento. Este fato tem motivado a discussão acerca dos métodos que modificam estes parâmetros ao longo da execução.

1.3. OBJETIVOS E PASSOS METODOLÓGICOS

O presente trabalho consiste no estudo e implementação dos métodos de adaptação ou controle automático de parâmetros em algoritmos evolutivos (de Computação Evolucionária), mais especialmente no algoritmo de Programação Genética. Este objetivo geral foi alcançado a partir de uma série de objetivos específicos seqüenciais.

Em uma primeira fase da pesquisa, o objetivo foi estudar os diferentes métodos utilizados para a configuração de parâmetros que vêm sendo apresentados nos últimos anos, destacando os trabalhos e resultados experimentais mais importantes dentro de cada categoria.

Em seguida, propusemos a escolha e o estudo aprofundado de um dos métodos mais recentes e de maior impacto do ponto de vista dos parâmetros escolhidos, dentro do escopo dos Algoritmos Genéticos [Lobo 2000]. A partir deste trabalho, o objetivo passou a ser o de implementar este método em Programação Genética, fazendo uso de uma ferramenta de domínio público, reconhecida e utilizada por pesquisadores em todo o mundo [Banzhaf 1998].

Finalmente, foram executadas várias baterias de testes de alguns problemas clássicos freqüentemente utilizados para este fim, com o objetivo de analisar o impacto desta metodologia, comparando-a com o algoritmo tradicional de PG. Os resultados experimentais são discutidos, procurando contribuir com o avanço do conhecimento nesta área.

1.4. ORGANIZAÇÃO DO TEXTO

O capítulo seguinte faz a revisão de alguns conceitos de Algoritmos

Genéticos e Programação Genética necessários para o entendimento do trabalho. Em seguida, no capítulo 3, procura-se classificar as principais técnicas de configuração de parâmetros em AG/PG, destacando os trabalhos mais importantes dentro de cada uma. Um desses métodos é então escolhido e explicado mais detalhadamente no capítulo 4. O capítulo 5 descreve como esta técnica foi adaptada e implementada em um sistema de Programação Genética, incluindo os experimentos realizados e a metodologia utilizada nos testes. Em seguida, no capítulo 6, os resultados experimentais são analisados com o auxílio de gráficos. E finalmente, o último capítulo resume as conclusões desta pesquisa, propondo temas e trabalhos para o futuro.

2. CONCEITOS BÁSICOS DE COMPUTAÇÃO EVOLUCIONÁRIA

2.1. INTRODUÇÃO

A Computação Evolucionária (CE) engloba um conjunto de técnicas que fazem uso de um novo paradigma para a solução de problemas, fundamentado na Teoria da Evolução das Espécies de Charles Darwin [Darwin 1859].

"...Se variações úteis para qualquer organismo devam ocorrer para que ele venha a existir, certamente indivíduos assim caracterizados terão a melhor chance de serem preservados na luta por sobrevivência; e do forte princípio de hereditariedade, eles tenderão a produzir gerações com características similares. Este princípio de preservação, eu batizei, para ser sucinto, de Seleção Natural." [Darwin 1859]

Segundo a teoria de Darwin, na natureza sobrevivem os seres que têm a melhor capacidade de se adaptarem às mudanças que ocorrem no meio ambiente. Esses indivíduos passarão suas características genéticas para seus descendentes e, ao final de muitas gerações, teremos uma população de indivíduos selecionados naturalmente.

Inspirado nestes conceitos incorporados da Biologia, John Holland [Holland 1975] propôs, em 1960, uma versão computacional deste algoritmo de evolução natural. Seu objetivo inicial era estudar os fenômenos relacionados à adaptação das espécies e da seleção natural que ocorre na natureza, bem como desenvolver uma maneira de incorporar estes conceitos aos computadores [Mitchell 1996]. A técnica desenvolvida por Holland vem sendo amplamente discutida e aplicada ao longo dos anos e ficou conhecida pelo nome de Algoritmos Genéticos (AG).

Em 1992, John Koza aplicou AG para evoluir programas de computador e deu origem a uma nova e muito importante área de pesquisa dentro de Computação Evolucionária. A Programação Genética (PG) [Koza 1992] é uma técnica de geração automática de programas de computador, que propõe a criação e manipulação de

software aplicando conceitos herdados da Biologia.

De modo geral, pode-se encarar a Programação Genética como uma técnica de busca, que atua sobre o espaço de todos os programas possíveis que podem ser gerados em uma determinada linguagem, e cujo objetivo final é encontrar um programa que resolva um dado problema.

2.2. VISÃO GERAL DO ALGORITMO DE AG/PG

As técnicas de AG e PG fazem uso de conceitos como a Seleção Natural no processo de busca da solução para um determinado problema. Cada solução em potencial tem suas características mapeadas sob a forma de genes e é vista como um indivíduo. Os indivíduos são agrupados para formar populações.

Populações são, neste sentido, conjuntos de pontos de exploração do espaço de busca. Populações pequenas demais podem não ser capazes de prover a diversidade necessária para a solução de um problema, enquanto populações grandes exigem um alto custo de processamento e memória. Este é, portanto, um parâmetro crítico e que tem forte influência sobre a taxa de acerto do algoritmo. Ele é o foco central desta pesquisa e esta discussão será retomada ao longo do trabalho.

Partindo de uma população inicial, o algoritmo faz constantes recombinações entre os indivíduos, semelhantes às que ocorrem na natureza e, geração após geração, seleciona e mantém ativos os melhores indivíduos, isto é, aqueles que mais se aproximam da solução.

O objetivo final é chegar a uma solução para o problema em questão por meio da recombinação das características dos indivíduos (genes); ou seja, encontrar um indivíduo que resolve o problema proposto. Assim, a CE pode ser vista como uma técnica de busca, aplicada sobre o universo de todas as possíveis soluções.

Como não há informação inicial sobre a forma da solução, o algoritmo parte de uma população inicial de indivíduos gerados aleatoriamente, procurando reproduzir a diversidade necessária para que seja possível se compor a solução.

Após a geração da população inicial, o algoritmo entra em um laço que é executado até que um dos critérios de término seja atingido. Este laço consiste de duas ações principais: [Koza 1992]

a) Seleção - avalia cada programa por meio de uma função heurística especial (função de *fitness* ou função de aptidão, a ser apresentada posteriormente), que expressa quão próximo cada programa está da solução ideal.

b) Procriação - cria uma nova população selecionando os indivíduos de acordo com o valor da *fitness* e aplicando os operadores genéticos: reprodução, cruzamento e mutação.

Cada execução deste laço representa uma nova geração. Como na natureza, o processo evolutivo teoricamente não teria fim, o que significa que uma solução perfeita nunca seria alcançada. Na versão computacional deste algoritmo, critérios de término são estabelecidos para evitar esta repetição infinita. As duas condições normalmente utilizadas são: a descoberta de uma solução válida e a execução de um número máximo de gerações previamente definido.

2.3. FUNÇÃO DE *FITNESS* OU DE APTIDÃO

Na natureza os seres vivos são selecionados com base no seu grau de adaptabilidade ao meio ambiente. Em CE este conceito é expresso pela função de *fitness*. Os programas que mais se aproximarem da solução esperada receberão melhores valores de *fitness* e, conseqüentemente, terão maior chance de serem selecionados.

A escolha da função de *fitness*, assim como a escolha do método de avaliação utilizado por ela, está intrinsecamente ligada à natureza do problema [Banzhaf 1998]. Boas escolhas são essenciais para se obterem bons resultados, já que a função de *fitness* é a força-guia que direciona o algoritmo de CE [Gritz 1999].

2.4. MÉTODOS DE SELEÇÃO

Dada uma população em que a cada indivíduo foi atribuído um valor de *fitness*, existem vários métodos para selecionar os programas sobre os quais serão aplicados os operadores genéticos. Os mais comuns são:

- Método da Roleta ou Proporcional à *Fitness* - Cada indivíduo recebe uma fatia do todo, proporcional ao seu valor de *fitness*. Um número aleatório entre zero e o valor do todo (soma da *fitness* de todos os indivíduos) determina o indivíduo que será escolhido. Quanto maior for o valor da *fitness*, maior será a possibilidade de ele ser selecionado.
- Método do Torneio - N indivíduos da população são escolhidos aleatoriamente para formar uma sub-população temporária. Deste grupo, o programa selecionado será aquele que possuir o maior valor de *fitness*. Este método é o mais utilizado pois oferece a vantagem de não exigir uma comparação entre todos os indivíduos da população [Banzhaf 1998].

2.5. RECOMBINAÇÃO E OPERADORES GENÉTICOS

Uma vez que os indivíduos tenham sido selecionados, a recombinação de suas características para a geração de um descendente ocorre por meio da aplicação de um operador genético. A escolha entre os operadores é aleatória mas cada operador tem uma probabilidade diferente de ser selecionado. Estes valores recebem o nome de "taxas" e a soma das taxas de todos os operadores vale 100 %.

Os três operadores mais freqüentemente utilizados são:

- Cruzamento - Dois indivíduos-pais são recombinados para gerar dois filhos potencialmente diferentes. Um ponto aleatório de cruzamento é escolhido em cada indivíduo-pai e os trechos são intercambiados. É equivalente à reprodução sexuada dos seres vivos.
- Mutação - Um novo conjunto de genes gerado aleatoriamente é

acrescentado, ou substitui uma parte do indivíduo. Teoricamente, esta operação ajuda a manter a diversidade na população, que é importante para evitar a convergência precoce do algoritmo para uma única solução. Entretanto, existem evidências de que a mutação pode ser praticamente ignorada [Koza 1992].

- Reprodução - Um indivíduo é copiado para a próxima geração sem sofrer nenhuma mudança em sua estrutura. Equivale à reprodução assexuada dos seres vivos. Em algumas situações a reprodução não é vista como um operador, já que nenhuma alteração é feita. Desta forma, é possível considerar que aqueles indivíduos que não foram selecionados para cruzamento nem mutação serão copiados; ou seja, reproduzidos sem alteração na próxima geração.

Outros operadores podem ser definidos dependendo do problema em questão, tais como a inversão (permutação), a edição e o encapsulamento [Koza 1992].

2.6. CRITÉRIO DE TÉRMINO

O critério de término é responsável por interromper o laço de repetição do processo evolutivo que, idealmente, não teria fim. Normalmente são estabelecidos casos de treinamento (*fitness cases*), por meio da determinação de valores de entrada e respectivas saídas esperadas. Desta forma, um possível critério de término seria a satisfação de um certo número de *fitness cases*. É importante estabelecer uma margem de erro (*threshold*) para fazer a comparação entre o valor de saída esperado e o que foi obtido pelo programa. Como pode ser muito custoso ou mesmo impossível chegar a uma solução "perfeita" em tempo finito, aceitam-se resultados que estejam dentro da margem de erro.

Outro critério de término é o número máximo de gerações, usado para evitar que a repetição continue infinitamente caso nenhuma solução seja encontrada.

2.7. PARÂMETROS DO ALGORITMO

Os parâmetros controlam o funcionamento do algoritmo de AG/PG, permitindo ajustes no processo evolutivo. No entanto, a função mais importante dos parâmetros é a de estabelecer limites, uma vez que os recursos da máquina (tempo e memória) são limitados.

Os principais parâmetros utilizados por um algoritmo de PG são:

- Tamanho da população – É o número total de programas em cada população.
- Tamanho do torneio – É o número de programas selecionados aleatoriamente para constituir a sub-população que é utilizada pelo método de seleção de Torneio (descrito no item 2.4).
- Taxa de cruzamento – Representa a probabilidade de se escolher o operador de cruzamento.
- Taxa de mutação - É a probabilidade de se escolher o operador de mutação.
- Taxa de reprodução - Indica a probabilidade de que o indivíduo seja copiado sem alteração. O valor da taxa de reprodução é igual a 100 % menos a soma das taxas dos demais operadores.
- Número máximo de gerações - Se tal valor for atingido sem que tenha sido encontrada uma solução válida, o processo é interrompido para evitar a repetição infinita.
- Profundidade mínima do indivíduo (PG) - É o menor tamanho de um indivíduo, usado pelo método de inicialização.
- Profundidade máxima do indivíduo (PG) - Utilizado durante a criação da população inicial e após operações que possam alterar o tamanho do indivíduo, como é o caso do operador de cruzamento.
- Limiar ou margem de erro - É o valor da precisão que é considerada para comparar os resultados de um indivíduo com os resultados

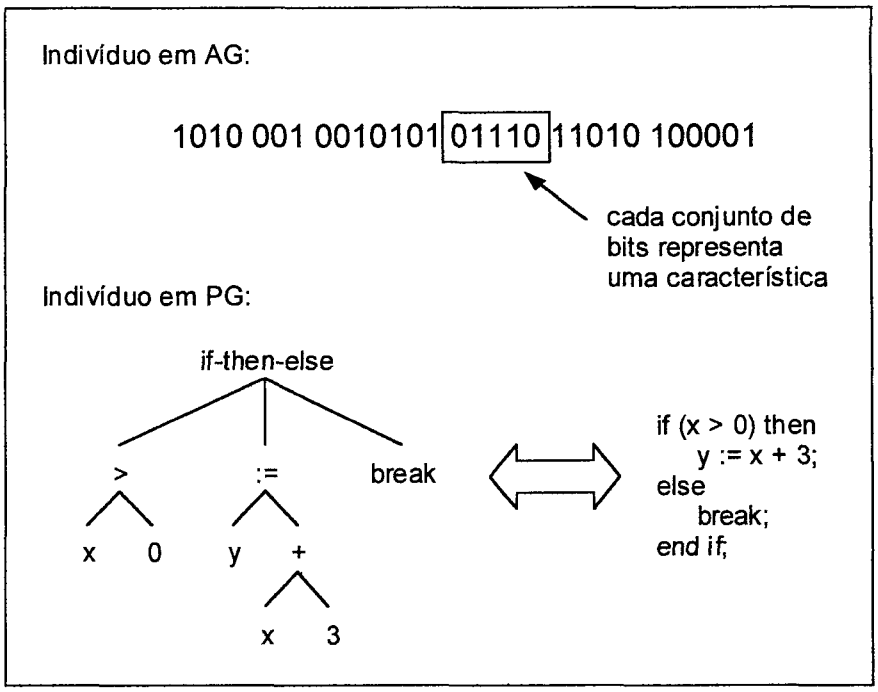
desejados. Se a distância for menor do que esta margem de erro, considera-se que houve um acerto.

- Método de inicialização (PG) - Especifica de que forma é criada a primeira geração, para que ela tenha uma boa diversidade no que diz respeito ao tamanho dos indivíduos. Os métodos mais utilizados são o *full*, o *grow*, e o *ramped-half-and-half*.

2.8. DIFERENÇAS ENTRE AG E PG

Algoritmos Genéticos e Programação Genética compartilham a mesma base teórica. Ambas as técnicas mantêm uma população de indivíduos que representam soluções em potencial. Nas duas, os indivíduos são selecionados e recombinados geneticamente por meio da ação de operadores genéticos ao longo de gerações. Entretanto, a diferença essencial entre AG e PG está na estrutura dos indivíduos. Enquanto em AG eles são representados por uma seqüência de *bits* ou números, em PG os indivíduos são programas de computador, usualmente representados por árvores, como mostra a figura 1.

FIGURA 1 - ESTRUTURA DOS INDIVÍDUOS EM AG E PG



Cada nó da árvore pode ser uma função ou um terminal. Funções não têm valor próprio; elas recebem parâmetros e são calculadas com base neles. Uma função pode ser um comando ou um operador lógico ou aritmético, por exemplo. Já os terminais têm um valor definido, como uma variável ou uma constante. É por este motivo que nos nós-folha só é possível encontrar terminais e, conseqüentemente, a avaliação de uma árvore deve ocorrer das folhas para a raiz.

O comprimento dos indivíduos em AG é normalmente fixo, enquanto em PG os tamanhos das árvores podem variar de acordo com a complexidade de cada programa. Este aspecto acrescenta uma certa dificuldade em todos os processos do algoritmo, desde a seleção até a recombinação, sempre que a estrutura do indivíduo tiver que ser lida ou alterada.

Além disso, o tamanho das árvores é normalmente limitado para evitar que se ultrapasse o limite da capacidade de armazenamento, e também para que os indivíduos não cresçam descontroladamente, sem que haja necessariamente uma melhoria qualitativa. Isto faz com que as operações de cruzamento, por exemplo,

tornem-se especialmente complicadas, pois é preciso verificar se os filhos gerados após o processo poderão realmente ser armazenados.

Finalmente, como em PG as estruturas manipuladas são programas de computador, passíveis de serem interpretados ou compilados/executados, é preciso manter a consistência sintática e semântica sempre que qualquer tipo de manipulação for feito nessas estruturas. Diferentes soluções têm sido apresentadas para realizar este controle, incluindo o uso de gramáticas. Porém, uma exploração mais detalhada do tema foge ao escopo deste trabalho.

3. CONFIGURAÇÃO DE PARÂMETROS: CLASSIFICAÇÃO E PRINCIPAIS TRABALHOS

Na solução de um problema por meio de uma das técnicas de Computação Evolucionária, os parâmetros que controlam o algoritmo determinam, entre outras coisas, a probabilidade de que uma solução para o problema seja encontrada. A escolha desses valores pode, em certos casos, dificultar excessivamente a busca ou mesmo impedir o algoritmo de trabalhar corretamente. Da mesma forma, uma configuração adequada pode aumentar as taxas de acerto, aumentando assim a confiabilidade no algoritmo. Em algumas situações, uma escolha correta poderá poupar tempo e espaço de armazenamento, reduzindo os custos de processamento e/ou memória, o que é extremamente importante no caso de sistemas de Computação Evolucionária.

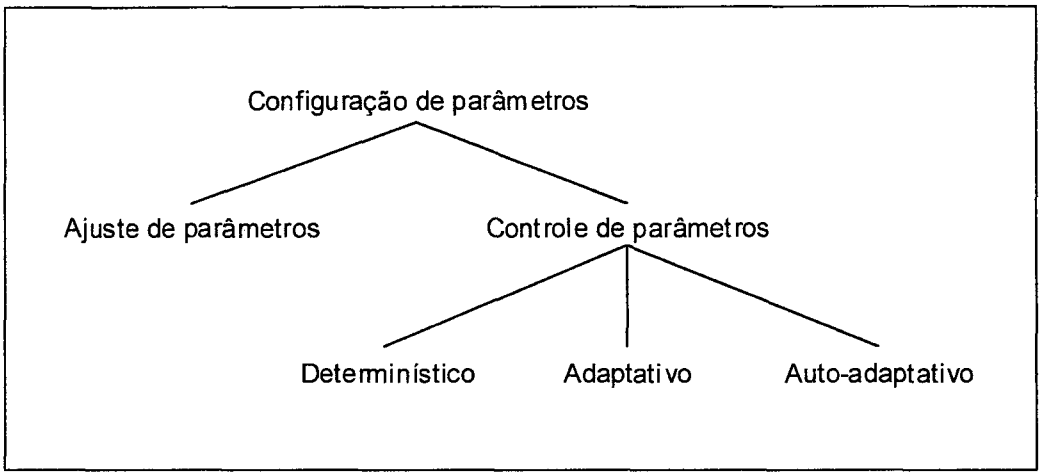
Ainda hoje, em grande parte das aplicações de AG e PG, os valores dos parâmetros são escolhidos no início do processo, permanecendo constantes durante toda a busca. Por outro lado, várias experiências têm sido feitas, principalmente na área de Algoritmos Genéticos (AG), no sentido de adaptar automaticamente os valores dos principais parâmetros que controlam o funcionamento do algoritmo. Na pesquisa realizada para este trabalho, constatou-se que a maioria dos estudos se restringe a adaptar alguns parâmetros, uma vez que seria bastante complexo tentar fazê-lo com todos. Desta forma, pretende-se isolar o efeito de cada um para permitir uma análise pontual de sua influência.

Ao longo deste capítulo vamos analisar os estudos mais relevantes nesta área, procurando classificá-los de acordo com o tipo de abordagem utilizada. O objetivo não é relatar todos os estudos dentro de cada abordagem, mas sim apresentar aqueles cuja contribuição tenha se mostrado importante no aprimoramento de uma das técnicas de adaptação de parâmetros.

Uma recente classificação das diferentes estratégias utilizadas para configurar os parâmetros em algoritmos evolutivos foi apresentada por Eiben,

Hinterding e Michalewicz [Eiben 2000], e divide as maneiras de se escolherem os valores dos parâmetros como mostrado na figura a seguir.

FIGURA 2 - TIPOS DE CONFIGURAÇÃO DE PARÂMETROS



O chamado "ajuste de parâmetros" é o método tradicional de fixar valores iniciais, que permanecerão imutáveis durante a execução (*run*).

De um outro lado, apresentam-se as técnicas de "controle de parâmetros", onde os valores dos parâmetros genéticos são modificados durante a execução. Essas técnicas podem ser divididas em 3 abordagens principais:

a) Controle determinístico - Nele, a regra responsável por alterar os parâmetros é fixa, atuando sempre da mesma maneira, independentemente das condições específicas de cada execução.

b) Controle adaptativo - Os parâmetros são modificados com base em alguma informação dinâmica obtida do algoritmo. Dados estatísticos de eficiência e qualidade são as informações mais freqüentemente utilizadas para calcular os valores dos parâmetros.

c) Controle auto-adaptativo – Aqui, a idéia é submeter os próprios parâmetros ao algoritmo evolutivo, embutindo estes valores nos cromossomos dos indivíduos. Neste caso os valores dos parâmetros passam a estar sujeitos à evolução por meio da seleção natural, sofrendo a ação dos operadores genéticos de cruzamento, mutação etc.

Lobo, em sua tese de Doutorado [Lobo 2000], faz uma classificação semelhante, porém inclui um quarto tipo de abordagem bastante importante:

d) Meta-algoritmo - Neste caso, um algoritmo externo é responsável pelo controle global do processo. Ele cria várias instâncias de evolução, cada uma com um conjunto diferente de parâmetros. Depois disso, compara os resultados obtidos para definir quais combinações de parâmetros são mais adequadas ao problema em questão.

A seguir, serão apresentados alguns trabalhos de pesquisa realizados nos últimos anos dentro das 3 principais abordagens para controle dos parâmetros:

- Controle Adaptativo
- Controle Auto-adaptativo
- Controle por Meta-algoritmo

A análise que se segue tem o objetivo de fornecer um panorama geral do estado da arte segundo a classificação recém apresentada, mostrando como estão sendo traçados os rumos da pesquisa nesta área.

3.1. CONTROLE ADAPTATIVO

Neste tipo de técnica, os valores de alguns parâmetros são modificados durante a execução do algoritmo, de acordo com uma regra que leva em consideração o histórico do processo evolutivo.

A maioria dos trabalhos que utilizam um procedimento deste tipo o fazem para escolher automaticamente as taxas dos operadores genéticos a cada geração. Alguns estudos recentes são apresentados a seguir.

3.1.1. Julstrom 1995

Neste conjunto de trabalhos [Julstrom 1995] [Julstrom 1996] [Julstrom 1997], o autor descreve uma técnica à qual deu o nome de ADOPP (*Adaptative Operator Probabilities*), que evoluiu a partir de outros trabalhos semelhantes [Davis

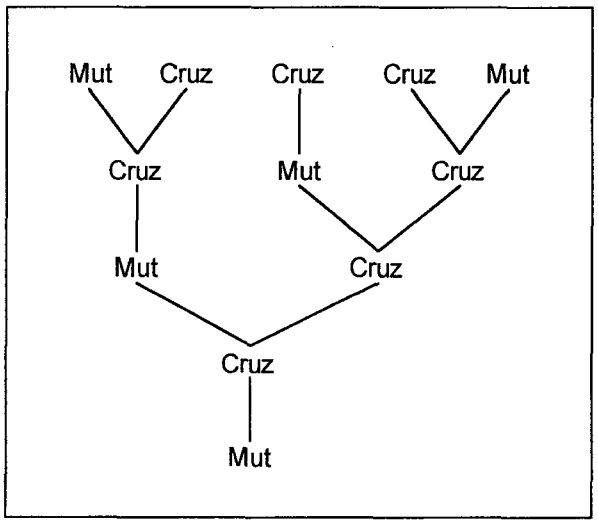
1989].

A proposta de Julstrom pode ser aplicada diretamente em um AG onde estejam presentes apenas 2 operadores: cruzamento e mutação. O controle acontece justamente sobre as taxas destes operadores, e está fundamentado nos conceitos de produtividade e recompensa. Um operador mais produtivo deve ter sua taxa aumentada, enquanto operadores improdutivos têm uma participação cada vez menor na geração de novos indivíduos.

Os valores das taxas de cruzamento e mutação são atualizados de acordo com a recompensa que eles recebem. A recompensa de um operador é calculada em função da qualidade dos indivíduos que sofreram a ação deste operador. Um indivíduo é considerado "melhorado" (*improved*) se ele é melhor do que a média da população. Neste caso, o operador que o originou recebe crédito imediato igual a 1. Caso contrário, o crédito imediato vale 0 (zero).

Além disso, cada indivíduo guarda um histórico cuja estrutura é uma árvore binária invertida, que indica quais operadores foram aplicados para a geração de cada um de seus ancestrais (pais, avós etc), como mostra a figura 3. A profundidade do histórico (número de níveis da árvore) é um parâmetro que pode ser modificado.

FIGURA 3 - HISTÓRICO DE OPERADORES DE UM CROMOSSOMO



O operador que gerou o indivíduo é o que aparece na raiz da árvore. O nível

dos pais é o acima deste. Na árvore da figura 3, por exemplo, o indivíduo atual foi gerado a partir de um único pai utilizando o operador de mutação. O pai foi gerado a partir de 2 indivíduos por meio de cruzamento, e assim por diante.

Além do crédito imediato, cada operador do histórico também recebe um crédito atenuado por um fator de decremento (*decay*) entre 0 e 1. A cada nível o fator de decremento é multiplicado mais uma vez. Assim, o nível dos pais recebe um crédito igual ao crédito imediato, vezes o fator de decremento. O nível dos avós recebe um crédito igual ao crédito imediato, vezes o fator de decremento ao quadrado. No caso do indivíduo cujo histórico está representado na figura 3, supondo um decremento de 0,8 , se o indivíduo fosse considerado "melhorado" (*improved*) o operador de cruzamento receberia crédito total de $0,8 + (0,8)^2 + 2(0,8)^3 + 3(0,8)^4 = 3,69$ e o de mutação $1 + (0,8)^2 + (0,8)^3 + 2(0,8)^4 = 2,97$.

Além disso, a técnica ADOPP de Julstrom prevê que os valores dos créditos não são aplicados instantaneamente sobre os operadores. Eles entram em uma fila de comprimento *Qlen* determinado, um valor que pode ser modificado. Após o processamento de cada indivíduo, são colocados na fila o nome do operador imediato (cruzamento ou mutação) e os valores dos créditos de cruzamento (*Cc*) e mutação (*Cm*). Até que a fila esteja cheia, as taxas de cruzamento e mutação permanecem fixas. Depois disso, a cada novo elemento que entra, o mais antigo é desempilhado para que seja feito o ajuste nas taxas de cruzamento (*Pc*) e mutação (*Pm*), de acordo com as equações:

$$Pc = \frac{Cc / Nc}{Cc / Nc + Cm / Nm}$$

$$Pm = 1 - Pc$$

Nc e *Nm* são as quantidades de vezes que os nomes dos operadores de cruzamento e mutação, respectivamente, aparecem na fila como sendo os operadores imediatos para a obtenção dos indivíduos. A função da fila é manter uma janela de

observação. Se a fila tiver comprimento 5, por exemplo, as taxas de cruzamento e mutação do indivíduo número 26 serão ajustadas em função dos créditos calculados com base nos indivíduos 21 a 25, que fazem parte da fila naquele momento.

Uma das conclusões desta série de trabalhos de Julstrom é que talvez este tipo de adaptação de parâmetros possa ser aplicado com mais propriedade para descobrir os valores ideais das taxas de cruzamento e mutação inicialmente, em uma primeira execução. Depois disso, os valores ideais calculados permaneceriam constantes pelo resto das execuções do algoritmo.

Vale observar ainda que, para permitir a adaptação de 2 parâmetros (taxa de cruzamento e taxa de mutação), são criados 3 novos parâmetros (profundidade da árvore de histórico, fator de decremento e tamanho da fila).

3.1.2. Barbosa 2000

Este estudo [Barbosa 2000] utiliza uma metodologia semelhante à de produtividade e recompensa [Julstrom 1995] aplicada a Algoritmos Genéticos. Para o cálculo da produtividade são utilizadas medidas locais, que comparam a *fitness* do filho com as de seus pais [Lobo 1996], e medidas globais, que fazem a comparação com a *fitness* média da população.

Em qualquer das opções, é comum dividir-se o valor da recompensa pelo custo computacional; isto é, o processamento necessário para o operador genético em questão (reprodução, cruzamento, mutação etc). Desta forma, o valor da recompensa também será inversamente proporcional ao número de funções que tiveram que ser calculadas (custo computacional de processamento), como mostra a fórmula a seguir, onde *ri* é a recompensa e *ni* é o custo (número de funções calculadas).

$$r_i \leftarrow \frac{r_i}{n_i}$$

Para calcular a recompensa acumulada ao longo de várias gerações, Barbosa

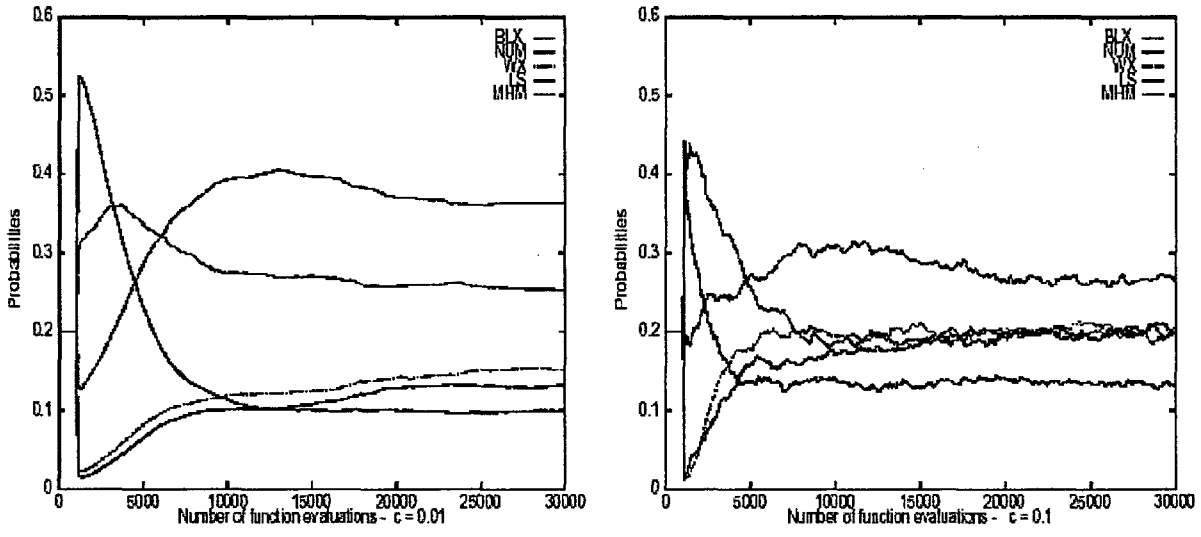
e Sá [Barbosa 2000] adotam a proposta de Lobo e Goldberg [Lobo 1996], utilizando a seguinte expressão:

$$R_i \leftarrow (1-c).R_i + c.r_i$$

Ri é a recompensa acumulada e *ri* é a recompensa instantânea ou atual. O fator *c* é um número positivo utilizado apenas para controlar o uso da memória. Quanto maior for o valor de *c*, mais importância é dada ao valor atual com relação aos valores passados de recompensa. Conseqüentemente, menos memória é consumida.

Na figura 4 é possível observar o impacto do fator *c*. No gráfico à esquerda, $c = 0.01$; isto é, o valor atual da recompensa tem uma participação de 1% contra 99% das recompensas das gerações anteriores. No gráfico à direita, $c = 10\%$. Percebe-se que quanto menor for o valor de *c* mais lenta e gradual será a adaptação do parâmetro.

FIGURA 4 - RESULTADOS DE BARBOSA



Finalmente, a taxa de cada operador (ajustada automaticamente) é calculada como sendo uma proporção da recompensa acumulada de cada operador em relação ao total das recompensas acumuladas de todos os operadores:

$$p_i \leftarrow \frac{R_i}{\sum_1^n R_i}$$

O trabalho analisa resultados obtidos em experimentos, comparando as opções de cálculo de produtividade descritas anteriormente. Os resultados mostram que a adaptação de parâmetros deixou o algoritmo mais robusto, em detrimento de uma pequena perda de qualidade na solução. A adaptação neste caso mostrou-se útil na determinação do conjunto de operadores que será mais eficiente para a solução de um problema qualquer.

3.1.3. Lobo 2000: "AG sem parâmetros"

Um dos trabalhos mais recentes, no qual boa parte da nossa pesquisa está fundamentada, propõe um Algoritmo Genético sem parâmetros [Harik 1999] [Lobo 2000].

Em sua tese de Doutorado, Lobo mostra que é possível fixar os valores de alguns parâmetros, como a taxa de cruzamento e o tamanho do torneio, garantindo que o algoritmo caminhe na direção da uma solução.

Já para otimizar o tamanho da população, Lobo propõe o uso de várias populações, cada uma com o dobro do número de indivíduos que a anterior. As populações se alternam durante as execuções e competem entre si. Eventualmente, uma população com melhores resultados pode eliminar uma outra. O objetivo principal é resolver o problema com o menor número possível de indivíduos.

Este trabalho será analisado mais profundamente no próximo capítulo.

3.2. CONTROLE AUTO-ADAPTATIVO

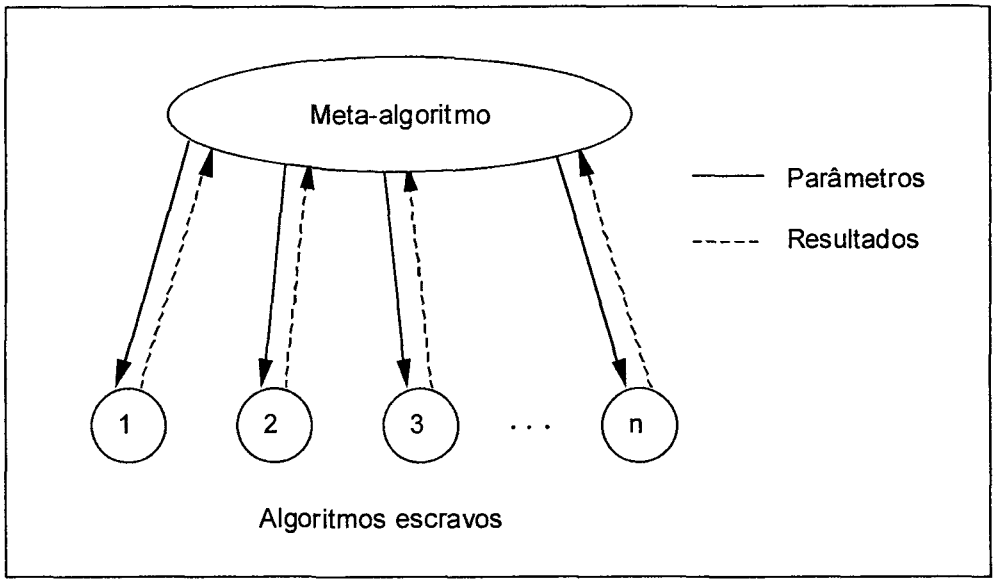
Ao contrário do que se pode imaginar, a idéia de que os parâmetros de um algoritmo também podem ser submetidos ao processo evolutivo foi introduzida em 1967 por Bagley [Bagley 1967]. Este tipo de técnica de adaptação tem sido mais

comumente aplicado nas chamadas Estratégias de Evolução (ES: *Evolution Strategies*), uma variação dos Algoritmos Genéticos na qual o operador de mutação adquire uma importância maior no processo, superando os demais operadores. Com o crescimento dos AGs, alguns autores acreditam que o controle auto-adaptativo pode também ser aplicado em AGs [Bäck 1992] [Smith 1996].

3.3. CONTROLE POR META-ALGORITMO

Esta técnica introduz um algoritmo genético externo de controle, responsável pela escolha dos melhores parâmetros para os AG escravos. Esta idéia surgiu com Weinberg [Weinberg 1970], e teve a primeira implementação com Mercer [Mercer 1978]. Para o algoritmo mestre, cada indivíduo da população é uma *string* que representa uma possível configuração de parâmetros. Para avaliar cada indivíduo, o meta-algoritmo executa um algoritmo escravo com aquele conjunto de parâmetros, e verifica a qualidade dos resultados. Ao término da execução, o indivíduo vencedor representa a configuração de parâmetros mais adequada.

FIGURA 5 - ADAPTAÇÃO DE PARÂMETROS USANDO UM META-ALGORITMO



Koch [Koch 2000] apresenta um exemplo prático de aplicação do meta-algoritmo para a solução de um problema na indústria: o posicionamento de ventosas de fixação em uma peça de madeira a ser trabalhada por robôs. Neste caso, um meta-algoritmo é responsável por controlar vários algoritmos de otimização, cada um com diferentes valores de parâmetros. A partir da avaliação da *fitness* do melhor indivíduo produzido por cada AG escravo, o meta-algoritmo mestre decide qual é o melhor conjunto de valores para os parâmetros.

Outra abordagem possível [Greffenstette 1986] considera 6 parâmetros principais, dando origem a 218 possíveis configurações.

3.4. DISCUSSÃO DOS MÉTODOS

São várias as possibilidades quando se pretende adaptar parâmetros em um algoritmo. Algumas o fazem por meio de testes para determinar o conjunto ótimo de valores, outras fazem a alteração dinâmica desses valores com base em dados estatísticos das últimas *n* gerações. Alguns trabalhos procuram embasamento teórico para fixar parâmetros padrão, outros acreditam que eles devem ser escolhidos de acordo com cada problema.

Cada abordagem tem suas virtudes e também suas desvantagens. O método do meta-algoritmo, por exemplo, procura encontrar os melhores valores em de uma fase inicial de "evolução de parâmetros", comandada por um algoritmo mestre. Se por um lado perde-se tempo para isto, depois da determinação dos parâmetros o processo é bem mais rápido.

A proposta de Julstrom, semelhante ao trabalho de Barbosa, acompanha a filosofia evolutiva de que os valores dos parâmetros podem ser alterados durante o processo, o que parece uma maneira bastante correta de realizar este ajuste. Em contrapartida, é preciso não esquecer que para manter esta estrutura funcionando são necessários outros parâmetros, que controlam a fila, o histórico etc.

Uma idéia especialmente interessante é aquela proposta por Lobo, que trabalha com um dos parâmetros mais importantes: o tamanho da população, sem acrescentar nenhum parâmetro novo. A definição do tamanho da população é extremamente importante pois este determina a dimensão do espaço de busca. Se for muito grande, o custo computacional será maior do que o necessário. Se ele for muito pequeno, pode até mesmo inviabilizar o processo, uma vez que não será possível encontrar na população a diversidade genética necessária para a composição da solução.

Por estes motivos, decidiu-se implementar o método "AG sem parâmetros" proposto por Lobo [Lobo 2000] em um sistema de Programação Genética. O capítulo a seguir apresenta, detalhadamente, este método.

4. O MÉTODO "AG SEM PARÂMETROS"

Este capítulo descreve o funcionamento do método de controle de parâmetros para AG proposto por Lobo [Lobo 2000], cujo principal objetivo é eliminar a necessidade de configuração de parâmetros em Algoritmos Genéticos.

A maior motivação que levou Lobo a propor este método é que muitas vezes o usuário de um sistema de AG não é um especialista na área, e por isso pode não ter conhecimento do efeito dos parâmetros para poder fazer uma boa escolha. Soma-se a isto o fato de que os valores ótimos para os parâmetros podem variar dependendo do tipo de problema que se está resolvendo.

Assim, Lobo estabelece a diferença entre os tipos de codificação, os tipos de operadores e o que são, realmente, os parâmetros genéticos. A partir deste ponto a pesquisa passa a concentrar-se nos 3 parâmetros que, segundo o autor, afetam em maior nível a eficiência do algoritmo. São eles a taxa de seleção (*s*), a taxa de cruzamento (*pc*) e o tamanho da população.

4.1. ESCOLHA DA TAXA DE SELEÇÃO E TAXA DE CRUZAMENTO

Para definir os valores da taxa de seleção (*s*) e da taxa de cruzamento (*pc*) é utilizado o Teorema do Esquema, que justifica o bom funcionamento dos Algoritmos Genéticos [Goldberg 1989] [Mitchell 1996].

Esquemas (*H*) são modelos de indivíduos que representam um conjunto de soluções possíveis. Um esquema pode conter 0, 1 ou * (indicando qualquer valor). A ordem de um esquema *o(H)* é o número de posições definidas (1 ou 0). Para calcular o comprimento de um esquema *d(H)* faz-se a subtração entre os números da última e da primeira posição do esquema que tem valor definido. Para o esquema 011**0*, o comprimento $d = 6 - 1 = 5$. A função *m(H,t)* representa a quantidade de indivíduos que representam um esquema *H* em um tempo *t*. Considerando uma população de tamanho *n*, a probabilidade de se selecionar o esquema *H* é igual à relação entre a *fitness* do

esquema *f(H)* e a soma de todos os valores de *fitness* da população. Portanto, em um tempo $t + 1$ teremos a seguinte quantidade de representantes do esquema H:

$$m(H,t+1) = m(H,t).n.\frac{f(H)}{\sum f} = m(H,t).\frac{f(H)}{\bar{f}}$$

Assim, esquemas que possuem *fitness* acima da média da população crescerão em número de representantes.

Para analisar o efeito do operador de cruzamento é preciso considerar em que ponto do cromossomo ele ocorre, uma vez que esta operação tende a separar características de um mesmo indivíduo. Por exemplo, o esquema *1***0 tem maior chance de ser destruído do que o esquema ****10. Quanto mais afastadas estiverem as características definidas (1 e 0) de um esquema, maior é a probabilidade de que o cruzamento ocorra entre elas, fazendo com que o esquema desapareça da população. Em um esquema com tamanho l existem $(l - 1)$ pontos onde pode ocorrer cruzamento. A probabilidade de desaparecimento de um esquema é portanto a razão entre o seu comprimento *d(H)* e $(l - 1)$. O inverso, a probabilidade de sobrevivência (*ps*) é, portanto:

$$p_s = 1 - \frac{\delta(H)}{l-1}$$

Considerando a probabilidade de ocorrer cruzamento (*pc*) temos:

$$p_s \geq 1 - p_c.\frac{\delta(H)}{l-1}$$

A nova equação do número de representantes do esquema no momento $(t + 1)$ com relação ao momento t fica sendo:

$$m(H,t+1) \geq m(H,t).\frac{f(H)}{\bar{f}}.\left(1-p_c.\frac{\delta(H)}{l-1}\right)$$

De forma equivalente, a probabilidade de sobrevivência de um esquema ao operador de mutação é igual a $(1 - pm).o(H)$, onde a ordem $o(H)$ é o número de bits com valor definido (1 ou 0). Como os valores da probabilidade de mutação são normalmente muito pequenos, é possível aproximar esta expressão para $(1 - o(H).pm)$.

$$m(H,t+1) \geq m(H,t).\frac{f(H)}{\bar{f}}.\left(1-p_c.\frac{\delta(H)}{l-1}-p_m.o(H)\right)$$

Esta é a expressão do Teorema do Esquema, base da teoria de Algoritmos Genéticos. Ele enfatiza as três características que garantem a manutenção de um determinado esquema na população. Em primeiro lugar, a *fitness* do esquema deve ser maior que a *fitness* média da população. Em seguida, os termos de *pc* e pm devem ser minimizados, isto é, esquemas com comprimento $d(H)$ pequeno e ordem $o(H)$ baixa serão beneficiados com um número crescente de representantes.

Em seu trabalho, Lobo utiliza o Teorema do Esquema para justificar a configuração dos valores da taxa de seleção (s) e da taxa de cruzamento (pc). A taxa de seleção representa a pressão do algoritmo em manter nas futuras gerações os indivíduos da população atual que têm os melhores valores de *fitness*. A taxa de cruzamento indica a participação da recombinação dos genes dos pais na formação de novos indivíduos. Em Algoritmos Genéticos, estes dois parâmetros têm o propósito de garantir o crescimento dos *building blocks*, ou seja, pedaços de cromossomo que são úteis na formação de um indivíduo-solução.

Desconsiderando o operador de mutação e simplificando a expressão geral do Teorema do Esquema obtém-se a taxa de crescimento:

$$\frac{m(H,t+1)}{m(H,t)} = s.(1-p_c)$$

Como vimos anteriormente, para garantir a sobrevivência dos esquemas (aumentar a taxa de crescimento) podemos aumentar a taxa de seleção (s) ou diminuir a probabilidade de cruzamento (pc). Considerando uma taxa de crescimento igual a 2, suficiente para garantir o crescimento dos *building blocks*, o autor propõe fixar a taxa de seleção em 4 e a taxa de cruzamento em 0,5. Com isso, o problema de adaptação dos parâmetros para encontrar boas soluções passa a ser um problema de escolher corretamente o tamanho da população.

4.2. CONTROLE DO TAMANHO DA POPULAÇÃO

Eliminar o tamanho da população é mais complexo por se tratar de um parâmetro capaz de interferir fortemente no processo de busca e no uso dos recursos do sistema. Quanto maior for a população, maior será a quantidade de pontos de exploração no espaço de busca e, portanto, será mais provável que a solução seja encontrada em poucas interações. Por outro lado, um aumento excessivo na quantidade de indivíduos na população pode representar um alto custo em termos de memória e tempo de execução. O ideal é que o tamanho da população seja o menor possível com o qual se consiga chegar a uma solução em um tempo aceitável.

Lobo propõe que a escolha do valor do tamanho da população seja feita de uma forma dinâmica, estabelecendo uma competição entre várias populações de tamanhos distintos. Cada população (i + 1) tem o dobro do tamanho da população anterior i. O processamento se divide de modo a executar as gerações de forma intercalada. Para equilibrar a disputa, faz-se com que para cada geração da população (i + 1) sejam executadas 4 gerações da população i. Dessa forma, a facilidade que as maiores populações têm de encontrar a solução é contrabalançada pelo número maior de execuções das populações menores. Esta estratégia aumenta a probabilidade de que uma população pequena chegue a uma solução mais rapidamente, o que é interessante uma vez que o objetivo é resolver o problema com a menor população possível. A

tabela 1 mostra um exemplo de execução.

TABELA 1 - SEQÜÊNCIA NORMAL DE EXECUÇÃO DAS POPULAÇÕES

Contador	População	Geração
100000...	1	1
200000...	1	2
300000...	1	3
010000...	2	1
110000...	1	4
210000...	1	5
310000...	1	6
020000...	2	2
120000...	1	7
220000...	1	8
320000...	1	9
030000...	2	3
130000...	1	10
230000...	1	11
330000...	1	12
001000...	3	1
101000...	1	13

Como se pode observar, uma nova população só poderá ser criada se a população imediatamente anterior já tiver sido executada 3 vezes.

Em uma situação normal, o mais provável é que as populações menores produzam melhores resultados. Isto significa que a média da *fitness* dos indivíduos de populações menores tende a ser maior do que a de populações maiores, como mostra a tabela 2. Estamos considerando que quanto maior o valor de fitness, melhor é este valor.

TABELA 2 - VALORES NORMAIS DA *FITNESS* MÉDIA DE CADA POPULAÇÃO

População	Geração	*Fitness* Média
1	30	12,6
2	6	11,8
3	1	7,8

Eventualmente, a *fitness* média de uma população maior poderá ultrapassar a de uma população menor. Quando isto ocorrer não haverá mais sentido em continuar executando a população menor, e ela será descartada, como mostrado na tabela 3. Este processo é chamado de *overtaking*.

TABELA 3 - *OVERTAKING* DA POPULAÇÃO 2 SOBRE A 1

População	Geração	*Fitness* Média
1	30	12,6
2	7	13,2
3	1	7,8

Nesta situação, o contador é re-iniciado e a população 2 passa a ser a menor população em execução. Este processo de competição continua até que uma das populações chegue a uma solução ou até que o tamanho máximo de população seja atingido (critério de término).

O tópico a seguir traz um resumo dos resultados obtidos por Lobo utilizando a técnica "AG sem parâmetros".

4.3. RESULTADOS OBTIDOS POR LOBO

Em sua tese de Doutorado [Lobo 2000], o autor avalia sua proposta através de alguns experimentos clássicos e de um estudo de caso de um problema real que consiste em projetar automaticamente uma rede de distribuição de energia elétrica

otimizada, com base na localização geográfica das subestações, transformadores e residências. Apresentaremos agora um resumo das principais conclusões deste trabalho.

Com relação aos parâmetros configurados com base na análise teórica do teorema do esquema, os valores de 0,5 para a taxa de cruzamento (*pc*) e 4 para a taxa de seleção (*s*) foram escolhidos com o objetivo de evitar-se os casos extremos, nos quais a pressão de seleção é muito baixa ou muito alta. O autor admite que é possível conseguir uma melhor eficiência em alguns casos modificando-se estes valores de *pc* e *s*, porém, o objetivo de seu trabalho não é atingir a máxima eficiência possível.

Nos testes realizados com a adaptação do tamanho da população, a técnica "AG sem parâmetros" mostrou-se mais lenta do que AG tradicional com parâmetros ótimos. O fator de atraso aumenta com a complexidade do problema. No entanto, o autor justifica esta lentidão lembrando que não é possível saber *a priori* quais são os valores ótimos que conferem a melhor eficiência possível a um determinado problema.

Os resultados obtidos pelos diversos testes feitos com diferentes tipos de implementações de AG comprovam ainda que o uso da técnica sem parâmetros é completamente independente da implementação escolhida, podendo ser usada em qualquer AG cuja base seja a operação de cruzamento.

Lobo destaca as duas principais vantagens da técnica adaptativa em comparação com o AG tradicional. Em primeiro lugar ela torna o algoritmo mais robusto e confiável, pois ele é capaz de perceber a necessidade de uma população maior. Em segundo lugar, a técnica claramente simplifica o uso de AG como ferramenta para a solução de problemas reais, uma vez que não é exigido do usuário qualquer conhecimento mais profundo a respeito da influência dos parâmetros mais críticos, deixando que esta configuração seja feita dinamicamente pelo próprio algoritmo.

O capítulo seguinte mostra como este método foi implementado em um sistema de Programação Genética.

5. IMPLEMENTAÇÃO E EXPERIMENTOS

Este capítulo mostra em detalhes como a técnica de controle de parâmetros proposta por Lobo foi adaptada e implementada no presente trabalho em um sistema de Programação Genética. Os experimentos escolhidos e a metodologia utilizada nos testes também são descritos no final deste capítulo.

Como base para a implementação, foi escolhida uma ferramenta reconhecida mundialmente chamada *lil-gp* [Zongker 1998], freqüentemente utilizada por diversos pesquisadores [Banzhaf 1998]. Para comandar a ferramenta, foi implementado um Módulo de Controle de Populações externo, que será apresentado neste capítulo.

Para a fase de experimentos foram selecionados problemas clássicos, freqüentemente utilizados para avaliar sistemas em Computação Evolucionária.

5.1. A FERRAMENTA *LIL-GP*

Lil-gp [Zongker 1998] é uma ferramenta de domínio público criada por um grupo de pesquisa da *Michigan State University* para facilitar o desenvolvimento de aplicações de Programação Genética que segue as especificações de John Koza [Koza 1992].

A ferramenta *lil-gp* evolui programas de computador representados sob a forma de árvores, conforme mostrado no capítulo 2, em que os nós são ponteiros para funções escritas em C. Este tipo de estrutura faz com que a avaliação dos indivíduos seja extremamente rápida, permitindo a manipulação de problemas complexos. Além disso, o *lil-gp* é capaz de emular o "*Simple LISP Code*" e os parâmetros padrão conforme descritos na obra de John Koza [Koza 1992].

Especificamente com relação a este trabalho, o *lil-gp* possui algumas características que foram essenciais para a implementação:

- definição de parâmetros pela linha de comando e/ou arquivo de configuração;

- leitura e gravação das populações em arquivo, permitindo a interrupção e recuperação do processo a qualquer momento;
- geração de arquivos de estatísticas para o acompanhamento do processo;
- código em ANSI C portável para Linux/Windows/Mac.

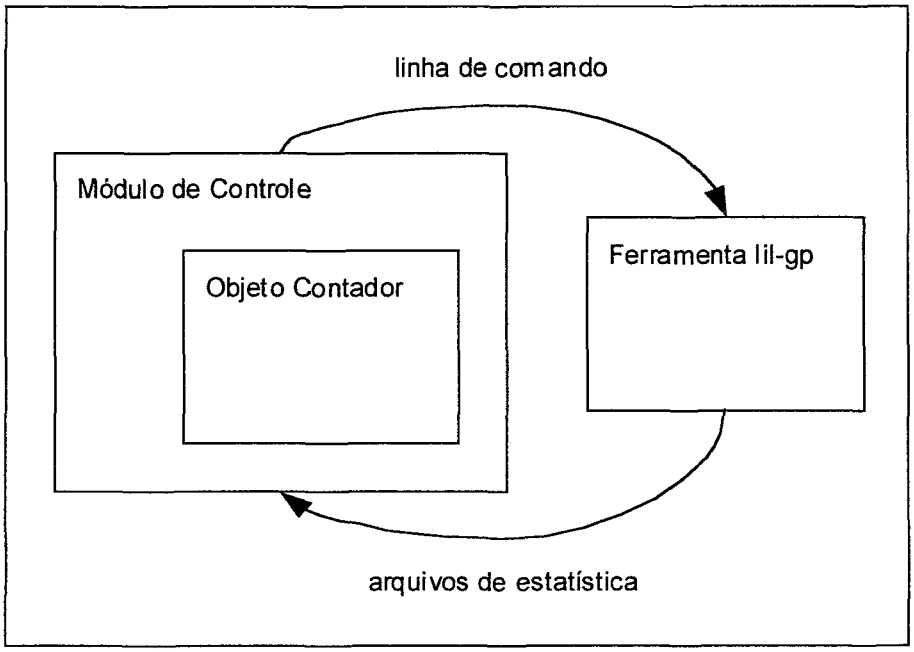
Os recursos do *lil-gp* permitem que o processo de adaptação do tamanho da população seja controlado por uma aplicação independente que faz chamadas ao *lil-gp* passando os parâmetros necessários. Desta forma, nenhuma alteração foi feita no código central da ferramenta (*kernel*), o que garante a confiabilidade dos resultados obtidos e a possibilidade de comparação direta com a ferramenta pura, antes da adaptação.

5.2. MÓDULO DE CONTROLE DE POPULAÇÕES

O módulo de controle de populações foi implementado para controlar a ferramenta *lil-gp* e reproduzir em PG a técnica de adaptação do tamanho da população proposta por Lobo inicialmente para AG. Segundo esta técnica, várias populações de tamanhos diferentes competem entre si, e cada população tem o dobro do tamanho da anterior.

O algoritmo principal deve alternar a execução das populações com base em um contador de base 4, como mostrado no capítulo anterior. Uma população de tamanho 2x só poderá ser processada quando a população imediatamente anterior em tamanho (tamanho x) já tiver sido executada 3 vezes. Este comportamento é determinado por um objeto da classe "contador" que, com base em algumas variáveis de controle, retorna o número da próxima população que deve ser tratada. O programa principal recebe este número e gera a linha de comando apropriada para o *lil-gp*, como mostra a figura 6.

FIGURA 6 - FUNCIONAMENTO DO MÓDULO DE CONTROLE



O módulo de controle possui uma estrutura de dados baseada em dois vetores principais:

- Vetor de contagem - Cada posição do vetor varia de 0 a 2 e indica em que execução está aquela população. Por exemplo, a população 2 somente será executada quando a população 1 (posição 1 do vetor de contagem) for igual a 2 (já foi executada 3 vezes). A execução da população 2 faz com que o valor da contagem da população 2 seja incrementado e a contagem da população 1 volte a 0.
- Vetor de *fitness* média - Cada posição do vetor armazena o valor da *fitness* média que foi alcançado por aquela população na sua última execução. Este valor é um importante indicador da qualidade dos indivíduos de uma população e é utilizado por um método que verifica a ocorrência de *overtaking* (sempre que uma população maior ultrapassar em qualidade uma população menor, a menor será eliminada).

Durante uma execução existem várias populações ativas, que são atualizadas em disco a cada geração utilizando o recurso de *checkpoint* fornecido pelo *lil-gp*. As

populações eliminadas durante o processo são apagadas do disco e deixam de existir na contagem. As demais populações continuam competindo normalmente até que uma delas encontre uma solução, ou até que o critério de parada seja verificado.

Em um algoritmo sem adaptação, o critério de parada (que impede que a busca pela solução continue indefinidamente) é o número máximo de gerações. Com a introdução de múltiplas populações, este conceito perde seu valor, uma vez que cada população terá o seu número de geração particular. Neste caso, usa-se como critério de parada o tamanho máximo da população. Através do módulo de controle, pode-se definir qual será este tamanho e, caso o algoritmo tente criar uma população maior que esta, o processo é interrompido e considera-se que uma solução não foi encontrada. Este controle é muito importante pois, dependendo das limitações da máquina (memória), pode não ser possível trabalhar com populações excessivamente grandes.

Para possibilitar a análise estatística dos resultados, o módulo de controle gera ainda um arquivo de estatística personalizado, com base nos dados de 6 outros arquivos de estatística fornecidos pela ferramenta *lil-gp* a cada geração.

5.3. EXPERIMENTOS

Para testar o módulo de controle, a ferramenta *lil-gp* foi compilada para resolver dois problemas clássicos freqüentemente utilizados na avaliação de sistemas de Programação Genética: o da trilha da formiga e o da regressão simbólica e. A seguir são apresentadas as descrições desses problemas e os parâmetros adotados para os testes.

5.3.1. O Problema da Trilha da Formiga

O problema da trilha da formiga [Koza 1992] é uma aplicação clássica de Programação Genética muito utilizada como *benchmark* por seu nível de dificuldade [Langdon 1998].

Um plano bi-dimensional dividido em células (*grid*) representa a área sobre a

qual pode caminhar uma formiga artificial. Algumas células de comida são distribuídas sobre o *grid* e o objetivo é fazer com que a formiga passe por cima do maior número possível de células com comida. Como resultado teremos um programa de computador cuja execução faz a formiga caminhar sobre o plano.

O conjunto de funções que podem ser executadas sobre a formiga inclui uma função principal e três auxiliares. A sintaxe da função principal é:

(if-food-ahead (ação1) (ação2))

Se houver comida na célula imediatamente à frente, a ação1 será executada. Caso contrário, a ação2 o será. As três funções auxiliares são usadas para executar mais de uma ação: *progn2* (2 argumentos), *progn3* (3 argumentos) e *progn4* (4 argumentos).

O conjunto de terminais possui três comandos:

- *move* - faz com que a formiga ande uma célula para a frente;
- *right* - faz com que a formiga se vire para a direita;
- *left* - faz com que a formiga se vire para a esquerda.

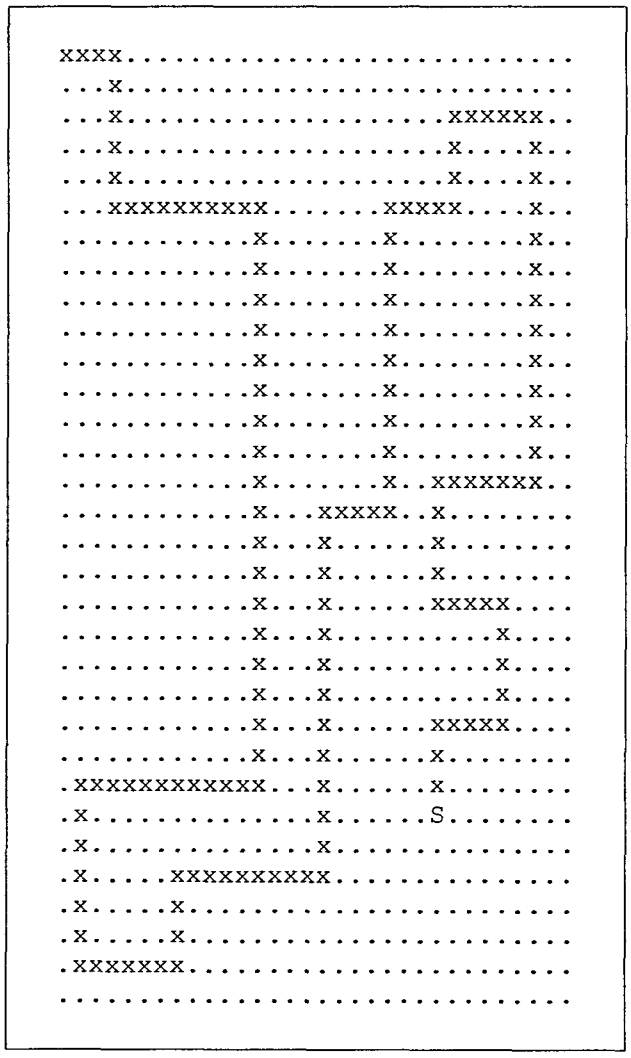
Dependendo do arranjo das células de comida, a configuração ou trilha recebe um nome. Uma das mais importantes é a trilha de Santa Fé, um *grid* de 32 x 32 células com 89 células de comida. Existe também a trilha de Los Altos com 100 x 100 células sendo 105 de comida. A configuração utilizada para os testes foi a trilha de Santa Fé.

O código-fonte de um exemplo de solução para este problema é apresentadona figura 7. A figura 8 mostra o caminho percorrido pela formiga, obtido pela execução do programa da figura 7. Neste caso, a formiga consegue percorrer todas as 89 células de comida.

FIGURA 7 - EXEMPLO DE SOLUÇÃO PARA O PROBLEMA DA FORMIGA DE SANTA FÉ

```
(progn2 (if-food-ahead (if-food-ahead (progn3 move move move)
      (if-food-ahead (progn2 move right)
            (progn3 (progn2 move right)
                  (if-food-ahead left move)
                  (if-food-ahead (if-food-ahead right left) left))))
      (progn2 (progn2 left left)
            (if-food-ahead (if-food-ahead (progn2 (progn2 move move)
                  (if-food-ahead left move))
            (progn2 move
                  (progn3 right left right)))
            (progn2 (if-food-ahead (progn2 move right) right) move))))
      (if-food-ahead (progn2 move right) right))
```

FIGURA 8 - TRILHA PERCORRIDA PELO PROGRAMA DA FIGURA 7



A avaliação dos programas (*fitness*) é medida diretamente pelo número de células de comida alcançadas pela formiga. Sendo assim, um indivíduo é considerado solução para o problema se a sua execução for capaz de direcionar a formiga por todas as 89 células que contém comida.

Como restrição para evitar programas excessivamente grandes, o número máximo de passos da formiga foi limitado em 400. Cada terminal (*move*, *right* ou *left*) conta como 1 passo. As funções (*if-food-ahead*, *progn2*, *progn3* e *progn4*) não são consideradas.

A lista completa com todos os parâmetros bem como a metodologia utilizada nos testes será apresentada na tabela 4, no final deste capítulo.

5.3.2. O Problema da Regressão Simbólica

Neste problema, o objetivo é encontrar a equação de uma curva que passe por um conjunto de pontos conhecidos. A ferramenta *lil-gp* conhece a equação real da curva e calcula automaticamente os valores de 20 pontos da curva-objetivo.

Para o cálculo do valor de aptidão ou *fitness*, cada par (x,y) é um *fitness case* (caso de treinamento). O programa recebe cada valor de x e retorna o valor de y correspondente. Este valor é então comparado com o valor de y esperado (fornecido pelo *lil-gp*). Se a diferença entre eles for menor do que o valor da margem de erro, considera-se que houve 1 acerto. O valor da *fitness* é igual ao valor de acertos obtidos pelo programa após a análise de todos os casos de treinamento (*fitness cases*).

O conjunto de funções inclui as principais operações matemáticas: soma, subtração, multiplicação, divisão protegida (divisão por 0 retorna 1), seno, cosseno, exponenciação e logaritmo protegido (log do módulo do número, log de 0 retorna 0).

Os terminais podem ser ou a variável independente x ou uma constante aleatória entre -1 e $+1$, cujo valor é escolhido no início da execução ou em operações de mutação, permanecendo fixo durante o restante do tempo.

5.4. METODOLOGIA E PARÂMETROS UTILIZADOS

Foram realizadas várias baterias de testes com diferentes configurações, procurando isolar a influência de cada um dos três fatores que poderiam influenciar os resultados:

a) Adaptação do tamanho da população - Nos testes com e sem adaptação, a ferramenta *lil-gp* foi configurada da mesma maneira, para garantir a confiabilidade na comparação dos resultados. A única diferença está na presença ou não do módulo de controle.

b) Tipo de problema implementado - Para cada um dos dois casos, com e sem adaptação, foram implementados dois dos principais tipos problemas, freqüentemente utilizados para a avaliação de eficiência em sistemas de Programação Genética. Com isto pretende-se obter uma relativa independência dos resultados em relação ao contexto do problema proposto.

c) Parâmetros do algoritmo genético - Para cada um dos dois problemas, os testes foram feitos com dois conjuntos de parâmetros: os parâmetros padrão propostos por Koza em seu segundo livro [Koza 1994] e os parâmetros modificados por Lobo [Lobo 2000] em sua análise envolvendo o Teorema do Esquema, conforme mostrado no capítulo anterior. O objetivo é verificar a influência real deste estudo teórico nos resultados, comparando-os com os parâmetros normalmente utilizados.

A seguir será apresentada a tabela 4, que mostra todos os parâmetros utilizados em cada uma das baterias de testes.

TABELA 4 - PARÂMETROS UTILIZADOS NOS EXPERIMENTOS

Módulo de controle →	Desligado (sem adaptação)				Ligado (com adaptação)			
Tipo de problema →	Formiga		Regressão		Formiga		Regressão	
Configuração de parâmetros →	Koza2	Lobo	Koza2	Lobo	Koza2	Lobo	Koza2	Lobo
Tamanho da população	500, 1000, 2000, 4000, ..., 32000				(automático)			
Número máximo de gerações	100				(sem limite)			
Método de inicialização	*Ramped half-and-half*							
Profundidade inicial dos programas	2 a 6							
Profundidade máxima dos programas	17							
Método de seleção	Torneio							
Tamanho do torneio	7	4	7	4	7	4	7	4
Taxa de reprodução	10 %	50 %	10 %	50 %	10 %	50 %	10 %	50 %
Taxa de cruzamento	90 %	50 %	90 %	50 %	90 %	50 %	90 %	50 %
Taxa de mutação	0 %							

Com relação à metodologia utilizada nos testes, foram feitas ainda as seguintes considerações adicionais:

a) Em todos os testes foram executadas 50 rodadas. Com este valor podemos garantir com grande confiabilidade que os resultados representam o comportamento geral da média das execuções, e não foram influenciados por condições locais. Desta forma, cada uma das curvas mostradas nos gráficos é na verdade a curva média de 50 curvas que representam as 50 execuções realizadas.

b) O *lil-gp*, como a maioria dos sistemas de Programação Genética, permite que a semente que alimenta o gerador de números aleatórios seja escolhida externamente. Nos testes realizados, uma semente diferente foi escolhida para cada rodada. Além de garantir a diversidade entre os números aleatórios gerados, o uso de uma semente conhecida permite também a fiel reprodução dos testes, caso haja alguma interrupção ou simplesmente para a conferência dos resultados.

6. RESULTADOS

Durante os testes, as informações estatísticas foram sendo armazenadas em arquivos de texto, posteriormente incorporados a diversas planilhas eletrônicas integradas. A partir desses dados e após vários cálculos estatísticos e de natureza lógica, é possível verificar o que ocorreu, passo a passo, em cada uma das 600 rodadas realizadas em 12 baterias de testes.

Para mostrar a evolução de cada teste ao longo das gerações foram feitos gráficos que mostram a probabilidade de sucesso em cada etapa. A probabilidade de sucesso é uma medida relativa entre 0 e 100 % que indica qual é a chance de se encontrar uma solução válida em uma rodada hipotética qualquer. Em outras palavras, para calcular a probabilidade de sucesso até uma dada geração, divide-se o número de rodadas que obtiveram sucesso até aquela geração pelo número total de rodadas executadas (50). Isto é feito para cada uma das gerações, resultando em um gráfico cuja tendência é crescente, uma vez que a cada geração que passa aumentam as chances de que o algoritmo já tenha encontrado a solução.

Por outro lado, como nosso objetivo é comparar o comportamento de populações de diferentes tamanhos, não faz sentido utilizar o número de gerações como referência no eixo x. Uma geração de uma população com 500 indivíduos não pode ser comparada, por exemplo, com uma geração de uma população com tamanho 32000, que produz um custo computacional 64 vezes maior. Para permitir a comparação entre os resultados, a solução encontrada foi fazer o gráfico de probabilidade de sucesso em função do número de indivíduos processados, e não mais do número de gerações. Para tanto, basta multiplicar o número da geração pelo número de indivíduos da população em questão. Assim, a probabilidade de sucesso da 10ª geração da população de tamanho 500 aparecerá no gráfico em $x = 500 \times 10 = 5000$. Se a população fosse de 32000, o valor seria marcado em $x = 32000 \times 10 = 3200000$. Desta forma é possível desenhar em um mesmo gráfico as curvas das diversas populações e fazer comparações entre elas.

6.1. RESULTADOS SEM CONTROLE DO TAMANHO DA POPULAÇÃO

Nesta fase inicial dos testes, a ferramenta *lil-gp* trabalhou sozinha, sem a utilização do módulo de controle.

Cada um dos resultados é apresentado em 2 gráficos. No gráfico principal são traçadas as 7 curvas referentes às 7 populações testadas: com 500, 1000, 2000, 4000, 8000, 16000 e 32000 indivíduos. Estas curvas representam a probabilidade de sucesso ao longo da evolução; ou seja, qual é, em cada instante, a probabilidade de que uma execução hipotética qualquer consiga chegar a uma solução válida até aquele ponto do processo evolutivo. A unidade de tempo comum, que permite a comparação direta mesmo entre populações de diferentes tamanhos, é o número de indivíduos processados, que expressa a dimensão do custo computacional. Vale ressaltar que, como foram feitas 50 execuções de cada experimento, cada uma das 7 curvas traçadas é na realidade a curva média de 50 curvas obtidas experimentalmente.

O segundo gráfico mostra, para cada população que conseguiu encontrar a solução, em qual das 100 gerações isso ocorreu. A primeira barra se refere ao menor valor, à primeira geração. A barra da direita mostra o valor da última geração, aquela de maior valor. A barra central é a mais significativa e indica a geração média em que a maior parte das soluções está sendo encontrada.

6.1.1. Como Interpretar os Gráficos

No gráfico principal podemos visualizar a evolução da taxa de acerto de cada população. Quanto mais tempo o algoritmo tiver para evoluir, maior será a probabilidade de que ele chegue a uma solução. Desta forma, em uma situação normal, esse é um gráfico crescente que tente a atingir 100 % de sucesso em um tempo infinito. Quanto mais rápido for o crescimento da curva, melhor será o desempenho do algoritmo, pois isto significa que ele está obtendo sucesso com um menor custo de processamento.

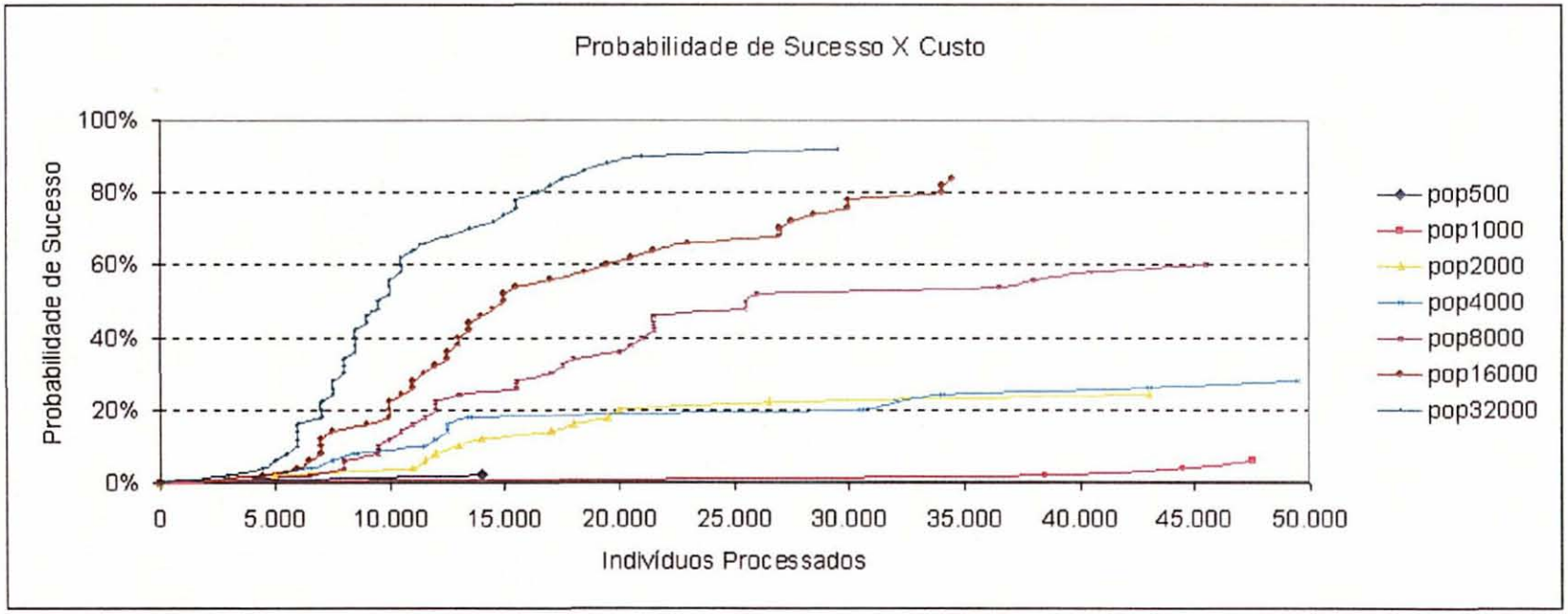
O segundo gráfico deve ser analisado junto com o primeiro. Ele mostra, nas

execuções que obtiveram sucesso, em que geração as soluções estão sendo encontradas. Como ele só pode considerar as execuções válidas, ele precisa do gráfico principal para ser analisado corretamente. Assim, valores baixos no segundo gráfico não significam necessariamente que o algoritmo está obtendo sucesso. Significam apenas que, nas vezes em que o algoritmo encontrou a solução, ela foi encontrada em uma das primeiras gerações. Para sabermos o percentual de vezes em que uma solução está sendo encontrada recorremos ao primeiro gráfico.

A seguir são mostrados todos os 6 conjuntos de gráficos dos testes sem adaptação, para os 3 problemas, cada um com os 2 conjuntos de parâmetros:

- Problema da trilha da formiga de Santa Fé, com parâmetros do livro Koza 2.
- Problema da trilha da formiga de Santa Fé, com os parâmetros propostos por Lobo.
- Problema da regressão simbólica com a equação $x^3 + x^2 + x$, com parâmetros do livro Koza 2.
- Problema da regressão simbólica com a equação $x^3 + x^2 + x$, com os parâmetros propostos por Lobo.
- Problema da regressão simbólica com a equação $x^4 + x^3 + x^2 + x$, com parâmetros do livro Koza 2.
- Problema da regressão simbólica com a equação $x^4 + x^3 + x^2 + x$, com os parâmetros propostos por Lobo.

FIGURA 9 - GRÁFICOS DO EXPERIMENTO DA TRILHA DA FORMIGA, SEM ADAPTAÇÃO, USANDO PARÂMETROS DO KOZA 2



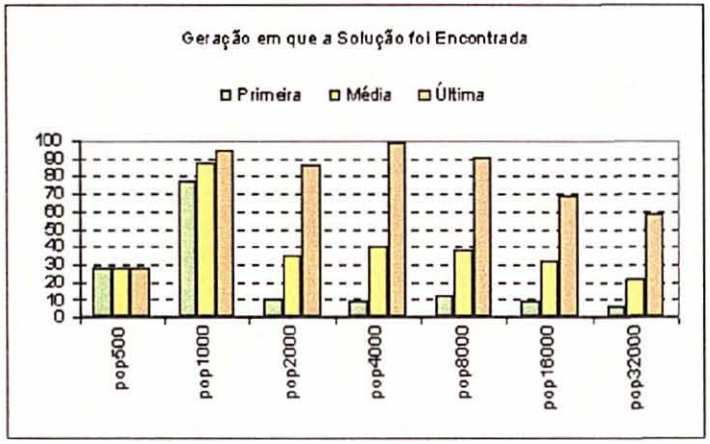
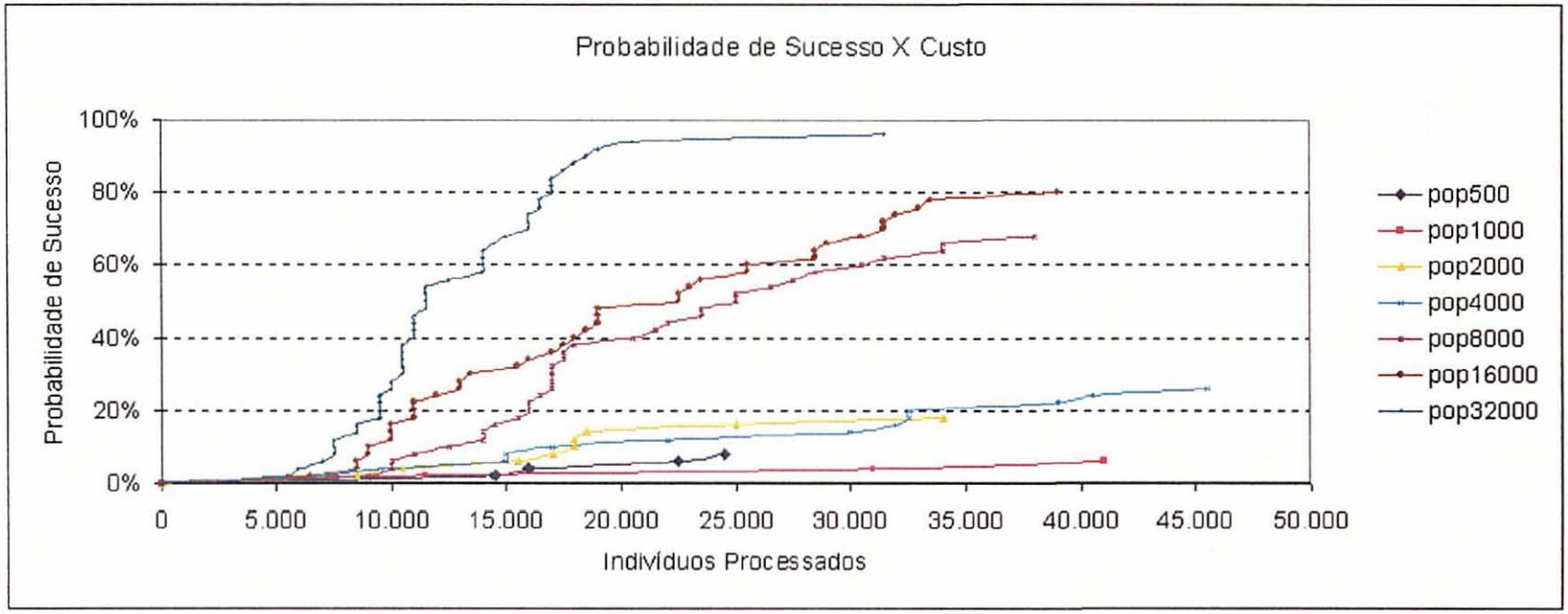


FIGURA 10 - GRÁFICOS DO EXPERIMENTO DA TRILHA DA FORMIGA, SEM ADAPTAÇÃO, USANDO PARÂMETROS DO LOBO



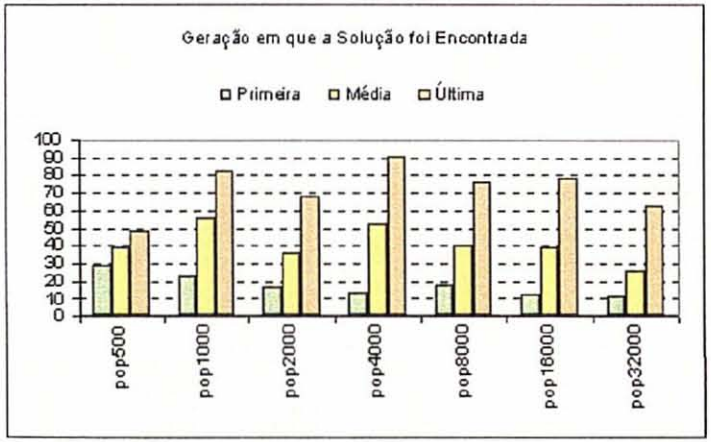
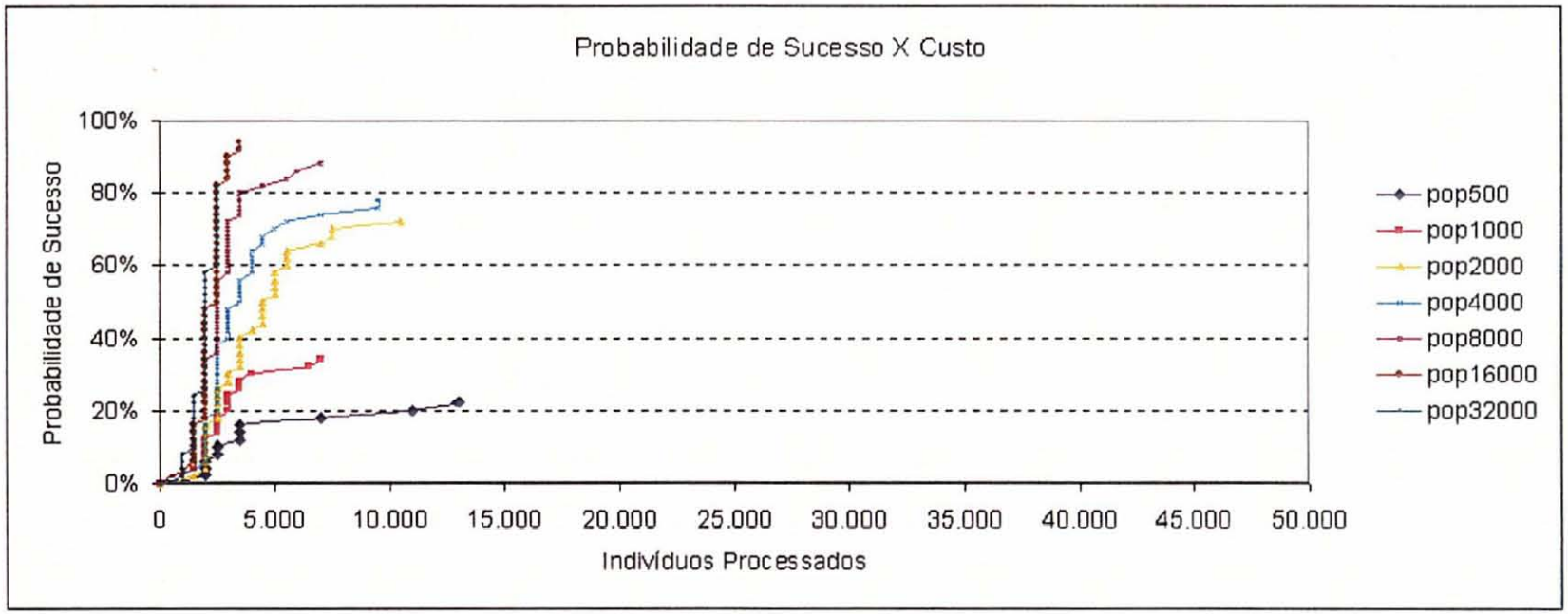


FIGURA 11 - GRÁFICOS DO EXPERIMENTO DA REGRESSÃO DE ORDEM 3 (X^3), SEM ADAPTAÇÃO, USANDO PARÂMETROS DO KOZA 2



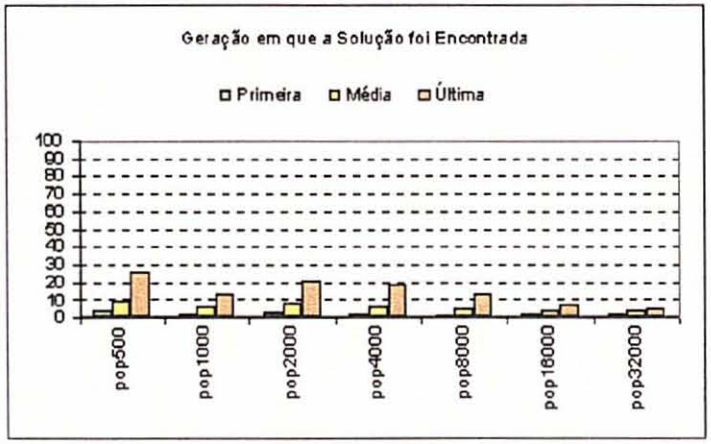
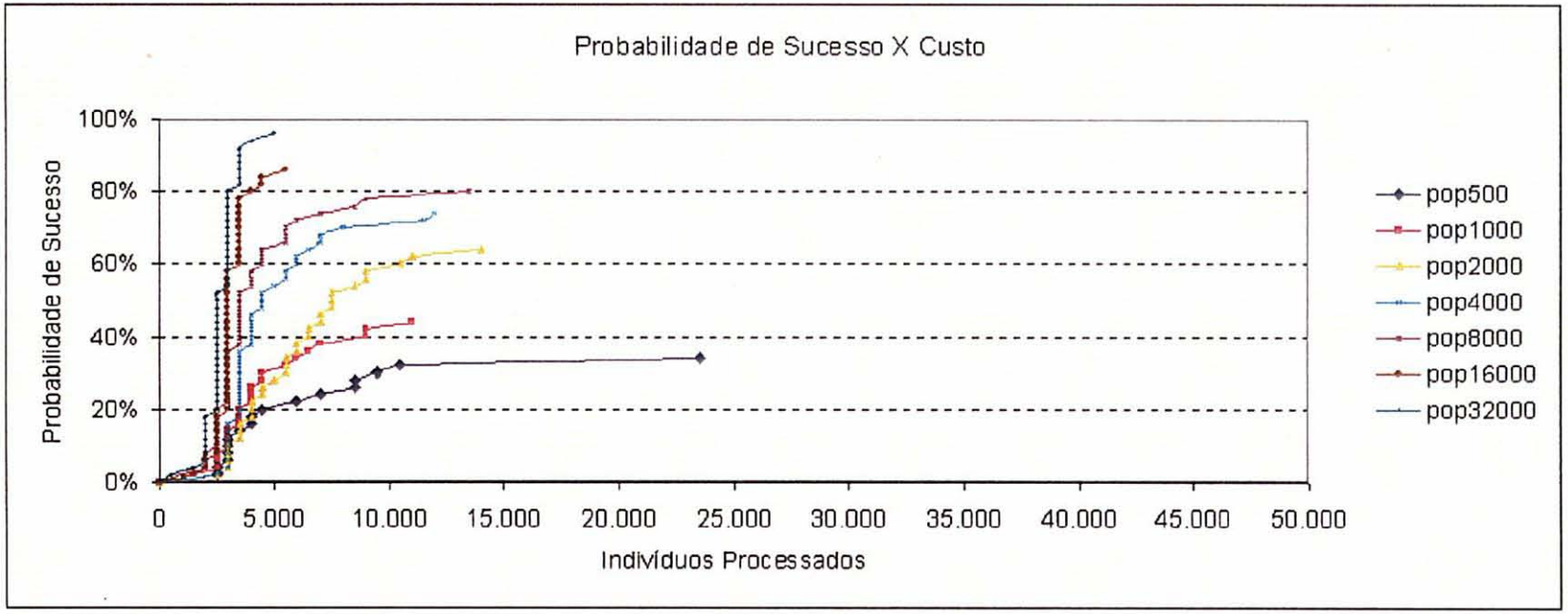


FIGURA 12 - GRÁFICOS DO EXPERIMENTO DA REGRESSÃO DE ORDEM 3 (X^3), SEM ADAPTAÇÃO, USANDO PARÂMETROS DO LOBO



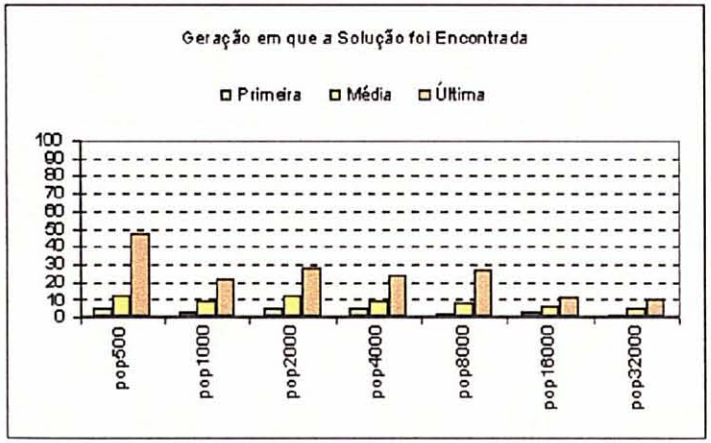
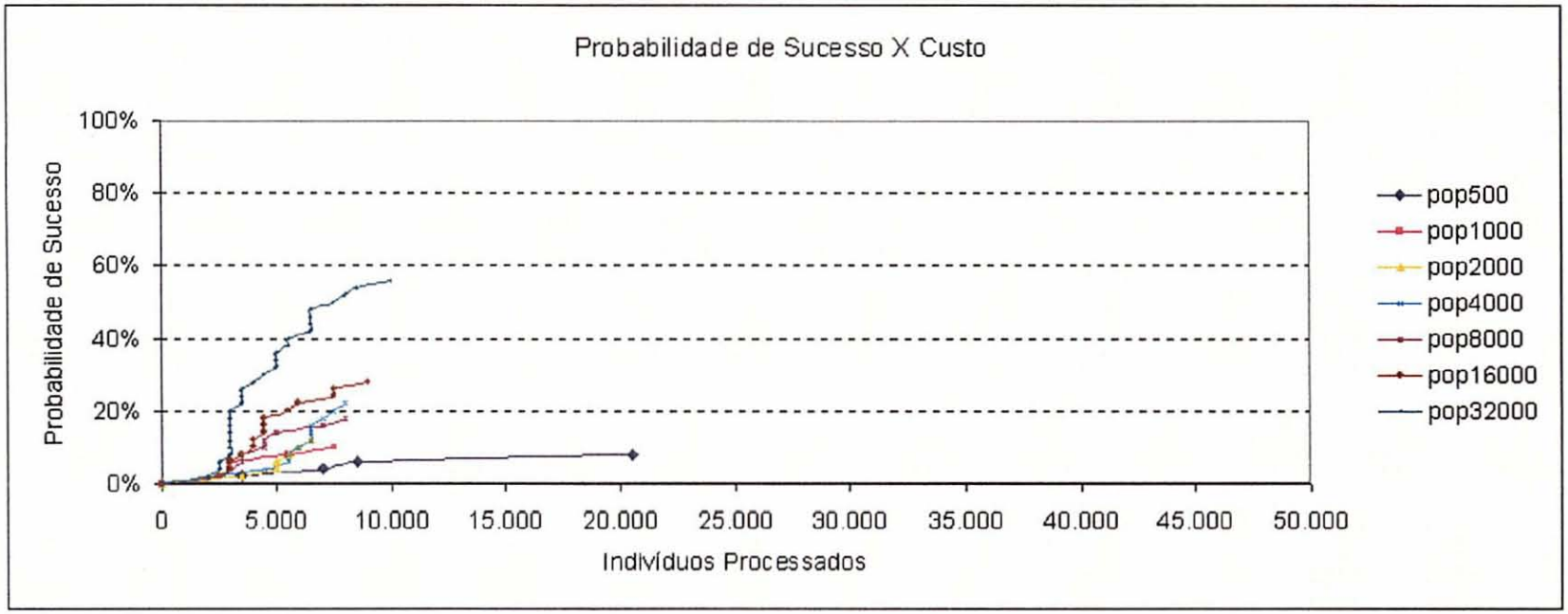


FIGURA 13 - GRÁFICOS DO EXPERIMENTO DA REGRESSÃO DE ORDEM 4 (X^4), SEM ADAPTAÇÃO, USANDO PARÂMETROS DO KOZA 2



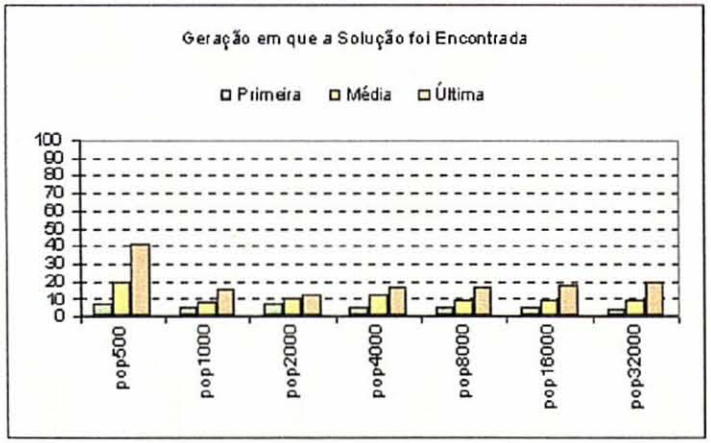
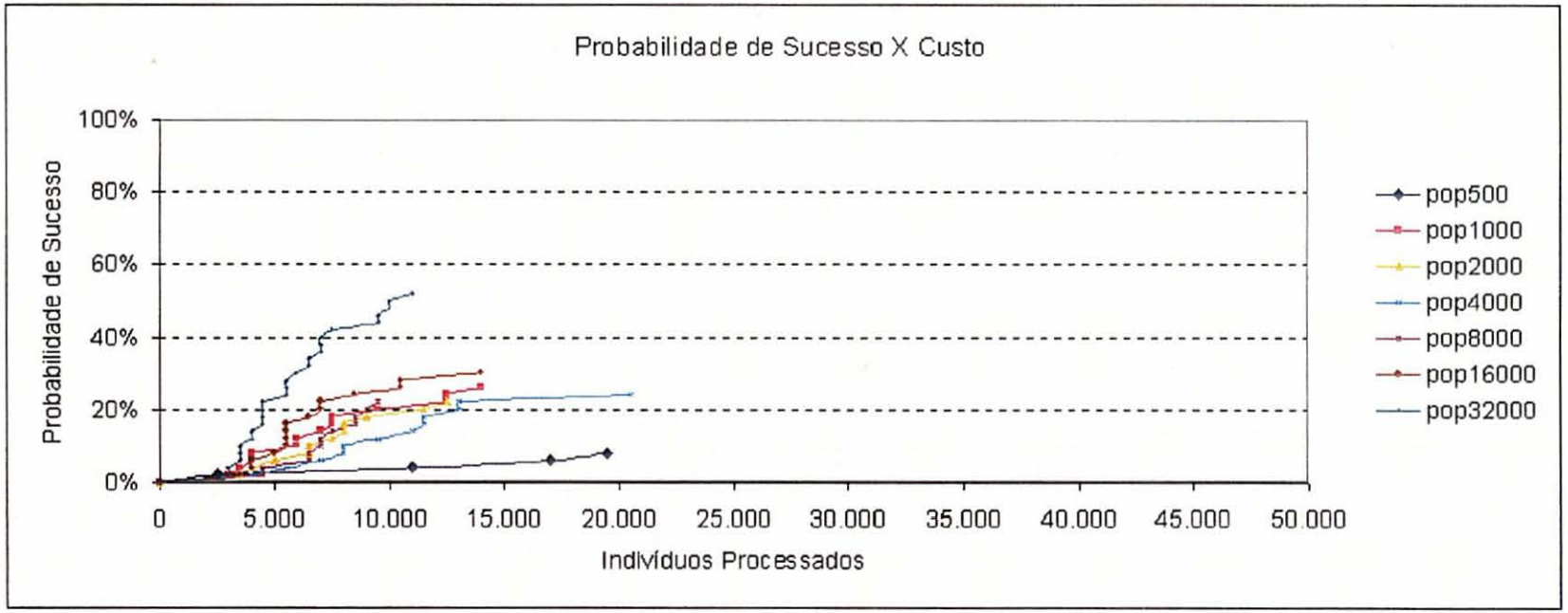
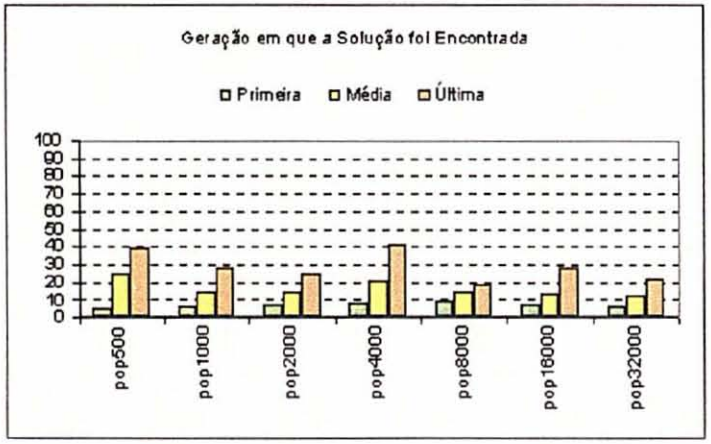


FIGURA 14 - GRÁFICOS DO EXPERIMENTO DA REGRESSÃO DE ORDEM 4 (X^4), SEM ADAPTAÇÃO, USANDO PARÂMETROS DO LOBO





6.1.2. Análise dos Resultados sem Adaptação

Os três problemas implementados apresentaram níveis de dificuldades distintos, o que pode ser observado pela velocidade de crescimento e pelo valor final da probabilidade de sucesso. O problema da regressão de ordem 3 foi o mais simples, atingindo os melhores índices de sucesso, mesmo com populações pequenas. O problema da trilha da formiga é de dificuldade média, apresentando uma boa separação das curvas das populações. Finalmente, o problema da regressão de ordem 4 foi o que apresentou menores taxas de sucesso, mesmo em populações grandes como a de 32000 indivíduos. Este resultado confirma o fato de que o nível de dificuldade deste tipo de problema cresce muito rapidamente com o aumento na ordem do polinômio que se deseja obter.

Quanto à variação na configuração dos parâmetros (taxa de cruzamento e tamanho do torneio), percebemos pouca diferença entre os gráficos obtidos usando os parâmetros do livro Koza 2 e os propostos por Lobo. Em alguns casos, como no experimento da regressão de ordem 3, os parâmetros de Lobo melhoraram os resultados para as populações de 500 e 1000, mas pioraram para as de 2000, 4000 e 8000. Já no problema da regressão de ordem 4 vemos que os parâmetros de Lobo tiveram uma pequena influência negativa sobre a população de 32000. Porém, como as diferenças são pequenas, não é possível afirmar com confiança que a mudança dos parâmetros de Koza 2 para Lobo tenha ocasionado qualquer mudança significativa no comportamento do algoritmo.

De uma forma geral, concluímos que os resultados sem adaptação correspondem às expectativas. Populações maiores apresentam maiores índices de sucesso, enquanto populações menores podem entrar em um estado de estagnação, não conseguindo sair dele mesmo após diversas interações. Esse fenômeno pode ser claramente observado no gráfico do problema da trilha da formiga, onde as populações de 500 e 1000 obtiveram taxas de sucesso próximas a 0 % enquanto a população de

32000 ultrapassou a marca de 90 % mesmo com o limite máximo de 100 gerações. Estes resultados comprovam o fato de que o tamanho da população é um parâmetro crítico para algoritmos evolutivos, e que valores inadequados para este parâmetro podem complicar ou até mesmo impossibilitar a descoberta da solução. A explicação para isto está no fato de que populações pequenas não são capazes de prover a diversidade genética necessária para que o algoritmo seja capaz de compor uma solução válida, mesmo com um número grande de interações.

6.1.3. A Inércia das Grandes Massas

Um outro resultado bastante interessante merece destaque nos testes sem adaptação. Utilizando os dados obtidos no teste da trilha da formiga, foram feitos 2 gráficos adicionais, mostrados na figura 15. No primeiro são marcados os valores da *fitness* dos melhores indivíduos presentes nas 7 populações em cada geração. Neste gráfico, 1 geração de uma população de 500 indivíduos é considerada igual a 1 geração da população de 32000.

Porém, como o custo computacional para processar 500 indivíduos é totalmente diferente do custo para processar 32000, foi feito um segundo gráfico no qual a escala de tempo (eixo horizontal) deixa de ser o número de gerações e passa a ser o número de indivíduos processados. Para isso, basta multiplicar o número da geração pelo tamanho da população, em cada uma das 7 curvas.

Agora, comparando os 2 gráficos, observa-se que ocorre uma inversão na ordem das curvas. No gráfico superior da figura 15 podemos ver que, em 100 gerações, a qualidade do melhor indivíduo produzido pela população de 32000 é maior do que na de 16000, que é maior do que na de 8000, e assim por diante.

Em contrapartida, quando analisamos esta característica com respeito ao custo computacional produzido, verificamos que a ordem das curvas se inverte; ou seja, populações menores se antecipam às populações maiores, produzindo resultados mais rapidamente. Em outras palavras, o tempo de resposta das populações menores é

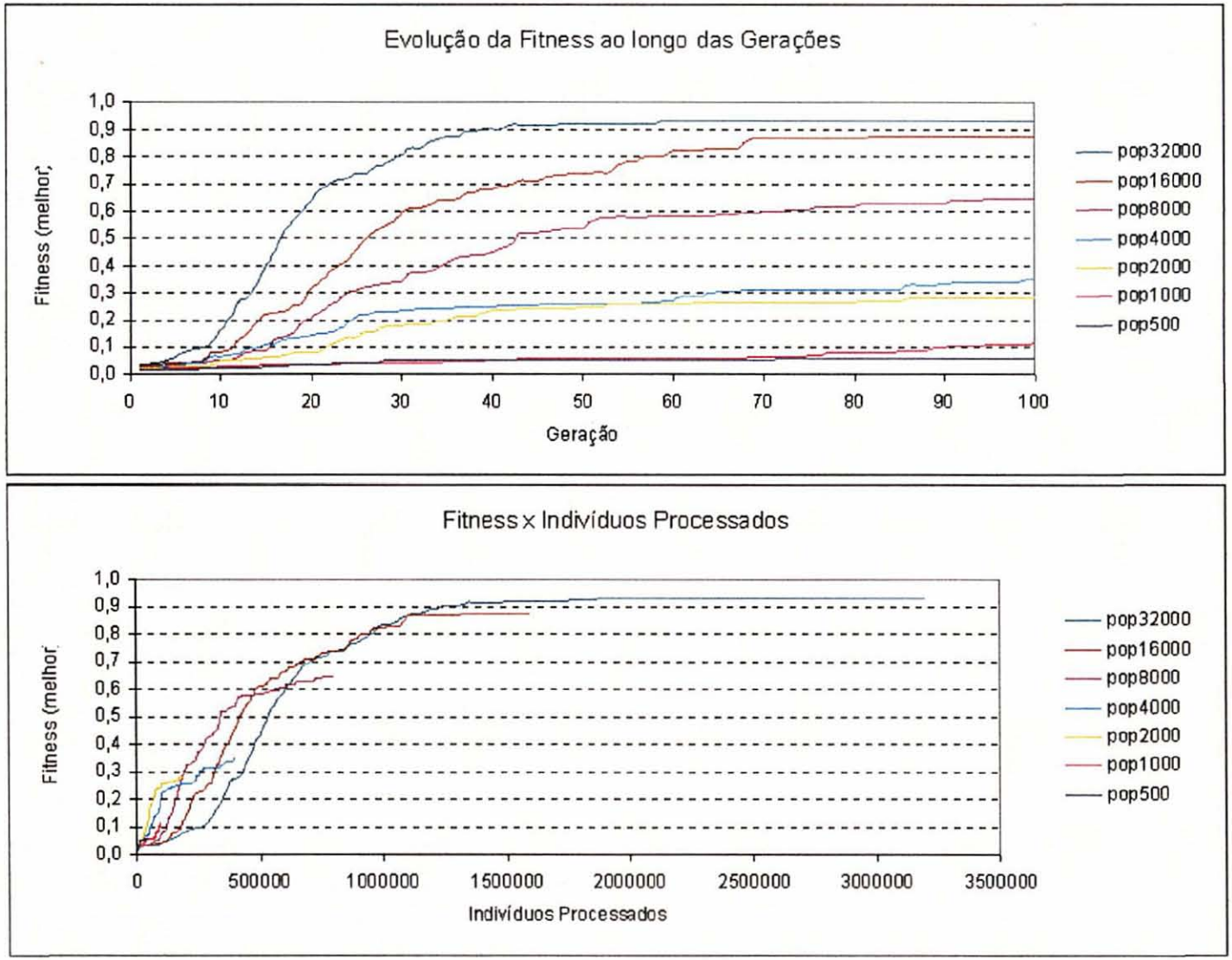
também menor, obviamente porque elas precisam processar um número menor de indivíduos a cada geração.

Então por que usar populações grandes, se as pequenas produzem resultados mais rapidamente? Esta pergunta pode ser respondida com uma análise mais cuidadosa do segundo gráfico. Observando a parte final de cada curva, percebemos que, passado um certo tempo inicial de acomodação, as populações menores tendem a ser ultrapassadas pelas populações maiores. Isto ocorre porque como as populações maiores possuem maior diversidade genética, nelas é possível compor indivíduos que se aproximem mais da solução ideal; ou seja, indivíduos que terão um melhor valor de *fitness*.

Este tipo de comportamento lembra o conceito de inércia, em que um objeto de maior massa precisa de mais força para ser retirado do repouso. Em compensação, uma vez em movimento, ele é capaz de manter sua velocidade por mais tempo, vencendo a força contrária de atrito com maior facilidade.

Com o tamanho da população ocorre um fenômeno semelhante. Populações pequenas atingem rapidamente bons níveis de qualidade, enquanto populações maiores demoram mais para produzir resultados. Por outro lado, populações pequenas tendem à estagnação em níveis de qualidade mais baixos do que as populações maiores. As menores, por serem pequenas demais, não são capazes de lapidar com tanta definição as soluções, pois seu espaço de busca é limitado.

FIGURA 15 - A INÉRCIA DAS GRANDES MASSAS - TRILHA DA FORMIGA, SEM ADAPTAÇÃO, USANDO PARÂMETROS DO KOZA 2



6.2. RESULTADOS COM CONTROLE DO TAMANHO DA POPULAÇÃO

A partir deste momento, a ferramenta *lil-gp* passou a ser comandada pelo módulo de controle do tamanho da população. Em vez de 7 populações, temos agora apenas 1 população ativa em cada momento. Por este motivo, o gráfico principal mostra agora apenas 1 curva, que representa a taxa de acerto da população ativa em cada instante. Esta curva única é, na realidade, a média de 50 curvas obtidas experimentalmente nas 50 execuções realizadas para cada teste.

Um segundo gráfico em forma de pizza nos permite saber qual foi a participação de cada um dos 7 tamanhos de população na descoberta da solução. Em outras palavras, ele calcula, para as execuções válidas, o percentual de vezes em que a solução foi encontrada pela população de 500, 1000, 2000, 4000, 8000, 16000 e 32000.

Informações adicionais indicam ainda quantas das 50 execuções foram consideradas válidas. Execuções válidas são aquelas que obtiveram sucesso antes do critério de parada; isto é, não ultrapassaram o limite estabelecido da maior população conter no máximo 32000 indivíduos. Como, ao contrário dos testes sem adaptação, não existe neste caso um limite máximo de gerações, foi necessário incluir algum tipo de condição de parada para evitar que o algoritmo continuasse infinitamente criando populações de 64000, 128000 etc. Vale lembrar que nos problemas sem adaptação também existia o conceito de rodadas válidas e inválidas. A diferença é que naquele caso o critério de parada era o número máximo de gerações (100) e agora o critério de parada é o tamanho máximo de população (32000).

É muito importante ressaltar que, para a construção do gráfico principal, foram consideradas todas as execuções realizadas, válidas e inválidas. Por este motivo o gráfico de probabilidade de sucesso não atinge 100 % de acerto, pois em alguns casos o algoritmo foi interrompido no momento em que ele tentou criar uma população de 64000 indivíduos. Se esta condição de parada não existisse, o gráfico

tenderia a 100 % de acerto em um número infinito de indivíduos processados.

A distinção entre execuções válidas e inválidas foi feita apenas para a construção do gráfico menor, que mostra a participação de cada população na escolha da solução. Isto foi feito pois não é possível dizer, para as execuções inválidas, qual foi a população que encontrou a solução, uma vez que nas execuções inválidas a solução não foi encontrada devido a uma limitação imposta por uma condição de parada.

Além disso, as informações adicionais também mostram os números das gerações em que a solução foi encontrada. Esta informação não agrega muito valor à análise, pois no processo de controle do tamanho da população o termo "número da geração" ganha uma nova interpretação. Continua existindo o número da geração tradicional, que é contado para cada população desde o início da execução de cada uma. Porém, passa a existir agora um "número de geração geral", que é único e representa o número total de gerações processadas desde o início, independente das trocas de população que ocorrem durante o processo.

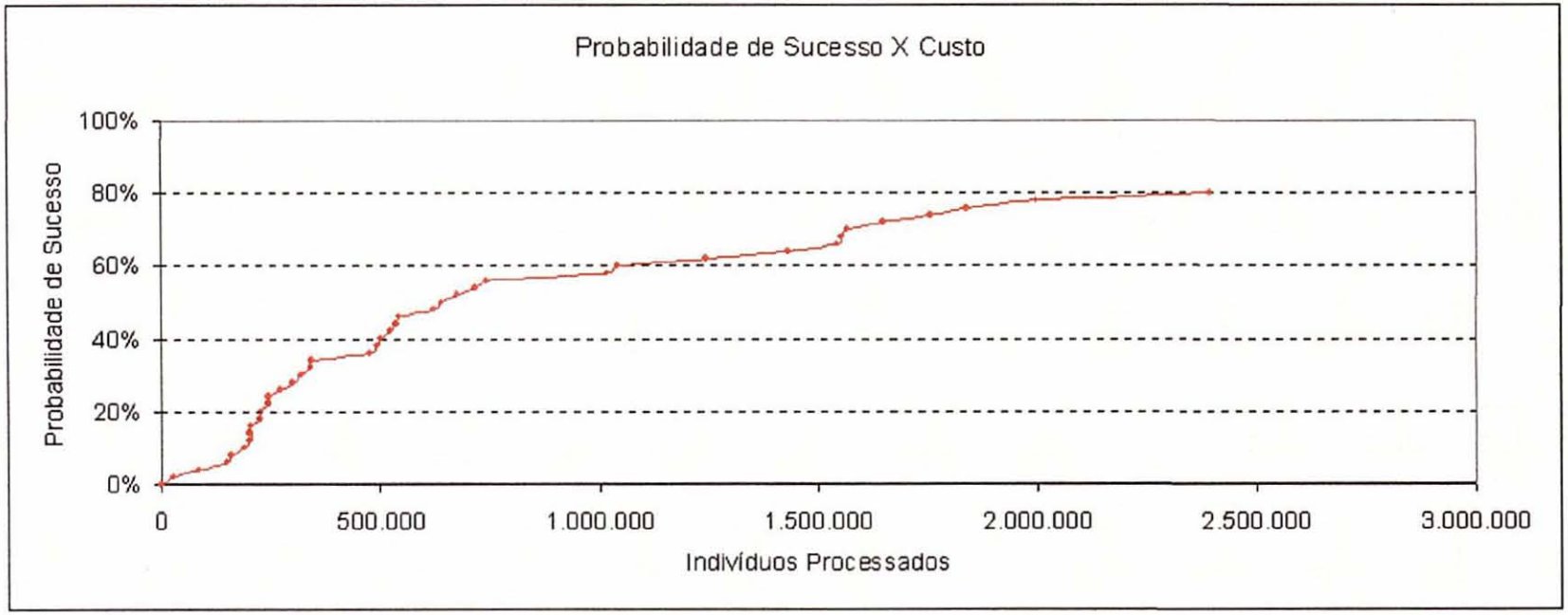
6.2.1. Como Interpretar os Gráficos

O gráfico da probabilidade de sucesso continua tendo o mesmo significado dos testes sem adaptação. Como as unidades utilizadas também são as mesmas, é possível estabelecer uma comparação direta entre eles.

A partir do segundo gráfico é possível determinar que tamanho de população está sendo mais adequado para resolver o problema em questão. Também é possível verificar que as populações pequenas demais não são capazes de resolver o problema na maior parte das vezes, e populações grandes demais demoram mais para chegar a uma solução pois são executadas um número menor de vezes.

A seguir são apresentados todos os gráficos obtidos com o controle do tamanho da população para cada problema e cada conjunto de parâmetros, exatamente na mesma ordem dos gráficos já mostrados para os experimentos sem adaptação.

FIGURA 16 - GRÁFICOS DO EXPERIMENTO DA TRILHA DA FORMIGA, COM ADAPTAÇÃO, USANDO PARÂMETROS DO KOZA 2



Total de runs executados	50
Runs válidos (não atingiram limite)	40

Nos runs válidos:

Geração em que foi encontrada a solução

Primeira	9
Média	77
Última	509

Geração Geral em que foi encontrada a solução

Primeira	37
Média	496
Última	1663

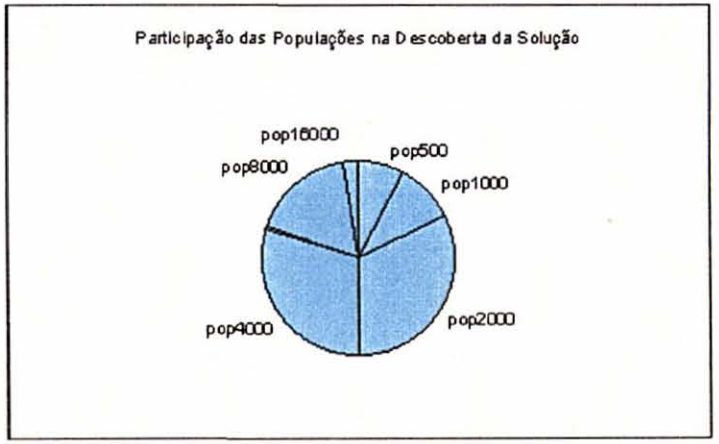
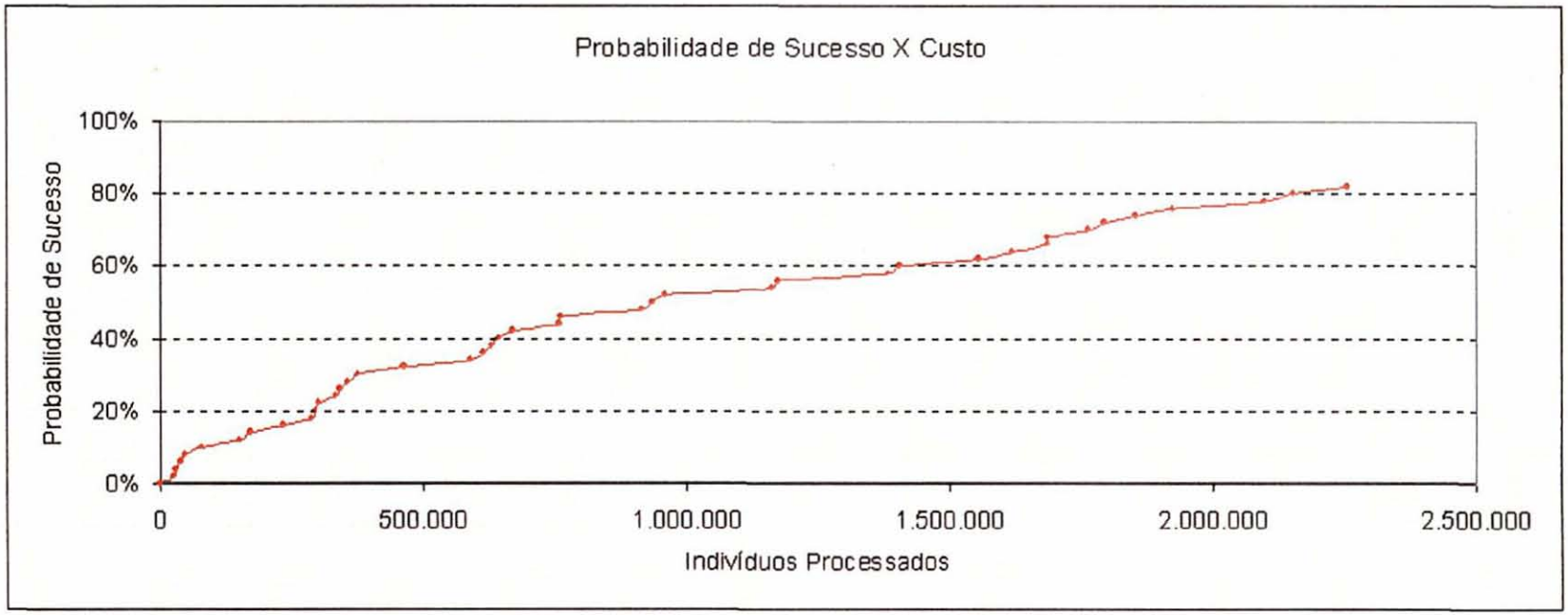


FIGURA 17 - GRÁFICOS DO EXPERIMENTO DA TRILHA DA FORMIGA, COM ADAPTAÇÃO, USANDO PARÂMETROS DO LOBO



Total de runs executados	50
Runs válidos (não atingiram limite)	41

Nos runs válidos:

Geração em que foi encontrada a solução

Primeira	18
Média	77
Ultima	355

Geração Geral em que foi encontrada a solução

Primeira	38
Média	611
Ultima	2325

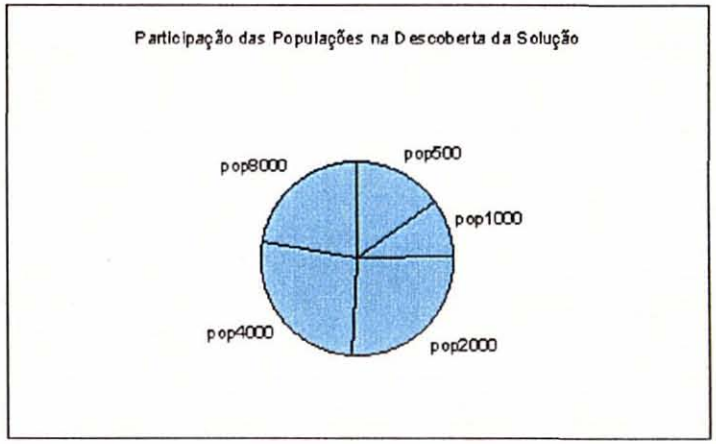
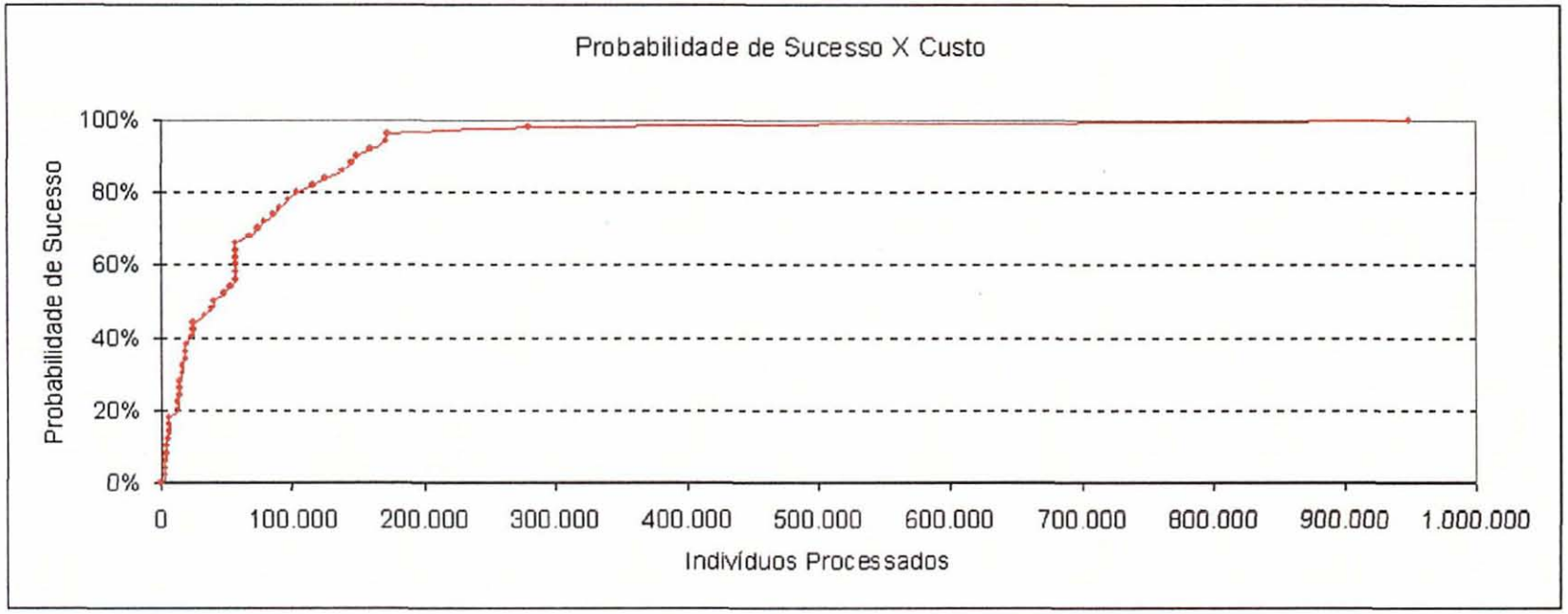


FIGURA 18 - GRÁFICOS DO EXPERIMENTO DA REGRESSÃO DE ORDEM 3 (X^3), COM ADAPTAÇÃO, USANDO PARÂMETROS DO KOZA 2



Total de runs executados	50
Runs válidos (não atingiram limite)	50

Nos runs válidos:

Geração em que foi encontrada a solução

Primeira	2
Média	8
Última	26

Geração Geral em que foi encontrada a solução

Primeira	5
Média	103
Última	1192

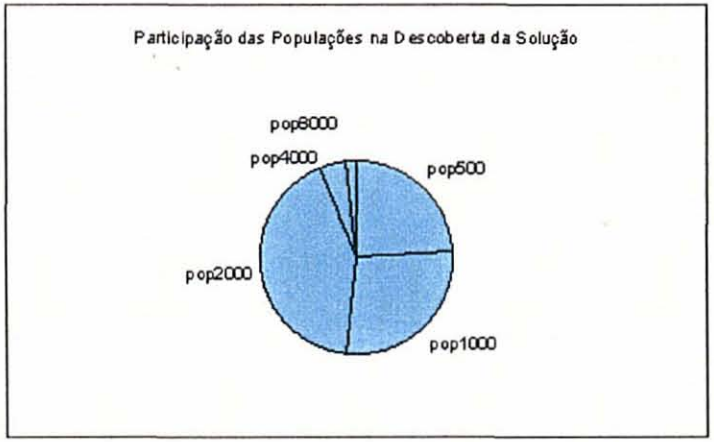
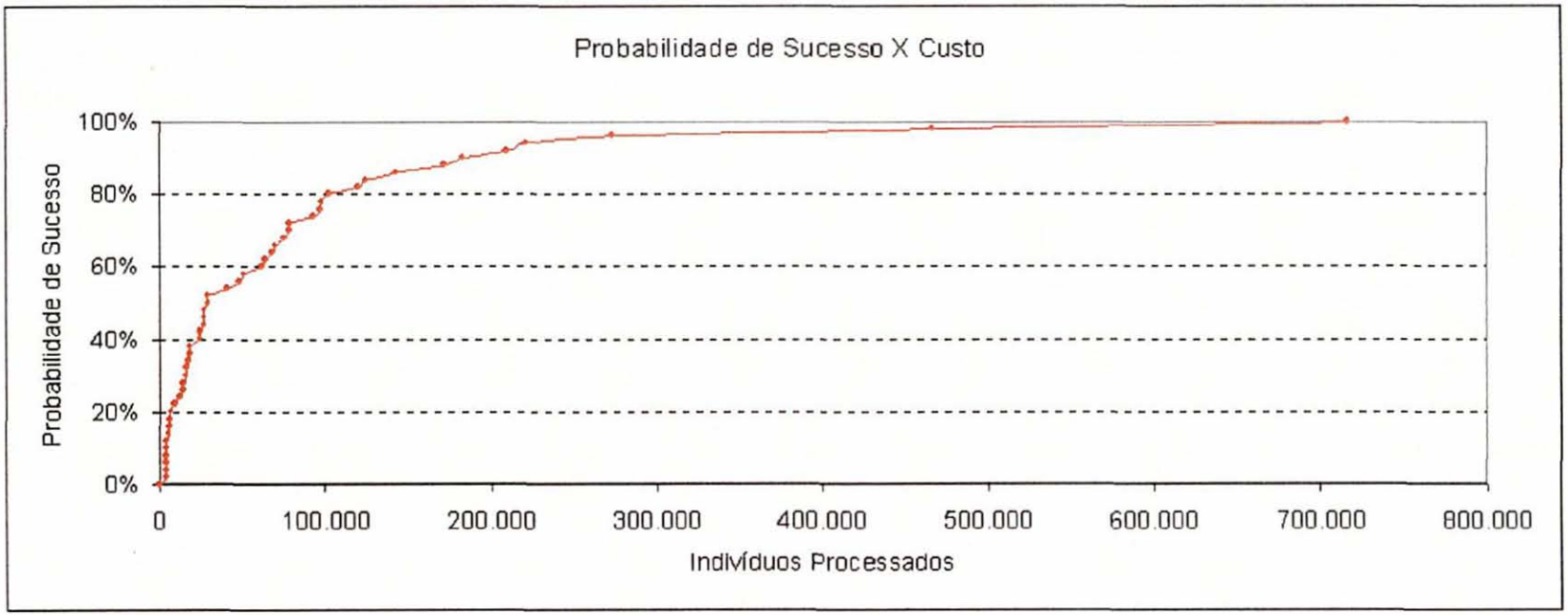


FIGURA 19 - GRÁFICOS DO EXPERIMENTO DA REGRESSÃO DE ORDEM 3 (X^3), COM ADAPTAÇÃO, USANDO PARÂMETROS DO LOBO



Total de runs executados	50
Runs válidos (não atingiram limite)	50

Nos runs válidos:

Geração em que foi encontrada a solução

Primeira	5
Média	13
Ultima	129

Geração Geral em que foi encontrada a solução

Primeira	6
Média	93
Ultima	663

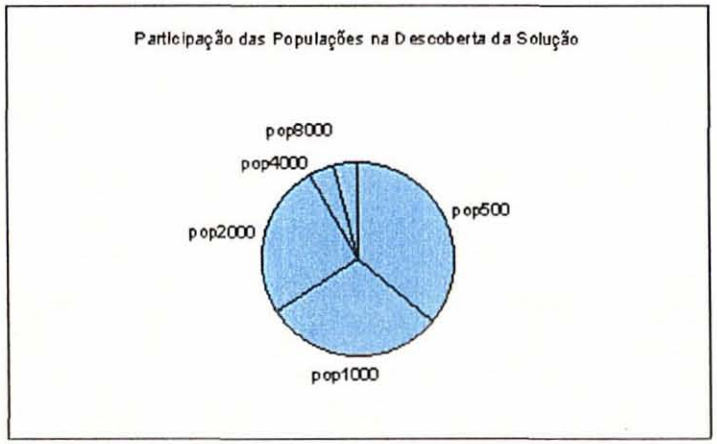
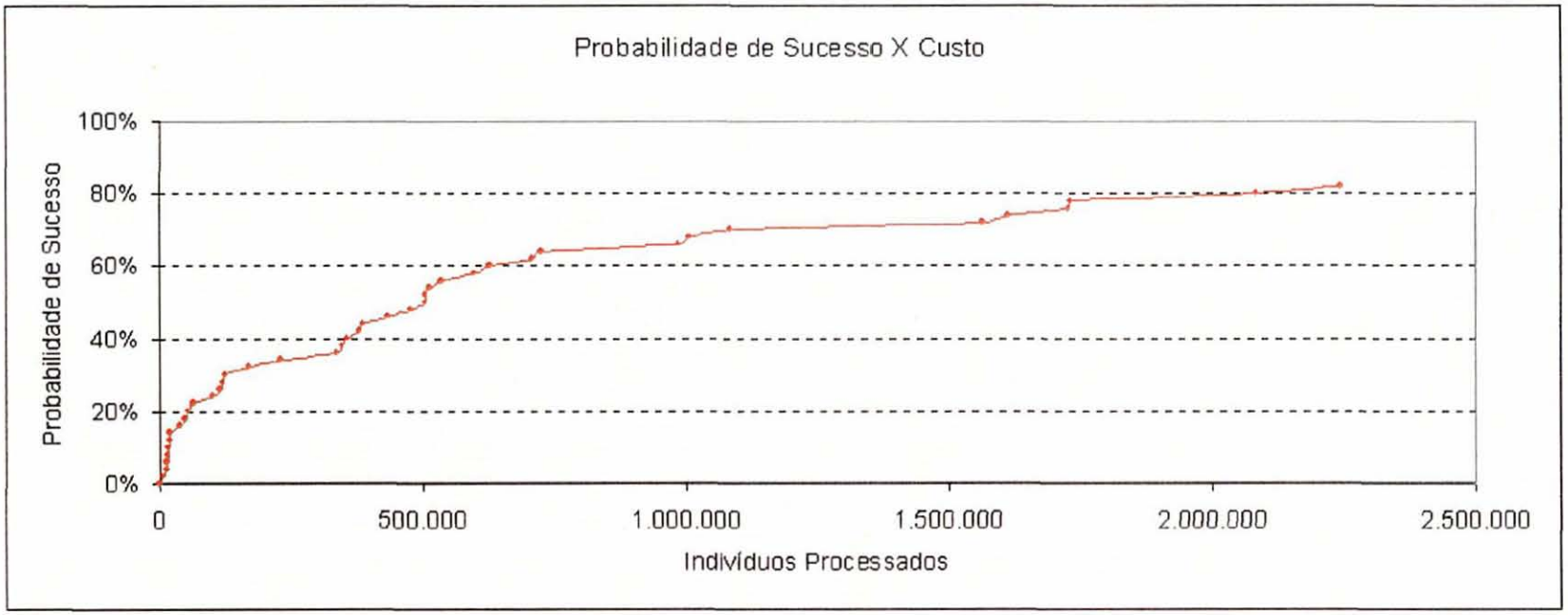


FIGURA 20 - GRÁFICOS DO EXPERIMENTO DA REGRESSÃO DE ORDEM 4 (X^4), COM ADAPTAÇÃO, USANDO PARÂMETROS DO KOZA 2



Total de runs executados	50
Runs válidos (não atingiram limite)	41

Nos runs válidos:

Geração em que foi encontrada a solução

Primeira	5
Média	13
Ultima	102

Geração Geral em que foi encontrada a solução

Primeira	9
Média	451
Ultima	2816

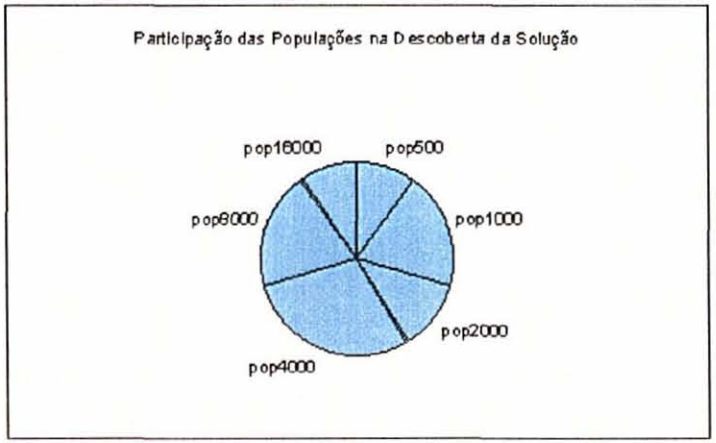
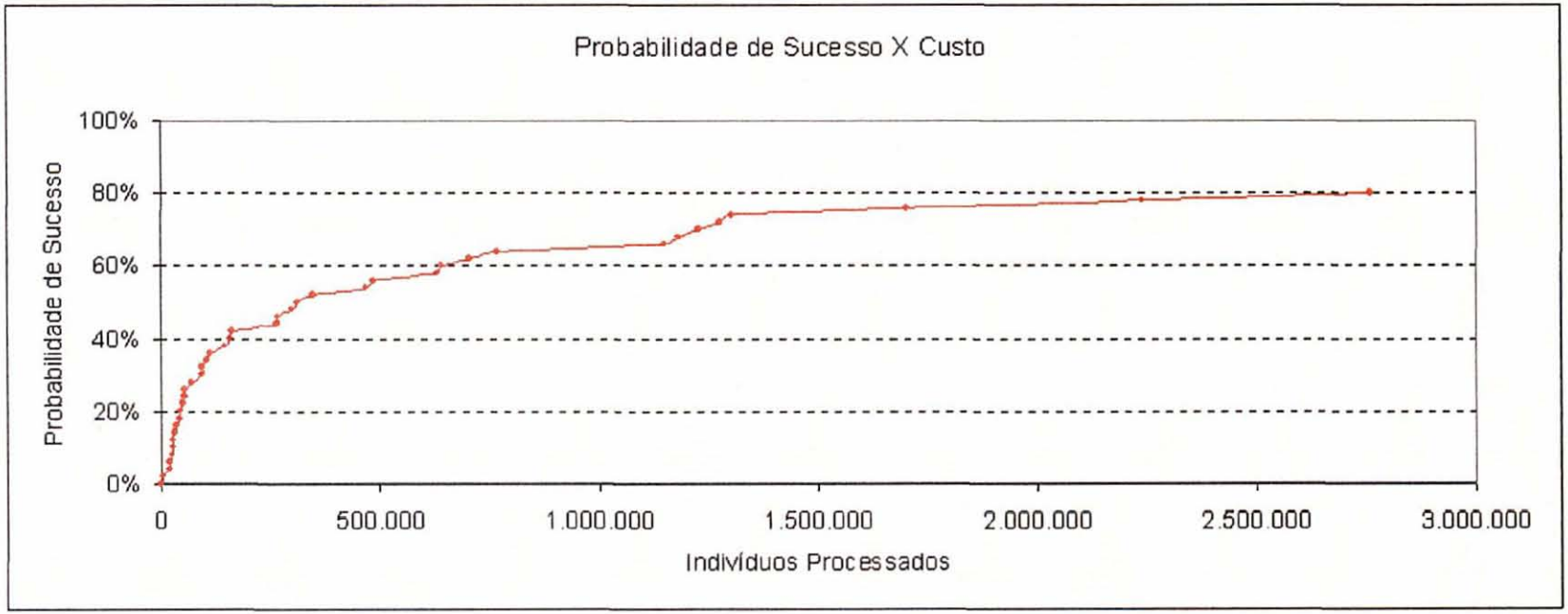


FIGURA 21 - GRÁFICOS DO EXPERIMENTO DA REGRESSÃO DE ORDEM 4 (X^4), COM ADAPTAÇÃO, USANDO PARÂMETROS DO LOBO



Total de runs executados	50
Runs válidos (não atingiram limite)	40

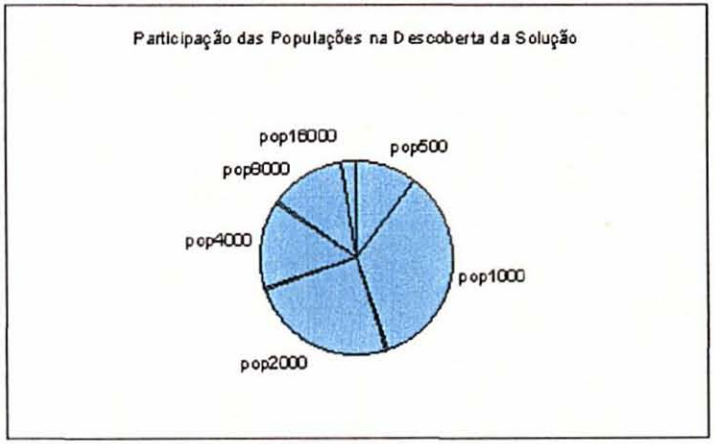
Nos runs válidos:

Geração em que foi encontrada a solução

Primeira	5
Média	20
Última	46

Geração Geral em que foi encontrada a solução

Primeira	6
Média	458
Última	2204



6.2.2. Análise dos Resultados com Adaptação

Com relação ao nível de dificuldade, observa-se que o problema da regressão simbólica de ordem 3 continuou sendo o mais simples, com sucesso em 100 % das execuções. Por outro lado, a diferença entre os outros 2 experimentos diminuiu, e ambos atingiram o mesmo índice de acerto, com pequena diferença na curvatura do gráfico.

A alteração na taxa de cruzamento e no tamanho do torneio, conforme recomendado no trabalho de Lobo [Lobo 2000], produziu um resultado bastante interessante. A curva de probabilidade de sucesso permaneceu praticamente a mesma, não sendo possível detectar mudanças significativas que possam ser justificadas pela mudança nos parâmetros. Entretanto, no gráfico que mostra a participação percentual de cada população na descoberta da solução, pode-se observar uma alteração importante. Com a substituição dos parâmetros padrão [Koza 1994] pelos parâmetros calculados com base no Teorema do Esquema [Lobo 2000], cresceu a participação das populações menores, enquanto as maiores tiveram seu percentual reduzido. Isto significa que com os parâmetros propostos por Lobo, o algoritmo foi capaz de encontrar soluções com um número menor de indivíduos, diminuindo a necessidade de aumento automático no tamanho da população ao longo do processo. No caso do problema da regressão simbólica, por exemplo, a participação da população de 500 indivíduos aumentou bastante em detrimento da participação da população de 2000.

Este resultado pode ser observado em todos os problemas, em maior ou menor intensidade, o que mostra que a escolha da taxa de cruzamento e do tamanho do torneio feita por Lobo com base no Teorema do Esquema realmente teve um impacto positivo sobre o processo, melhorando a atuação das populações menores.

Este fato pode ser entendido por meio da análise da taxa de crescimento dos esquemas, apresentada no capítulo 4. Seu cálculo depende dos valores da taxa de seleção (s), aqui representada pelo tamanho do torneio, e da taxa de cruzamento (pc),

como mostra a seguinte equação:

$$\frac{m(H,t+1)}{m(H,t)} = s.(1-p_c)$$

Utilizando os parâmetros de Koza, teremos uma taxa de crescimento igual a 7.(1-0,9) = 0,7. Com os parâmetros propostos por Lobo, o resultado é 4.(1-0,5) = 2, valor que o autor considerou razoável para garantir o crescimento dos esquemas [Lobo 2000]. Desta forma, é possível justificar o desempenho superior dos parâmetros de Lobo, uma vez que houve um aumento real na taxa de crescimento dos esquemas, permitindo que eles sobrevivessem na população e continuassem participando da formação de possíveis soluções.

Ainda sobre estes dois parâmetros, faz-se necessária uma análise mais detalhada a respeito da magnitude das mudanças dos valores padrão [Koza 1994] para aqueles indicados por Lobo.

O tamanho do torneio representa o número de indivíduos que são escolhidos aleatoriamente para fazerem parte de um subgrupo, do qual será selecionado o melhor deles. Assim, o valor deste parâmetro pode variar de 1 até o tamanho da população, caso extremo em que sempre se escolheria o melhor indivíduo. Estamos trabalhando com populações de 500 a 32000 indivíduos, portanto, a mudança do tamanho do torneio de 7 (Koza2) para 4 (Lobo) é relativamente pequena, se comparada com a ordem de grandeza do tamanho da população. A alteração relativa máxima não chega a 1%, como mostra a tabela 5.

TABELA 5 - ALTERAÇÃO RELATIVA NO TAMANHO DO TORNEIO E NA TAXA DE CRUZAMENTO

Parâmetro alterado	Valor mínimo possível	Valor máximo possível	Alteração absoluta	Alteração relativa
Tamanho do torneio	1	500	ABS (4-7) = 3	3 / (500 -1) = **0,60 %**
	1	1000		3 / (1000 -1) = **0,30 %**
	1	2000		3 / (2000 -1) = **0,15 %**
	1	4000		3 / (4000 -1) = **0,08 %**
	1	8000		3 / (8000 -1) = **0,04 %**
	1	16000		3 / (16000 -1) = **0,02 %**
	1	32000		3 / (32000 -1) = **0,01 %**
Taxa de cruzamento	0	1,00	ABS (0,50-0,90) = 0,40	0,40 / (1,00 -0) = **40 %**

A taxa de cruzamento, por sua vez, foi reduzida de 0,90 (Koza2) para 0,50 (Lobo), uma alteração relativa de 40 %; ou seja, uma mudança 66 vezes maior do que a maior alteração relativa feita no tamanho do torneio. Isto nos permite concluir que, independentemente do significado de cada parâmetro, é muito mais provável que as mudanças observadas (do gráfico 8 para o 9, do 10 para o 11 e do 12 para o 13) tenham sido ocasionadas pela significativa alteração da taxa de cruzamento, e não pela pequena variação sofrida pelo tamanho do torneio.

A partir destas evidências, fica claramente demonstrada a importância da taxa de cruzamento e a necessidade de estudos teóricos e experimentais que determinem mais claramente o impacto deste parâmetro no funcionamento do algoritmo. Entretanto, uma análise mais profunda a este respeito foge ao escopo deste trabalho.

Com relação ao gráfico de probabilidade de sucesso, que mostra a evolução do algoritmo na busca da solução, cabe aqui uma observação importante. Ao comparar este gráfico com e sem adaptação automática, observa-se que o número de indivíduos processados é muito maior quando se faz a adaptação. Esta diferença ocorre porque no algoritmo com adaptação existe uma competição entre populações de diferentes

tamanhos. Neste processo, algumas delas são eliminadas o que faz com que todo o investimento em termos de tempo de processamento seja perdido. Em outras palavras, processar indivíduos daquela população foi uma perda de tempo. Entretanto, não é possível saber a priori que populações serão eliminadas, mesmo porque, se esta informação fosse conhecida, não haveria razão para fazer a adaptação automática deste parâmetro.

Por outro lado, o processo de eliminar populações que não são capazes de resolver o problema traz um benefício muito mais importante do que a eficiência: a confiabilidade. O algoritmo se torna muito mais robusto, pois ele é capaz de perceber que um determinado tamanho de população é inadequado para solucionar aquele problema, substituindo esta população por uma maior.

7. CONCLUSÕES

Algoritmos de Computação Evolucionária aplicam a Teoria da Evolução das Espécies de Charles Darwin na busca da solução em problemas de Inteligência Artificial. As duas principais técnicas, Algoritmos Genéticos e Programação Genética, compartilham a mesma base teórica, porém são aplicadas em diferentes domínios.

O comportamento destes algoritmos é controlado por um conjunto de parâmetros que configuram de que forma a busca é feita, definem limites, e condições de parada. Cada um desses parâmetros precisa ser corretamente configurado no início do processo, e valores iniciais ótimos podem ser diferentes dependendo do problema proposto.

Diversas pesquisas têm sido feitas, principalmente no âmbito dos AG, no sentido de adaptar automaticamente alguns destes parâmetros. Uma das mais recentes, o "AG sem parâmetros" [Lobo 2000], propõe a escolha com base teórica das taxa de cruzamento e seleção e o controle automático do tamanho da população. Como vimos, o tamanho da população é um parâmetros de grande importância e tem forte impacto sobre a probabilidade de sucesso, uma vez que ele define a quantidade de pontos de exploração no espaço de busca.

Neste trabalho, propomos um módulo de controle para PG que aplica o método proposto por Lobo, usando a ferramenta de domínio público *lil-gp*.

Para avaliar os resultados foram feitas diversas baterias de testes com 2 experimentos clássicos freqüentemente utilizados para este fim: o problema da trilha da formiga e o problema da regressão simbólica. Os gráficos obtidos permitem visualizar claramente as mudanças e estabelecer uma comparação direta entre a abordagem de PG tradicional e a abordagem adaptativa.

Nos experimentos realizados com a ferramenta *lil-gp* e 7 diferentes tamanhos de população entre 500 e 32000, percebemos que este é realmente um parâmetro crítico para o algoritmo, que influencia diretamente a qualidade final da solução e a probabilidade de sucesso.

Com a utilização do módulo de controle, percebemos que o algoritmo ficou mais robusto, sendo capaz de perceber a condição de estagnação e eliminar automaticamente populações pequenas demais para o problema em questão. Esta eliminação faz com que o número de indivíduos processados seja maior, pois o processamento de populações que serão futuramente eliminadas é perdido. Mesmo assim, acreditamos que o método de controle do tamanho da população se justifica na medida em que ele garante uma confiabilidade maior de que uma solução será encontrada.

Quanto às mudanças na taxa de cruzamento e no tamanho do torneio sugeridas por Lobo, percebemos que estas alterações favoreceram as populações menores, aumentando sua participação na descoberta da solução. Este fato também contribui para a redução do número de indivíduos processados, porque diminui a necessidade de usar populações maiores no processo. Além disso, percebemos que a alteração relativa feita na taxa de cruzamento é maior do que no tamanho do torneio, o que sugere que as alterações percebidas nos gráficos sejam devidas à mudança do valor da taxa de cruzamento.

Com este trabalho percebemos a importância de utilizar um método de adaptação automática de parâmetros nos algoritmos de Computação Evolucionária, não apenas porque eles facilitam o uso e a aplicação desses algoritmos por usuários que não tenham profundos conhecimentos de AG ou PG, mas principalmente porque faz mais sentido utilizar parâmetros dinâmicos em algoritmos cuja proposta é justamente a evolução. Evoluir parâmetros é, em última análise, corrigir um erro conceitual; o erro de pensar que algoritmos evolutivos devam ser limitados por parâmetros estáticos.

No estudo aqui apresentado, foi possível melhorar a confiabilidade do algoritmo de PG. Não é justo comparar este método com a configuração tradicional feita com parâmetros ótimos, uma vez que eles só podem ser descobertos após testes exaustivos com diversas combinações possíveis.

Em suma, o que deve direcionar as pesquisas nesta área é a preocupação que

devemos ter em tornar os algoritmos de CE mais evolutivos; isto é, (1) mais capazes de modificarem as condições da busca de acordo com necessidades e situações específicas; (2) mais livres do contexto e da definição do problema; (3) menos suscetíveis a falhas por simples limitação do espaço de busca; (4) mais fáceis de serem usados, e, acima de tudo, (5) mais aplicáveis em situações reais.

7.1. PERSPECTIVAS E TRABALHOS FUTUROS

Neste sentido, listamos a seguir apenas algumas das muitas possibilidades e perguntas que surgem a partir do controle automático de parâmetros:

a) Dividir o processo em duas fases: treinamento e execução. Na primeira, os parâmetros ótimos seriam determinados, e na segunda eles seriam utilizados.

b) Utilizar uma população única com tamanho variável. Em vez de várias populações que podem ser eliminadas, apenas uma população cujo tamanho pode variar durante a evolução.

c) Alterar a base do contador, modificando o número de vezes que uma população i precisa ser executada para que uma população i+1 seja criada.

d) Combinar a técnica de controle do tamanho da população [Lobo 2000] com o método de adaptação das taxas dos operadores genéticos [Julstrom 1995].

Acreditamos que estes e outros trabalhos de pesquisa poderão aprimorar os métodos de controle de parâmetros, difundindo e incentivando o uso destas técnicas em algoritmos de Computação Evolucionária.

GLOSSÁRIO

benchmark	critério para avaliação de eficiência
building block	elemento de construção básico ou fundamental
checkpoint	ponto de inspeção ou verificação
fitness cases	casos de treinamento
fitness function	função que mede o grau de aptidão
kernel	parte central, núcleo de um sistema
overtaking	ultrapassar, sobrepujar
string	cadeia de caracteres
threshold	limiar

REFERÊNCIAS

AGUIRRE, H.; TANAKA, K.; SUGIMURA, T.; OSHITA, S. **Cooperative-competitive model for genetic operators**: contributions of extinctive selection and parallel genetic operators. Late Breaking Papers at the 2000 Genetic and Evolutionary Computation Conference. pp. 6-14. Las Vegas, EUA: Morgan Kaufmann, 2000.

ANGELINE, P. J. **Tracking extrema in dynamic environments**. Evolutionary Programming VI. Berlin: Springer, 1997.

BÄCK, T. **Self-adaptation in genetic algorithms**. VARELA, F. J. (Ed.); BOURGINE, P. (Ed.). Toward a practice of autonomous systems. Procedings of the First European Conference on Artificial Life. pp. 263-271. Cambridge, EUA: MIT Press, 1992.

BÄCK, T.; HAMMEL, U.; SCHWEFEL, H. **Evolutionary computation**: comments on the history and current state. IEEE Transactions on Evolutionary Computation. 1. pp. 3-17. [S.l.]: IEEE, 1997.

BAGLEY, J. D. **The behavior of adaptative systems which employ genetic and correlation algorithms**. Tese de Doutorado. UNIVERSITY OF MICHIGAN. Ann Arbor: [s.n.], 1967.

BANZHAF, W.; BANSCHERUS, D.; DITTRICH, P. **Hierarchical genetic programming using local modules**. Proceedings of the International Conference on Complex Systems. Reading, EUA: Addison-Wesley, 1998.

BANZHAF, W.; NORDIN, P. et al. **Genetic programming, an introduction**: on the automatic evolution of computer programs and its applications. San Francisco: Morgan Kaufmann, 1998.

BARBOSA, H. J. C.; SÁ, A. M. **On adaptative operator probabilities in real coded genetic algorithms**. UNIVERSIDADE FEDERAL DO RIO DE JANEIRO. Laboratório Nacional de Computação Científica. Rio de Janeiro: [s.n.], 2000.

DARWIN, C. **On the origin of species by means of natural selection or the preservation of favored races in the struggle for life**. London, Reino Unido: Murray, 1859.

DAVIS, L. **Adapting operator probabilities in genetic algorithms**. Em SCHAFFER, J. D. (Ed.). Procedings of the Third International Conference on Genetic Algorithms (pp. 61-69). San Mateo, EUA: Morgan Kaufmann, 1989.

DERIGS, U.; KABATH, M.; ZILS, M. **Adaptative genetic algorithms**: a methodology for dynamic autoconfiguration of genetic search algorithms. 2nd International Conference on Metaheuristics. Sophia-Antipolis, França: [s.n.], 1997.

EGGERMONT, J.; HEMERT, J. I.; **Adaptative genetic programming applied to new and existing simple regression problems**. Proceedings of EuroGP'2001. 2038. pp. 23-35. Lake Como, Itália: Springer-Verlag, 2001.

EGGERMONT, J.; EIBEN, A. E.; HEMERT, J. I. **Adapting the fitness function in GP for data mining**. Proceedings of EuroGP'99. 1598. pp. 193-202. Goteborg, Sweden: Springer-Verlag, 1999.

EIBEN, A.; HINTERDING, R.; MICHALEWICZ, Z. **Parameter control in evolutionary algorithms**. IEEE Transactions on Evolutionary Computation. 3. pp. 124-141. [S.l.]: IEEE, 1999.

FERNÁNDEZ, F.; TOMASSINI, M.; PUNCH III, W. F.; SÁNCHEZ, J. M. **Experimental study of multipopulation parallel genetic programming**. Third European Conference on Genetic Programming (Poster). Edinburgh, Reino Unido: [s.n], 2000.

FOGEL, L. J.; ANGELINE, P. J.; FOGEL, D. B. **An evolutionary programming approach to self-adaptation on finite state machines**. Proceedings of the Fourth Annual Conference on Evolutionary Programming. pp. 355-365. Cambridge, EUA: MIT Press,1995.

GENETIC programming kernel version 0.5.2 - parameter study. Disponível em: <http://ftp.eecs.umich.edu/people/daida/transfer/gpc/gpkernel_2.html> Acesso em: 06 dez. 2001.

GOLDBERG, D. **Genetic algorithms in search, optimization, and machine learning**. Boston, EUA: Addison-Wesley, 1989.

GREFENSTETTE, J. J. **Optimization of control parameters for genetic algorithms**. SAGE, A. P. (Ed.). IEEE Transactions on Systems, Man and Cybernetics. SMC-16(1). pp. 122-128. New York, EUA: IEEE, 1986.

GRITZ, L. I. **Evolutionary controller synthesis for 3-D character animation**. Tese de Doutorado. THE GEORGE WASHINGTON UNIVERSITY. Department of Electrical Engineering and Computer Science. Washington, EUA: [s.n.], 1999.

HARIK, G.; LOBO, F. **A parameter-less genetic algorithm**. Relatório técnico nº 99009, Illinois Genetic Algorithm Laboratory. Illinois, EUA: [s.n.], 1999.

HOLLAND, J. H. **Adaptation in Natural and Artificial Systems**. Cambridge, EUA: MIT Press, 1992.

IEEE. **Proceedings of the World Congress on Computation Intelligence WCCI 2002**, Congress on Evolutionary Computation. Honolulu, Havaí. IEEE, 2002.

JULSTROM, B. A. **What have you done for me lately**? Adapting operator probabilities in a steady-state genetic algorithm. ESHELMAN, L. J. (Ed.). Proceedings of the 6th International Conference on Genetic Algorithms and their Applications. pp. 81-87. Pittsburgh, EUA: Morgan Kaufmann, 1995.

JULSTROM, B. A. **An inquiry into the behavior of adaptative operator probabilities in steady-state genetic algorithms**. Proceedings of the Second Nordic Workshop on Genetic Algorithms and their Applications. pp. 15-26. Vaasa, Finland: [s.n.], 1996.

JULSTROM, B. A. **Adaptative operator probabilities in a genetic algorithm that applies three operators**. Em BRYANT B. (Ed.); CARROLL, J. (Ed.) OPPENHEIM, D. (Ed.); HIGHTOWER, J. (Ed.); GEORGE K. M. (Ed.). Proceedings of the 1997 ACM Symposium on Applied Computing. pp. 233-238. New York, EUA: ACM, 1997.

KAELBLING, L. P.; LITTMAN, M. L.; MOORE, A. W. **Reinforcement learning**: a survey. Journal of Artificial Intelligence. pp. 237-285. [S.l.]: Morgan Kaufmann, 1996.

KARGUPTA, H. **Information transmission in genetic algorithm and Shannon's second theorem**. Proceedings of the Fifth International Conference on Genetic Algorithms. pp. 640. [S.l]: Morgan Kaufmann, 1993.

KOCH, T.; SCHEER, V. et al. **A parallel, hybrid meta optimization for finding better parameters of an evolution strategy in real world optimization problems**. Proceedings of the 2000 Genetic and Evolutionary Computation Conference Workshop Program. pp. 17-19. Las Vegas, EUA: Morgan Kaufmann, 2000.

KOZA, J. R. **Genetic programming**: on the programming of computers by means of natural selection. Cambridge, EUA: MIT Press, 1992.

KOZA, J. R. **Genetic programming II**: automatic discovery of reusable programs. Cambridge, EUA: MIT Press, 1994.

KOZA, J. R. **Genetic Programming III**: Darwinian Invention and Problem Solving. San Francisco, EUA: Morgan Kaufmann, 1999.

LANGDON, W. B.; POLI, R. **Why ants are hard**. Genetic Programming 1998: Proceedings of the Third Annual Conference. pp. 193-201. Madison, EUA: Morgan Kaufmann, 1998.

LIANG, K.; YAO, X.; NEWTON, C. **Dynamic control of adaptative parameters in evolutionary programming**. Simulated Evolution and Learning, Second Asia-Pacific Conference on Simulated Evolution and Learning, SEAL'98. pp. 42-29. Canberra, Australia: Springer, 1999.

LOBO, F. G.; GOLDBERG, D. E. **Decision making in a hybrid genetic algorithm**. Relatório técnico nº 96009. Illinois Genetic Algorithm Laboratory. Illinois, EUA: [s.n.], 1996.

LOBO, F. G. **The parameter-less genetic algorithm**: rational and automated parameter selection for simplified genetic algorithm operation. Tese de Doutorado. UNIVERSIDADE NOVA DE LISBOA. Faculdade de Ciências e Tecnologia. Lisboa: [s.n.], 2000.

MERCER, R. E.; SAMPSON, J. R. **Adaptative search using a reproductive meta-plan**. Kybernetes. 7. pp. 215-228. [S.l.: s.n.], 1978.

MILLER, B.; GOLDBERG, D. **Genetic algorithms, selection schemes, and the varying effects of noise**. [S.l.: s.n.], 1995.

MILLER, B.; GOLDBERG, D. **Genetic algorithms, tournament selection, and the effects of noise**. Relatório técnico nº 95006. Illinois Genetic Algorithm Laboratory. Illinois, EUA: [s.n.], 1995.

MITCHELL, M. **An introduction to genetic algorithms**. Cambridge: MIT Press, 1996.

NIEHAUS, J.; BANZHAF, W. **Adaption of operator probabilities in genetic programming**. TOMASSINI, J. (Ed.); LANZI, P. (Ed.); RYAN, C. (Ed.); LANGDON, W. B. (Ed.) Procedings of 4th EuroGP Conference. pp. 325-336. Berlin: Springer, 2001.

O'NEILL, M.; RYAN, C. **Evolving multi-line compilable c programs**. [S.l.: s.n., ca 2000].

ROSS, B. **Logic-based genetic programming with definite clause translation grammars**. Relatório técnico nº CS-99-02. BROCK UNIVERSITY. Department of Computer Science. Ontario, Canadá: [s.n.], 1999.

RYAN, C.; COLLINS, J. J.; O'NEIL, M. **Grammatical evolution**: evolving programs for and arbitrary language. Proceedings of the First European Workshop on Genetic Programming. pp. 83-95. Paris, França: Springer-Verlag, 1998.

SMITH, R. E.; SMUDA, E. **Adaptively resizing populations**: algorithm, analysis, and first results. Complex Systems. 9. pp 47-72. [S.l.: s.n.], 1996.

SPECTOR, L.; GOODMAN, E.D.; WU, A.; LANGDON, W. B.; VOIGT, H.-M.; GEN, M.; SEN, S.; DORINGO, M.; PEZESHK, S.; GARZON, M. H.; BURKE, E. **Proceedings of the Genetic and Evolutionary Computation Conference**. San Francisco: Morgan Kauffman, 2001.

TUSON, A. L. **Adapting operator probabilities in genetic algorithms**. Tese de Mestrado. UNIVERSITY OF EDINBURGH. Department of Artificial Intelligence. Edinburgh, Reino Unido: [s.n.], 1995.

WEINBERG, R. **Computer simulation of a living cell**. Dissertations Abstracts International. 31 (9). 5312B. [S.l.: s.n.], 1970.

WILLIS, M. J.; HIDEN, H. G.; MARENBACH, P.;MONTAGUE, G.A. **Genetic programming**: an introduction and survey of applications. Proceedings of the 2nd International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications. pp. 314-319. London, Reino Unido: IEEE, 1997.

XIAO, J.; MICHALEWICZ, Z; ZHANG, L.; TROJANOWSKI, K. **Adaptative evolutionary planner/navigator for mobile robot**. IEEE Transactions on Evolutionary Computation. 1 (1). pp. 18-28. [S.l.]: IEEE, 1997.

ZONGKER, D.; PUNCH, B. **Lil-gp 1.01 user's manual**. MICHIGAN STATE UNIVERSITY. Disponível em: <http://garage.cps.msu.edu/software/lil-gp/lilgp-index.html> Acesso em: 16 abr. 2002.