EDUARDO OLIVEIRA COSTA

PROPOSTA DE UM ALGORITMO DE PROGRAMAÇÃO GENÉTICA BASEADO EM ESTRATÉGIAS EVOLUCIONÁRIAS

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Informática no Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, Universidade Federal do Paraná.
Orientadora: Profa. Dra. Aurora Trinidad Ramirez Pozo

CURITIBA

2006

EDUARDO OLIVEIRA COSTA

PROPOSTA DE UM ALGORITMO DE PROGRAMAÇÃO GENÉTICA BASEADO EM ESTRATÉGIAS EVOLUCIONÁRIAS

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Informática no Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, Universidade Federal do Paraná.
Orientadora: Profa. Dra. Aurora Trinidad Ramirez Pozo

CURITIBA

2006

Agradecimentos

A minha família cujo apoio e carinho foram fundamentais para esta conquista.

À minha orientadora, Aurora Trinidad Ramirez Pozo, pelos inestimáveis conselhos, pela grande ajuda e paciência, e principalmente por toda a confiança depositada na minha pessoa ao longo desses vários anos.

À Luzia Vidal e Leonardo Ramos Emmendorfer, pelo auxílio prestado.

Aos meus amigos que sempre me apoiaram nos momentos difíceis.

Aos colegas de Mestrado, funcionários e professores do Departamento de Informática da UFPR.

A todos aqueles que de alguma forma participaram do desenvolvimento deste trabalho.

SUMÁRIO

LISTA DE FIGURAS **iv**

LISTA DE TABELAS **v**

Resumo **vii**

Abstract **viii**

1 Introdução **1**
1.1 Motivação 1
1.2 Objetivos 3

2 Programação Genética e Estratégias Evolucionárias **5**
2.1 Programação Genética 5
2.1.1 Elementos da Programação Genética 6
2.1.1.1 Estrutura de Programas 6
2.1.1.2 População Inicial 7
2.1.1.3 Função de Avaliação (Fitness) 8
2.1.1.4 Seleção 9
2.1.1.5 Operadores Genéticos 9
2.1.1.6 Parâmetros Genéticos 10
2.1.2 Algoritmo Básico de Programação Genética 11
2.2 Estratégias Evolucionárias 12

3 Proposta do Algoritmo $(\mu + \lambda) - PG$ **16**
3.1 Motivações 16
3.2 Algoritmo Proposto 17
3.3 Aspectos de Implementação 17
3.4 Validação 21

4 Binomial–3 **23**
4.1 Fundamentação Teórica 23
4.2 Experimentos 24
4.3 Resultados 25

5 Séries Temporais **28**
5.1 Fundamentação Teórica 28
5.1.1 Previsão de Séries Temporais 30
5.1.2 Métodos de Previsão 30

5.1.3 Métodos de Decomposição de Séries Temporais 31
5.1.4 Métodos Simples de Previsão de Séries Temporais 32
5.1.5 Métodos Avançados de Previsão de Séries Temporais 33
5.2 Experimentos 35
5.3 Resultados 36

6 Modelagem da Confiabilidade de *Software* 39
6.1 Fundamentação Teórica 39
6.2 Experimentos 41
6.2.1 Experimento 1 41
6.2.1.1 Conjuntos de Dados 41
6.2.1.2 Pré–processamento 41
6.2.1.3 Avaliação dos Modelos 43
6.2.2 Experimento 2 44
6.2.2.1 Conjuntos de Dados 45
6.2.2.2 Avaliação dos Modelos 45
6.3 Resultados 46
6.3.1 Experimento 1 46
6.3.2 Experimento 2 48

7 *Santa Fe Artificial Ant* 51
7.1 Fundamentação Teórica 51
7.2 Experimentos 51
7.3 Resultados 52

8 Conclusões e Trabalhos Futuros 55

Referências Bibliográficas 58

Lista de Figuras

2.1 Exemplo de indivíduo na Programação Genética. 7

2.2 Pais escolhidos randomicamente e ponto de cruzamento. 10

2.3 Filhos gerados após a operação de cruzamento. 10

2.4 Algoritmo básico de Programação Genética. 12

2.5 Algoritmo (1+1)-ES. 14

3.1 Algoritmo $(\mu + \lambda) - PG$ proposto 18

3.2 Arquitetura do Lil-gp 19

3.3 Processo de Seleção 20

3.4 Operador de Mutação 20

3.5 Arquivo de Saída 21

4.1 Binomial-3 – Probabilidade de Sucesso 26

4.2 Binomial-3 – Evolução do fitness médio no conjunto 6 27

5.1 Exemplo de Série Temporal. 29

6.1 Tempo entre falhas (TBF). 42

6.2 Média móvel de 7a. ordem. 43

6.3 Falhas x Tempo 43

7.1 Trilha Santa Fe 52

7.2 *Artificial Ant* – Progresso do processo evolutivo 53

Lista de Tabelas

3.1 Configurações dos Parâmetros 22

4.1 Conjuntos de Constantes Aleatórias (ERCs) 25
4.2 Binomial-3 – Resultados PG Clássica 25
4.3 Binomial-3 – Resultados GP Proposto 25
4.4 Binomial-3 – p-valores resultando do teste t pareado 26

5.1 Séries Temporais - Resultados 37
5.2 Séries Temporais - Resultados (continuação) 38

6.1 Conjuntos de Dados 41
6.2 Confiabilidade - PG Clássica - Experimento 1 (Conjunto de Funções 1) . . 46
6.3 Confiabilidade - PG Clássica - Experimento 1 (Conjunto de Funções 2) . . 47
6.4 Confiabilidade - PG Proposta - Experimento 1 (Conjunto de Funções 1) . . 47
6.5 Confiabilidade (Experimento 1) - p–valores resultando de teste t pareado . 48
6.6 Cobertura - Modelo CBB 49
6.7 Cobertura - PG Clássica 49
6.8 Cobertura - PG Proposta 49
6.9 Confiabilidade (Experimento 2) - p–valores resultando de teste t pareado . 50

7.1 Funções e Terminais - Problema Santa Fé 52
7.2 *Artificial Ant* - Resultados 53
7.3 *Artificial Ant* - Resultados utilizando a mutação original 54

Lista de Abreviaturas

AE - All Edges (Todos-os-arcos)

AN - All Nodes (Todos-os-nós)

ANN - Artificial Neural Networks (Redes Neurais Artificiais)

AR - *AutoRegressive* (Modelo Autoregressivo)

ARIMA - *AutoRegressive Integrate Moving Average* (Modelo Autoregressivo Integrado e de Médias Móveis)

ARMA - *AutoRegressive Moving Average* (Modelo Autoregressivo e de Médias Móveis)

CBB - *Coverage Binomial Based* (Modelo de Cobertura Binomial)

ES - *Evolution Strategies* (Estratégias Evolucionárias)

GEO - Modelo Geométrico

JAM - Modelo Jalinski-Moranda

MA - *Moving Average* (Modelo de Médias Móveis)

MARMA - *Multivariable ARMA* (Modelo ARMA Multivariável)

PG - Programação Genética

PDU - *Potential-Uses-DU-path* (Todos-os-potenciais-DU-caminhos)

PU - *Potential-Uses* (Todos-os-potenciais-usos)

PUDU - *Potential-Uses/DU* (Todos-os-potenciais-usos/DU)

RMSE - *Root Mean Square Error* (Raiz do Erro Médio Quadrático)

TBF - *Time Between Failures* (Tempo entre Falhas)

Resumo

Este trabalho apresenta uma nova abordagem para a indução de programas pela Programação Genética (PG) utilizando as idéias das Estratégias Evolucionárias (ES). A meta deste trabalho é desenvolver uma variação do algoritmo de Programação Genética, realizando alterações no algoritmo clássico e adicionando conceitos da teoria das Estratégias Evolucionárias.

A abordagem proposta é avaliada utilizando problemas de dois domínios diferentes: Problemas de Regressão Simbólica e o Problema da Formiga (*Santa Fe Artificial Ant*). Dentre os problemas de Regressão Simbólica, são estudados os problemas *Binomial*–3, que caracteriza-se como um problema de dificuldade ajustável; Séries Temporais e Modelagem da Confiabilidade de Software.

Os resultados obtidos são comparados com os resultados obtidos com a PG clássica. Para os problemas de Regressão Simbólica obteve-se excelentes resultados e um melhoramento de desempenho significativo foi atingido, entretanto isto não aconteceu com o problema *Santa Fe Artificial Ant*.

Abstract

This work proposes a new approach to the induction of programs by means of Genetic Programming (GP) using ideas of Evolution Strategies (ES). The goal of this work is to develop a variety of Genetic Programming algorithm doing some modifications on the classical GP algorithm and adding some concepts of Evolution Strategies.

The new approach was evaluated using two instances of different domains. Symbolic Regression problems and the *Santa Fe Artificial Ant problem*. The following problems of Symbolic Regression were studied: *Binomial*–3 problem, a problem with tunably difficulty problem, a Time Series problem and the Modelling Software Reliability Growth.

The results found were compared with the classical GP algorithm. For the Symbolic Regression problems excellent results were obtained and a significant improvement was achieved, but this does not happened with the Artificial Ant problem.

Capítulo 1

Introdução

A Programação Genética (PG) é uma técnica de Aprendizado de Máquina que tem se mostrado um eficiente paradigma para resolver problemas em muitas áreas do conhecimento tais como circuitos digitais, *data mining*, biologia molecular, tarefas de otimização e outras [Koza, 1992] [Banzhaf et al., 1998] [Kaboudan, 2000].

Sob o ponto de vista estocástico, a PG explora praticamente e de forma robusta, mas não necessariamente com eficiência, grandes espaços de busca [Koza, 1992]. A PG pode ser usada para "aprender" funções matemáticas ou encontrar padrões em um conjunto de dados, tendo assim um grande campo de aplicabilidade.

1.1 Motivação

Por se tratar de um algoritmo relativamente novo, muitos aspectos sobre seu comportamento estão "em aberto". Em razão disto, muitos estudos [Daida et al., 2001] [Daida, 2005] [Stevens et al., 2005] [Silva and Costa, 2005] podem ser encontrados como propostas de melhoria para o desempenho da Programação Genética e também para a identificação das causas que afetam a habilidade da PG em determinadas situações. Em [Koza, 1992], são encontrados muitos dos principais fatores que causam estes problemas, tais como conjuntos de funções e terminais, métodos de inicialização e tamanho da população. Nesse trabalho, Koza, fornece uma fórmula semi-empírica que estima o número de tentativas necessárias para resolver um problema com uma probabilidade de sucesso específica.

Por outro lado, em [Daida et al., 2001], é realizado um estudo sobre os fatores que fazem com que um problema seja considerado *GP-Hard* e a análise de um problema de dificuldade ajustável na PG. Neste trabalho, o problema *Binomial–3* é apresentado e sua característica estatística é descrita como um problema que pode ser ajustado de um nível mais fácil para outro mais difícil. De acordo com [Daida et al., 2001], este problema adquire complexidade logarítmica em determinadas situações.

Experimentos relacionados a identificação de populações que aumentam a probabilidade de sucesso na PG são conduzidos em [Daida, 2005], utilizando o problema *Binomial–3*. Este trabalho, fornece uma estimativa detalhada do fluxo do material genético através da população, geração a geração, enquanto vários fatores que influenciam a dificuldade do problema são alterados. Também, de acordo com Daida [Daida, 2005], este trabalho tenta descobrir a métrica potencial de uma população inicial que pode ser utilizada para prever o sucesso do processo evolutivo. Algumas das conclusões deste trabalho são que a composição da população inicial pode determinar uma regra significativa que sugere como deve ser tratado um problema pela PG. O uso de seleção por torneio é proposto para aumentar a possibilidade de sucesso e, também, o uso de algoritmos alternativos para inicialização da população e para ajustar a composição dos conjuntos de terminais e funções é apresentado.

Em [Silva and Costa, 2005] uma técnica de controle de código inútil (*bloat control*), que é o excesso de crescimento de código causado por operadores genéticos na busca de soluções melhores, é apresentada utilizando a abordagem de recursos limitados dinâmicos (*Dynamic resource-limited*), que é uma combinação de duas abordagens: a primeira baseia-se no conceito de Árvore Dinâmica de Profundidade Máxima (*Dynamic Maximum Tree Depth*) [Silva and Almeida, 2003] e a segunda Programação Genética com Limitação de Recursos (*Resource-Limited GP*) [Wagner and Michalewicz, 2001] [Silva et al., 2005]. A primeira delas é uma técnica inspirada na técnica tradicional de limite da árvore pela profundidade. A segunda é outra técnica de controle de crescimento de código inútil que impõe limites no nível do indivíduo. Este método foi aplicado ao problema de regressão simbólica e também ao problema de *Santa Fe Artificial Ant*, obtendo resultados excelentes para o

último.

A idéia de múltiplos cruzamentos, é apresentada em [Stevens et al., 2005], para prevenir o crescimento de código inútil. Três abordagens de multi-cruzamento (*multicrossover*) foram definidas e experimentos foram realizados utilizando um problema de regressão simbólica e o problema 0–1–4 [Soule et al., 2002]. O problema 0–1–4 foi designado para facilmente permitir a inserção de íntrons, que são indivíduos que possuem a estrutura $(-XX)$, representados por 0's, sendo o gene uma cadeia de caracteres variável composta por 0's, 1's e 4's. Uma das conclusões deste trabalho é que o crescimento de código pode ser controlado pelo ajuste do número de cruzamentos que acontecem em uma operação de recombinação.

1.2 Objetivos

Tendo como motivação o contexto descrito anteriormente, esta dissertação apresenta uma proposta de uma nova abordagem de algoritmo de Programação Genética baseada nas pesquisas apresentadas em [Daida et al., 2001] [Daida, 2005] [Stevens et al., 2005], juntamente com conceitos da teoria de Estratégias Evolucionárias.

O principal objetivo do presente trabalho é apresentar um novo algoritmo de Programação Genética que seja baseado mais fortemente nas Estratégias Evolucionárias do que na teoria dos Algoritmos Genéticos, proposta por John Holland [Holland, 1992].

Este estudo contribuirá no sentido de aperfeiçoar a utilização da PG na resolução de problemas e também na ampliação de seu potencial agregado à nova abordagem de algoritmo proposta como, por exemplo, na melhoria da capacidade de encontrar uma solução, na minimização de erros de previsão e na modelagem mais precisa para resolução de problemas.

Foram escolhidos dois domínios distintos de problemas para a realização dos experimentos. O primeiro domínio foi o de problemas de Regressão Simbólica. Três problemas foram escolhidos:

- *Binomial–3*, um problema de dificuldade variável, proposto por [Daida et al., 2001];
- Séries Temporais, uma aplicação prática de regressão simbólica;

- Modelagem do crescimento da confiabilidade de *software*, outro exemplo de aplicação prática de regressão simbólica.

O segundo domínio escolhido compreende o problema clássico da Inteligência Artificial, denominado *Santa Fe Artificial Ant* ou Problema da Formiga.

Experimentos foram realizados e os resultados das duas abordagens (a clássica e a proposta) foram comparados e discutidos.

A estrutura do trabalho está organizada como descrito a seguir. O Capítulo 2 discorre sobre os conceitos de Programação Genética e Estratégias Evolucionárias. O Capítulo 3 apresenta a proposta do novo método de PG. Os Capítulos 4, 5, 6 e 7 descrevem os problemas estudados, experimentos realizados e resultados obtidos. Finalmente, o Capítulo 8 conclui o trabalho e apresenta as possibilidades para trabalhos futuros.

Capítulo 2

Programação Genética e Estratégias Evolucionárias

A Computação Evolucionária é um dos ramos da área da Ciência da Computação denominado Aprendizado de Máquina, que apresenta um novo paradigma para a resolução de problemas baseado nos mecanismos observados na natureza e formalizado na teoria da Evolução Natural das Espécies, proposta por Charles Darwin [Darwin, 1859].

As principais técnicas que pertencem a Computação Evolucionária são Algoritmos Genéticos [Holland, 1992], Estratégias Evolucionárias [Bäck et al., 2003], Programação Genética [Koza, 1992] e Programação Evolucionária [Bäck et al., 2003].

2.1 Programação Genética

O paradigma da Programação Genética (PG) foi proposto por John R. Koza (1992). Segundo, [Barone and cols., 2003], o objetivo principal da Programação Genética é prover uma forma de indução de programas usando os princípios da Teoria da Evolução.

Nesta técnica, programas de computador são criados e, aplicando-se o princípio Darwiniano de sobrevivência dos mais adaptados e operadores de cruzamento genético (recombinação), tais indivíduos evoluem. As operações são realizadas diretamente em programas de computadores que podem variar em forma e tamanho. Assim, pode-se resumir a tarefa como a busca pelo programa de computador mais adaptado no espaço de

todos os possíveis [Barone and cols., 2003]. A Programação Genética hoje é reconhecida como um paradigma efetivo de pesquisa em Inteligência Artificial e Aprendizado de Máquina, Classificação, Robótica e muitas outras áreas [Kaboudan, 2000], [Koza, 1992], [Banzhaf et al., 1998].

2.1.1 Elementos da Programação Genética

2.1.1.1 Estrutura de Programas

O conjunto de possíveis estruturas na Programação Genética é formado pelo conjunto de possíveis composições, a partir de um conjunto de n funções disponíveis, F = {f1, f2, f3,..., fn} e de um conjunto de m terminais disponíveis, T = {t1, t2, t3, ..., tm} [Barone and cols., 2003].

O conjunto de terminais contém as entradas do programa a ser obtido. O conjunto de funções é composto pelos operadores disponíveis no sistema de Programação Genética. Supondo que o programa deve calcular a seguinte expressão:

$$(a + b) - (a * b) \tag{2.1}$$

O conjunto de funções deve ser $\{+, -, *\}$ e o conjunto de terminais deve conter $\{a, b\}$.

A escolha dos conjuntos de funções e terminais deve ser analisada criteriosamente, pois influencia diretamente na solução apresentada. Se no conjunto F existirem poucos operadores, provavelmente não será obtida uma boa solução para o problema. Por outro lado, disponibilizando muitos operadores, o programa poderá ficar extenso, provocando maior esforço computacional e também produzindo expressões indecifráveis pelo ser humano.

O ideal neste caso é ir adicionando os operadores por etapas. Primeiramente incluir operadores básicos: soma, adição, subtração, divisão. Após, incluir outros operadores como, por exemplo: exponenciação, raiz quadrada, logaritmo e assim sucessivamente. Tal cuidado deve ser tomado também para a escolha de terminais, que devem representar somente as entradas para a função.

A estrutura de dados mais freqüentemente utilizada em Programação Genética é a

árvore. Cada árvore representa um indivíduo na população. A Figura 2.1 exemplifica um indivíduo gerado para o programa acima:

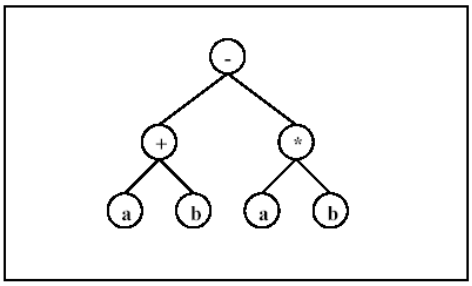

Figura 2.1: Exemplo de indivíduo na Programação Genética.

Segundo [Barone and cols., 2003], programas em programação genética podem ser representados por expressões LISP. Embora a técnica não esteja restrita a ela, o paradigma funcional de linguagem facilita a sua utilização. Várias outras características da linguagem LISP, tais como a facilidade da representação hierárquica dos programas e a possibilidade utilização de qualquer expressão como função justificam o seu uso em aplicações baseadas nessa técnica. De acordo com [Barone and cols., 2003], dois requisitos devem ser satisfeitos na escolha do conjunto de funções e terminais aplicáveis ao problema:

- Fechamento: os indivíduos em uma população devem ser compostos por funções e terminais que executem corretamente, sem erros.

- Suficiência: o conjunto de funções e terminais escolhidos deve ser capaz de compor uma solução para o problema.

2.1.1.2 População Inicial

Normalmente, a geração da população inicial é completamente aleatória. Em particular, na Programação Genética, devem ser respeitados os dois requisitos Fechamento e Suficiência na geração dos indivíduos. O tamanho da população é um parâmetro genético de extrema importância. Com uma população pequena não haverá grande variabilidade genética, o que, dependendo da evolução, poderá causar estagnação do processo evolutivo. Já uma população grande demais, poderá tornar o algoritmo extremamente lento. Existem diversos métodos para inicializar uma população. Os mais comuns são:

- *Grow*: os nós são selecionados aleatoriamente nos conjuntos T e F (exceto a raiz, que é retirada do conjunto F). Produz árvores de formatos irregulares [Luke and Painait, 2001].

- *Full*: Escolhe somente funções até que um nó de profundidade máxima seja selecionado, então ele passa a escolher somente terminais [Banzhaf et al., 1998]. Cada árvore atinge a profundidade máxima.

- *Ramped Half-and-Half*: Utiliza o método *Full* 50% das vezes e o método *Grow* nas outras 50%. Tem como objetivo gerar um número igual de árvores para cada profundidade [Koza, 1992].

- *Random-Branch*: Neste método é informado o tamanho máximo da árvore, em vez de se informar a profundidade máxima [Chellapilla, 1997]. O valor do tamanho da árvore é dividido entre as árvores de um nó-pai não terminal. Isso possibilita que muitas árvores impossíveis de serem construídas sejam geradas.

- Uniform: Cria árvores uniformemente, do conjunto de todas as árvores possíveis [Bohm and Geyer-Schulz, 1996]. O algoritmo calcula várias vezes quantas árvores poderão ser geradas para cada tamanho desejado, fazendo com que este método possua alto custo computacional.

2.1.1.3 Função de Avaliação (Fitness)

Segundo [Goldberg, 1989], o valor da função de *fitness* pode ser pensado como uma medida de lucratividade, utilidade e qualidade que queira se maximizar. Usando conceitos biológicos, o valor da função de *fitness*, é associado ao seu grau de resistência e adaptabilidade do indivíduo ao meio onde vive. Assim, indivíduos com maior *fitness* terão uma chance maior de sobreviver e serão responsáveis pela próxima geração. De acordo com [Barone and cols., 2003], a definição do *fitness* está ligada ao domínio do problema a ser resolvido.

Na literatura, quatro tipos de *fitness* podem ser encontrados como mais utilizados [Barone and cols., 2003]:

- Básico (*Raw Fitness*): é definido diretamente em função do domínio do problema.

- Padronizado (*Standardized Fitness*): está relacionado com o *fitness* básico e o modi-

fica, de forma que um valor mais baixo seja sempre o melhor.

- Ajustado: é utilizado somente em casos onde uma resposta exata é exigida para um problema.

- Normalizado (ou Proporcional): é usado normalmente para definição da probabilidade de um indivíduo ser selecionado para a próxima geração ou para alguma operação genética.

2.1.1.4 Seleção

Segundo a Teoria da Evolução de Darwin, os indivíduos passam por processos de seleção natural, onde os mais adaptados sobrevivem. Na Programação Genética, esse grau de adaptação é medido pelo *fitness* o que possibilita a seleção de alguns indivíduos para a aplicação de operadores genéticos e conseqüente prosseguimento no processo evolutivo. Existem vários métodos de se realizar a seleção. Dentre eles destacam-se [Goldberg, 1989]:

- Roleta (*Roulette Wheel*): cada indivíduo recebe um valor que analogamente seria sua porção na roleta. Este valor é uma proporção entre o seu fitness e a soma dos *fitness* da população. Isso faz com que indivíduos com maior *fitness* tenham maior chance de ser escolhidos. Entretanto, nesse método, existe a possibilidade de que o melhor indivíduo não seja escolhido para a próxima geração, pois sua probabilidade de ser selecionado não é 100%. Para que isto não ocorra, pode-se utilizar a Estratégia Elitista [Michalewicz and Schoemauer, 1996], onde uma porcentagem da população com os melhores *fitness* é preservada para a próxima geração automaticamente.

- Integral: respeita rigidamente o *fitness* relativo.

- Torneio: indivíduos são selecionados aleatoriamente e disputam um torneio, no qual o melhor (com *fitness* maior) é selecionado.

- Aleatória: são selecionados N indivíduos da população aleatoriamente.

2.1.1.5 Operadores Genéticos

Os principais operadores genéticos utilizados na Programação Genética são:

- Reprodução: Nesta operação um indivíduo é selecionado conforme seu *fitness* proporcional e é efetuada uma cópia deste indivíduo para a nova geração. Não é efetuada a recombinação genética (reprodução assexuada), fazendo com que indivíduos com alto grau de adaptação sejam selecionados para permanecerem no processo evolutivo [Barone and cols., 2003].

- Cruzamento: São escolhidos dois indivíduos para uma operação sexuada. Em cada um dos pais é escolhido de forma aleatória um ponto de cruzamento (*crossover*). A subárvore existente a partir do ponto de crossover do primeiro pai é inserida no ponto de *crossover* do segundo pai e vice-versa [Barone and cols., 2003]. A seguir pode-se observar um exemplo de cruzamento nas figuras 2.2 e 2.3.

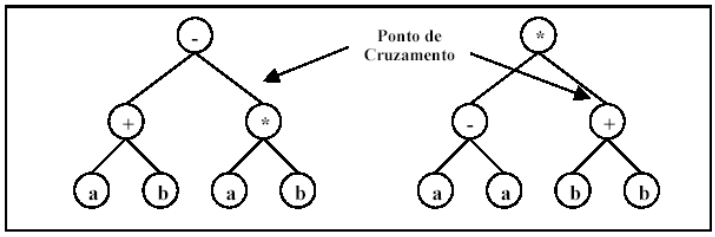

Figura 2.2: Pais escolhidos randomicamente e ponto de cruzamento.

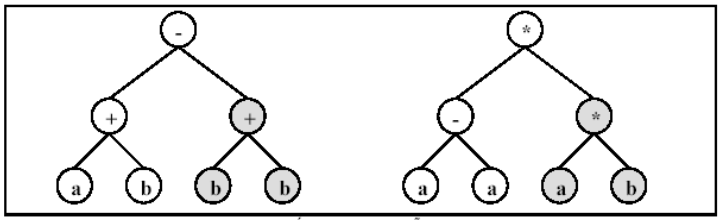

Figura 2.3: Filhos gerados após a operação de cruzamento.

- Mutação: Seleciona um ponto na árvore aleatoriamente e efetua troca da subárvore existente por uma nova árvore gerada randomicamente [Minglei, 2002].

2.1.1.6 Parâmetros Genéticos

Devem ser considerados os seguintes parâmetros genéticos na execução de um algoritmo de Programação Genética [Koza, 1992]:

- Tamanho da População: afeta diretamente o desempenho e eficiência do algoritmo. Uma população muito pequena oferece uma pequena cobertura do espaço de busca. Se a

população for muito grande, tornam-se necessários recursos computacionais maiores, ou um tempo maior de processamento. Logo, deve-se buscar um equilíbro no que diz respeito ao tamanho escolhido para a população.

- Probabilidade de *Crossover*: quanto maior esta probabilidade, mais rapidamente as estruturas serão introduzidas na população. Isto pode gerar um efeito indesejado, pois a maior parte da população será substituída, ocorrendo assim a perda da diversidade genética. Com uma taxa baixa, o algoritmo pode se tornar muito lento para oferecer uma resposta adequada.

- Probabilidade de *Mutação*: permite uma opção a mais na variabilidade genética, entretanto uma probabilidade de mutação muito alta pode tornar a busca essencialmente aleatória.

- Número de Gerações: define a quantidade de gerações do processo evolutivo. Deve-se buscar um equilíbrio neste valor, pois um número muito baixo de gerações pode ser ineficiente, não chegando ao resultado esperado ou um número muito alto, pode causar processamento computacional inútil.

2.1.2 Algoritmo Básico de Programação Genética

Na Figura 2.4, pode-se observar o algoritmo básico da Programação Genética, segundo [Barone and cols., 2003].

Inicialmente são criados indivíduos de forma aleatória. Tais indivíduos tendem a exibir um baixo grau de adaptação ao problema. A cada nova geração criada, eles tendem a exibir uma melhor adaptação, devido à pressão exercida pelo *fitness* e às operações de seleção, cruzamento e mutação [Barone and cols., 2003].

O termo adaptação no contexto da Programação Genética pode ser definido como o quão próximo do valor esperado é o resultado obtido pelos indivíduos gerados. Com o andamento do processo evolutivo e a aplicação de operadores genéticos, os indivíduos tendem a serem mais bem adaptados.

Dois pontos importantes a serem ressaltados são a variabilidade dinâmica dos programas, como uma característica importante do paradigma, sendo extremamente difícil

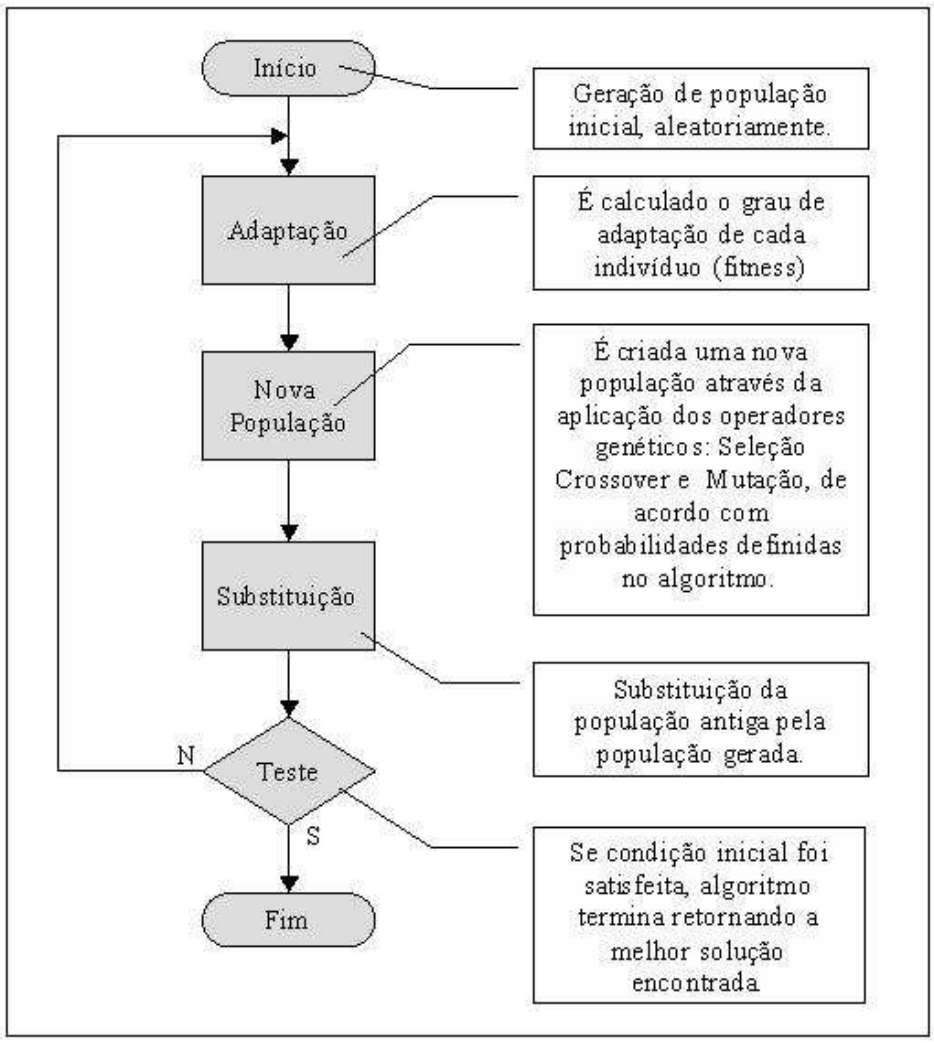

Figura 2.4: Algoritmo básico de Programação Genética.

restringir o tamanho e forma das soluções, o que poderia diminuir a chance de se encontrar boas soluções. Segundo, é a não necessidade de pré-processamento de entradas ou pós-processamento de saídas, uma vez que entradas, resultados intermediários e saídas são expressos diretamente em termos de instruções ou valores relativos ao domínio do problema [Barone and cols., 2003].

2.2 Estratégias Evolucionárias

De acordo com [Kusiak, 2000], Estratégia Evolucionária (*Evolution Strategies* – ES) é um algoritmo no qual indivíduos são codificados por um conjunto de variáveis de valores reais, o "genoma". As Estratégias Evolucionárias foram inicialmente desenvolvidas com o

propósito de otimização de parâmetros [Dianati et al., 2002].

O primeiro algoritmo de ES foi apresentado em 1964 na Universidade Técnica de Berlim. A idéia era imitar os princípios da evolução orgânica em otimização de parâmetros para aplicações tais como *pipe bending* ou controle de PID para um sistema não-linear [Dianati et al., 2002].

Uma ES é caracterizada pelo tamanho da população, número de descendentes produzidos em cada geração e se a nova população é selecionada de pais e descendentes ou somente de descendentes [Kusiak, 2000].

O primeiro algoritmo de Estratégia Evolucionária proposto utilizava um esquema mutação-seleção conhecido como *two-membered* ES ou $(1+1)-ES$ [Dianati et al., 2002]. Segundo [Bäck et al., 2003], o $(1+1)-ES$ trabalha com um indivíduo, que cria um descendente por meio de mutação, sendo que o melhor entre os dois é selecionado deterministicamente para integrar a próxima geração.

Para obter a probabilidade de mutação ótima deste esquema, Rechenberg calculou as taxas de convergência de duas funções modelo e seus desvio-padrão ótimos para mutações de sucesso [Dianati et al., 2002]. A partir disso surgiu a Regra de Sucesso 1/5, proposta por Rechenberg [Rechenberg, 1973], que determina que: *A proporção de mutações de sucesso para todas as mutações deve ser* $1/5$*, devendo ser incrementada ou decrementada a variância até obter este valor.* Na Figura 2.5 é mostrado o Algoritmo (1+1)-ES, extraído de [Dianati et al., 2002].

Rechenberg [Rechenberg, 1973], propôs o *multimembered* ES onde $\mu > 1$ pais participam na geração de 1 descendente. A notação utilizada foi $(\mu+1)-ES$. Neste método, todos os pais têm as mesmas probabilidades de casamento e, como no *two-membered* ES, o membro de menor *fitness* da população, incluindo todos os pais e um descendente, é eliminado em cada geração [Dianati et al., 2002].

O $(\mu+1)-ES$ não é uma estratégia muito utilizada, mas direcionou para avanços propostos por Schwefel em 1975 [Schwefel, 1975] [Schwefel, 1977] [Schwefel, 1981], para habilitar a auto-adaptação de parâmetros como o desvio-padrão para mutações. Os algoritmos $(\mu+1)-ES$, segundo [Dianati et al., 2002], implementaram auto-adaptação

INÍCIO

1. Escolha um único vetor pai que contém m parâmetros

$$\mathbf{X = \{x_1, x_2, ..., x_m\}}$$

Cada parâmetro é escolhido por um processo aleatório e satisfaz as restrições do problema.

2. Crie um novo descendente por meio de mutação. Para obter a mutação nesse método, adicione um vetor aleatório do tamanho de X com distribuição normal (média = 0 e variância = σ):

$$\mathbf{X' = X + N\ (0, \sigma)}$$

3. Compare as soluções para X e X'. Escolha o melhor membro para a próxima geração.

4. Repita os passos 2 e 3 até encontrar uma solução satisfatória, ou até que tenha atingido o número máximo de gerações.

FIM

Figura 2.5: Algoritmo (1+1)-ES.

submetendo os parâmetros de evolução (desvio-padrão de mutações) no processo de criação de novos indivíduos.

O $(\mu+1)-ES$ especifica que μ pais produzem λ descendentes ($\lambda > \mu$). Os descendentes competem com seus pais na seleção dos μ melhores indivíduos para a criação da próxima geração. Este procedimento apresenta problemas com ótimos locais e, para solucioná-los criou-se o $(\mu, \lambda)-ES$, onde o tempo de vida de cada indivíduo é de somente uma geração. Recentes estudos sugerem que o último algoritmo é tão bom ou melhor que o primeiro em aplicações práticas [Dianati et al., 2002].

A notação de um algoritmo ES pode ser visualizada abaixo:

$$(\mu + \lambda) - ES \quad ou \quad (\mu, \lambda) - ES$$

Onde:

μ: é o tamanho da população.

$+/,$: O operador "+" indica que μ pais e λ descendentes competirão para a formação da próxima geração. Os μ melhores indivíduos serão selecionados. O operador "," indica que os μ melhores indivíduos serão selecionados somente entre os descendentes.

λ: é o número de descendentes de cada geração.

Capítulo 3

Proposta do Algoritmo $(\mu + \lambda) - PG$

3.1 Motivações

Uma das motivações para propor esta nova abordagem de PG é a pesquisa apresentada em [Daida, 2005]. Neste trabalho, Daida realiza experimentos na tentativa de identificar uma métrica potencial em populações que aumente a probabilidade de sucesso nos problemas de Programação Genética. Uma de suas conclusões é que somente *pouco* material genético criado na população inicial resta ao final do processo evolutivo.

A idéia de *multicrossover* para prevenir a ocorrência de crescimento de código inútil, apresentado em [Stevens et al., 2005], também forneceu subsídios para formular o algoritmo proposto. A hipótese fundamentada através desta proposta de implementação é que muitas operações de cruzamento entre os melhores indivíduos poderiam evitar a perda de bom material genético e também aumentar a probabilidade de gerar bons indivíduos em potencial.

Além disto, neste trabalho, um estudo preliminar propondo alterações no operador de mutação, na tentativa de evitar geração de código desnecessário é realizado. O operador de mutação original, proposto por [Koza, 1992], somente seleciona aleatoriamente uma sub-árvore e a troca por outra escolhida aleatoriamente. Considerando que em algumas situações, as árvores contêm código desnecessário, a eliminação de uma sub-árvore poderia ser uma boa operação para o processo evolutivo.

3.2 Algoritmo Proposto

Considerando os fatos apresentados, é proposta a seleção de uma pequena porcentagem dos melhores indivíduos criados a cada geração, para serem aplicados os operadores genéticos. Neste passo, é utilizada a seleção elitista, escolhendo os $n\%$ melhores indivíduos.

Após esta etapa, os operadores genéticos (*crossover* e reprodução) são aplicados até que a população retorne ao tamanho original. Esta abordagem foi baseada no algoritmo $(\mu + \lambda) - ES$. Os descendentes competirão com os pais na seleção dos μ melhores indivíduos para a próxima geração [Dianati et al., 2002]. A seleção dos indivíduos neste estágio foi realizada utilizando o método da Roleta (*Roulette Wheel*) com seleção através do fitness proporcional [Goldberg, 1989].

O próximo passo é a aplicação do operador de mutação. A mutação é aplicada com algumas modificações. O operador de mutação original é preservado, sendo adicionada mais uma opção que é a remoção de uma sub-árvore, escolhida aleatoriamente. Estas duas opções de operadores são escolhidas de forma aleatória. De um modo similar ao passo anterior, os candidados à mutação são selecionados utilizando o método da Roleta.

Esta proposta de algoritmo possibilita uma seleção mais competitiva. Poucos pais com material genético promissor são selecionados e há uma alta taxa de reprodução entre eles, o que ocasiona uma grande troca de material genético de boa qualidade. Em contrapartida, no método atual de PG, a recombinação é realizada entre os pais da população, havendo grande probabilidade de haver uma perda de bom material genético através da recombinação. A Figura 3.1, apresenta o algoritmo $(\mu + \lambda) - PG$ proposto.

3.3 Aspectos de Implementação

Foi utilizado o *software* Lil-gp [Zongker and Punch, 1995] como base para a implementação do algoritmo proposto e também para utilização nos experimentos com o método de Programação Genética clássico. O Lil-gp por ser uma ferramenta livre e com código aberto e também possuir as funcionalidades necessárias para o desenvolvimento de algoritmos de Programação Genética é uma escolha interessante.

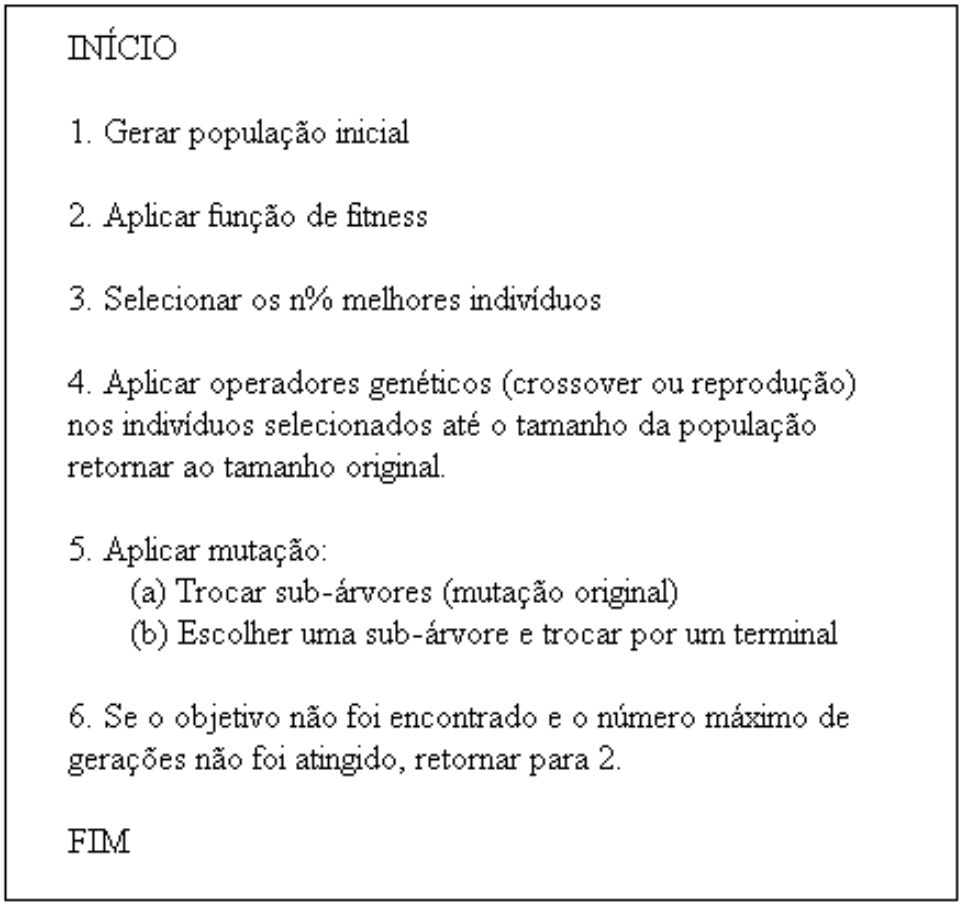

Figura 3.1: Algoritmo $(\mu + \lambda) - PG$ proposto

Inicialmente um estudo sobre a arquitetura do sistema Lil-gp foi realizado a fim de conhecer as funcionalidades e potencialidade do mesmo. De posse de tal conhecimento, as alterações necessárias foram esquematizadas para a construção do algoritmo proposto.

As alterações concentraram-se na sua maioria no núcleo do sistema Lil-gp, de modo que qualquer problema pudesse ser implementado em arquivos distintos, facilitando assim a sua manutenção. Na Figura 3.2 encontra-se um esquema com os arquivos contendo os programas alterados no sistema Lil-gp. No bloco contendo os arquivos do núcleo do Lil-gp encontram-se os programas que são responsáveis pelas operações básicas do sistema, tais como: seleção (select.c), reprodução (reproduc.c), *crossover* (crossovr.c) e o programa principal (gp.c). Foram alterados o programa para realização da criação da nova população através de operadores (change.c) e o programa responsável pela mutação (mutate.c). No segundo bloco, encontram-se os programas relativos a aplicação do Lil-gp. São os programas que contêm as regras (conjuntos de funções e terminais, função de

fitness, manipulação de dados de saída) dos problemas criados.

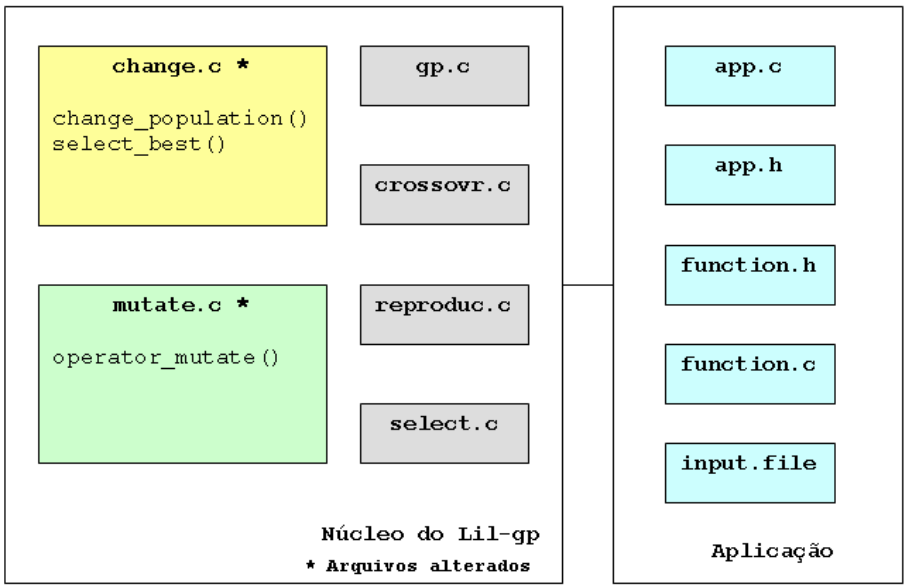

Figura 3.2: Arquitetura do Lil-gp

A primeira alteração importante foi realizada no processo de seleção. O método de seleção original do Lil-gp foi alterado, adicionando a chamada a uma função responsável pela seleção dos $n\%$ melhores indivíduos. Após a seleção dos $n\%$ melhores indíviduos são realizadas operações de recombinação – cruzamento ou reprodução – de acordo com as probabilidades definidas na configuração do algoritmo, até que o tamanho da população fique igual ao original. O processo de seleção funciona conforme a Figura 3.3.

Também foram implementadas alterações no operador de mutação que é mostrado na Figura 3.4. Em termos de implementação, foi criado apenas o mecanismo para substituição da sub-árvore escolhida aleatoriamente por um terminal válido. Também foi implementado um mecanismo de sorteio para a escolha dos dois operadores de mutação. É sorteado sempre 0 ou 1, de modo que os dois operadores têm 50% de chances de acontecerem.

Além das alterações citadas, cada problema estudado foi implementado de forma a possibilitar a geração de saídas automatizadas que pudessem facilitar a análise dos resultados obtidos. Além dos arquivos de saída, gerados automaticamente pelo Lil-gp, foram gerados arquivos de saída contendo estatísticas de cada rodada com dados relevantes tais como: número de gerações que foram processadas, *fitness* do melhor indivíduo de cada

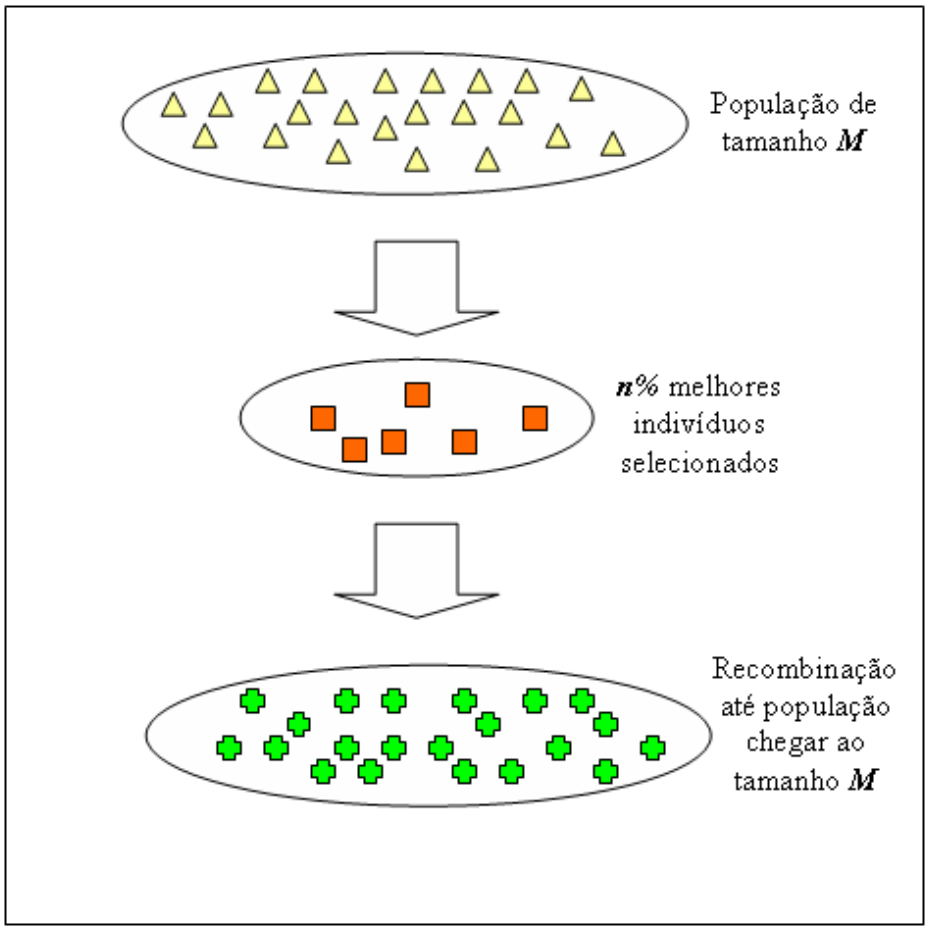

Figura 3.3: Processo de Seleção

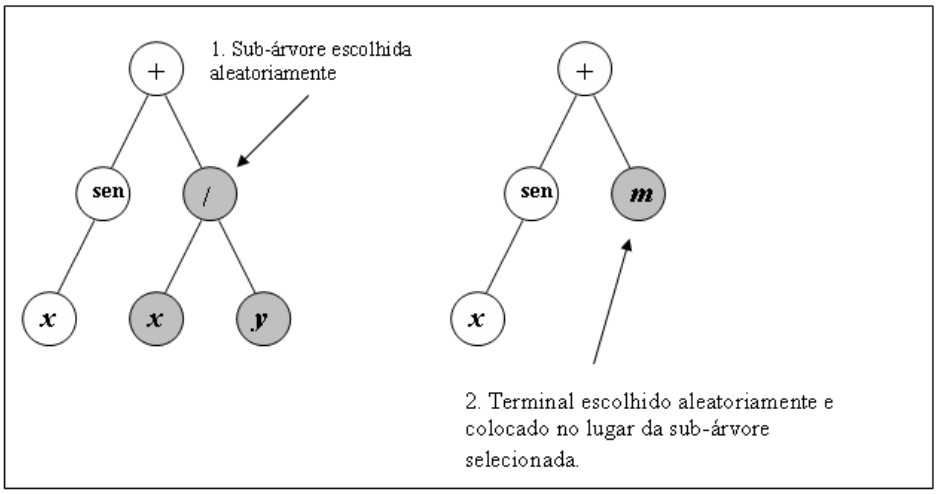

Figura 3.4: Operador de Mutação

geração, porcentagem de sucesso, número de *hits*, entre outros. Ao final de todo o processo um arquivo com estatísticas globais é criado, contendo os dados referentes aos melhores

indivíduos de cada rodada. Na figura 3.5 é mostrado um trecho do arquivo de saída final para ao problema *Binomial*–3.

run	gen	fitness	hits	% success
1	84	0.319443642	38	76.0
2	177	0.095986302	50	100.0
3	40	0.096052492	50	100.0
4	17	0.101728675	50	100.0
5	24	0.186675704	50	100.0
6	117	0.105622491	50	100.0
7	198	0.180300104	50	100.0
8	103	0.152419773	50	100.0
9	13	0.128402177	50	100.0
10	175	0.316071626	41	82.0
11	82	0.235466512	50	100.0
12	164	0.209867580	46	92.0
13	196	0.201182586	43	86.0
14	143	0.173500814	50	100.0

Figura 3.5: Arquivo de Saída

3.4 Validação

Conforme mencionado anteriormente, foram escolhidos dois domínios de problema para a validação do algoritmo apresentado. São eles: Regressão Simbólica, com os problemas: *Binomial*–3, Séries Temporais e Confiabilidade de Software e o Problema *Santa Fe Artificial Ant*.

A configuração dos parâmetros utilizada nos testes do problema *Binomial*–3 foi praticamente a mesma utilizada em [Daida et al., 2001] e para o problema *Santa Fe Artificial Ant* foi a encontrada no Capítulo 7 de [Koza, 1992]. Para os demais problemas, os valores foram definidos a partir de experimentos empíricos realizados com a finalidade de verificar o comportamento da PG sobre esses domínios de problema. Os valores dos parâmetros utilizados na configuração do software Lil-gp estão descritos na Tabela 3.1.

Os experimentos foram todos realizados em um *cluster* utilizando processadores Athlon 64, com 2 GBytes de memória.

Nos próximos capítulos cada problema que foi estudado é descrito, além de uma explanação dos todos os experimentos realizados. Por fim, os resultados obtidos são apresentados juntamente com uma discussão sobre os mesmos.

Tabela 3.1: Configurações dos Parâmetros

Parâmetro	*Binomial-3*	*Séries Temporais*	Confiabilidade	*Artificial Ant*
Número de rodadas	600	90	10 ou 20	600
Gerações	200	200	250	2000
Tamanho da População	500	500	4000	200
Método de Seleção	best	best	best	best
Melhores indivíduos selecionados	20%	20%	20%	20%
Método de inicialização	half_and_half	full	full	full
Profundidade inicial	2-6	2-10	2-10	2-10
Profundidade máxima	26	10	10	10
Número máximo de nós	100	50	50	50
Taxa de *crossover*	80%	80%	70%	80%
Taxa de reprodução	10%	10%	10%	10%
Taxa de mutação	20%	20%	20%	20%

Capítulo 4

Binomial–3

4.1 Fundamentação Teórica

O problema *Binomial–3* foi proposto como um problema de dificuldade ajustável por Daida et. al. em [Daida et al., 2001]. Este problema é um exemplo de problemas de regressão simbólica e é definido pela seguinte equação:

$$f(x) = 1 + 3x + 3x^2 + x^3 \tag{4.1}$$

O termo "binomial" se refere à seqüência dos coeficientes do polinômio e o "3", refere-se a ordem deste polinômio [Daida et al., 2001].

Este problema, também possui muitas propriedades que são comuns a outros problemas na Programação Genética, tais como:

- Permite a escolha de múltiplas abordagens para resolver o mesmo problema;

- Permite muitos tipos de *íntrons*, os quais são indivíduos que contêm a estrutura $(-X, X)$ e possibilitam o crescimento de código inútil (*bloat*).

De acordo com [Daida et al., 1999], o total de maneiras possíveis para se resolver o problema Binomial–3 é da ordem de mil possibilidades. Este problema também foi utilizado em [Daida, 2005], para estudar o efeito do ajuste de atributos de uma população para aumentar a probabilidade de sucesso da Programação Genética.

É definido um valor real α como o parâmetro de ajuste de dificuldade *tuning parameter*.

Variando a faixa através da qual as constantes aleatórias ocorrem, este problema pode ser ajustado de um grau considerado "fácil" até um grau considerado "difícil".

De acordo com [Daida, 2005], em geral, valores de α que são mais distantes de 1, resultam em configurações que aumentam a dificuldade para este problema ser resolvido pela PG.

4.2 Experimentos

Neste trabalho, o problema *Binomial–3* foi definido de acordo com [Daida et al., 2001]. O conjunto de casos de *fitness* compreende 50 pontos eqüidistantes gerados através da equação $f(x) = 1 + 3x + 3x^2 + x^3$, utilizando como valores de x, o intervalo $[-1, 0)$.

A função de *fitness* utilizada é a soma do erro absoluto. Um *hit* é atribuído ao indivíduo, se a diferença entre os valores reais e previstos for ≤ 0.01, sendo 50 o número máximo de *hits* que um indivíduo pode alcançar. O critério de parada do processo evolutivo é quando um indivíduo em uma população obtém 50 *hits*.

O conjunto de funções utilizado é $\{+, -, \div, \times\}$, que corresponde aos operadores aritméticos básicos (sendo, $\div$, a divisão protegida). O conjunto de terminais é definido como $\{x, \beta\}$, onde x é a variável simbólica e β é o conjunto de constantes aleatórias (*ephemeral random constants*- ERCs)

As constantes aleatórias são uniformemente distribuídas através de um intervalo espefífico definido por $[-\alpha, \alpha]$, onde α é um número real que define a faixa para as ERCs. Cada constante randômica foi gerada uma vez no momento da inicialização da população e não foi alterado seu valor durante o processo evolutivo da rodada de PG.

Sete valores de α foram utilizados, são eles: {0.1, 1, 2, 3, 10, 100, 1000}. Uma opção não utilizando ERCs também foi utilizada. Ao todo oito conjuntos de dados foram gerados e são mostrado na Tabela 4.1. Cada conjunto foi treinado usando 600 rodadas, totalizando 4800 rodadas.

Tabela 4.1: Conjuntos de Constantes Aleatórias (ERCs)

Conjunto	*ERC*	*Conjunto*	*ERC*
N	–	3	[-3, 3]
0	[-0.1, 0.1]	4	[-10, 10]
1	[-1, 1]	5	[-100, 100]
2	[-2, 2]	6	[-1000, 1000]

4.3 Resultados

As Tabelas 4.2 e 4.3 apresentam os resultados obtidos com os experimentos nas abordagens clássica e proposta, respectivamente.

Tabela 4.2: Binomial-3 – Resultados PG Clássica

Conjunto de Dados	*Média do Fitness*	*Média de Hits ± Desvio*	*% Sucesso*
N	0,191	46,45 ± 10,44	92,90
0	1,339	22,16 ± 14,26	44,32
1	0,357	40,37 ± 10,96	80,73
2	0,418	37,86 ± 12,12	75,73
3	0,481	36,44 ± 12,42	72,89
4	0,914	27,38 ± 14,55	54,86
5	1,603	15,93 ± 12,49	31,86
6	1,927	14,47 ± 12,22	28,94

Tabela 4.3: Binomial-3 – Resultados GP Proposto

Conjunto de Dados	*Média do Fitness*	*Média de Hits ± Desvio*	*% Sucesso*
N	0,164	46,85 ± 8,67	93,70
0	0,765	30,13 ± 14,29	60,26
1	0,248	44,66 ± 8,11	89,32
2	0,285	43,42 ± 9,15	86,85
3	0,294	45,30 ± 10,33	90,60
4	0,512	36,84 ± 12,22	73,70
5	0,692	33,23 ± 16,16	66,66
6	1,081	25,00 ± 15,12	49,89

De acordo com os resultados apresentados, a probabilidade de sucesso nos experimentos com a nova abordagem de PG é maior do que a obtida com a PG clássica. No conjunto 6 (que possui o maior nível de dificuldade), a diferença entre os métodos é em torno de 40%. Na Figura 4.1 um gráfico comparativo entre as probabilidades de sucesso das duas

abordagens é mostrado.

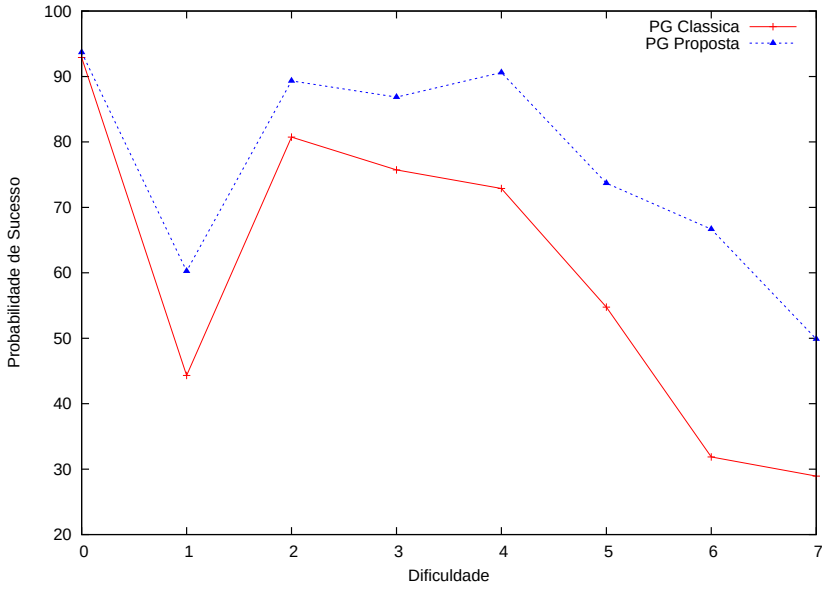

Figura 4.1: Binomial-3 – Probabilidade de Sucesso

Uma comparação entre o método proposto e a PG clássica foi realizada, utilizando teste t pareado, com 95% de nível de confiança. A diferença estatisticamente significativa ($p\text{-}valor < 0.05$) está em negrito. Na tabela 4.4 são mostrados os resultados da comparação dos métodos propostos com a PG clássica. Considerando os resultados, somente no conjunto N (que é o nível mais fácil), o $p\text{-}valor$ é > 0.05, o que permite considerar que não há diferença estatisticamente significativa entre os dois conjuntos comparados. Nos outros conjuntos de dados a diferença estatisticamente significativa é confirmada, o que comprova que o método proposto é estatisticamente melhor do que a PG clássica.

Tabela 4.4: Binomial-3 – p-valores resultando do teste t pareado

Conjunto	*p-valor*
N	0.536
0	$<$ **0.001**
1	$<$ **0.001**
2	$<$ **0.001**
3	$<$ **0.001**
4	$<$ **0.001**
5	$<$ **0.001**
6	$<$ **0.001**

Na Figura 4.2, um gráfico com a evolução do crescimento da solução é apresentado.

Os resultados do conjunto 6, que apresenta o maior grau de dificuldade, foram escolhidos para serem comparados.

Os valores apresentados são da média do *fitness* e o número de *hits* de 50 rodadas durante o processo evolutivo. Analisando o gráfico, pode-se observar que a evolução do *fitness* médio utilizando a abordagem proposta de PG é maior do que a abordagem clássica. Considerando o número de *hits*, esta diferença é ainda maior utilizando o algoritmo proposto.

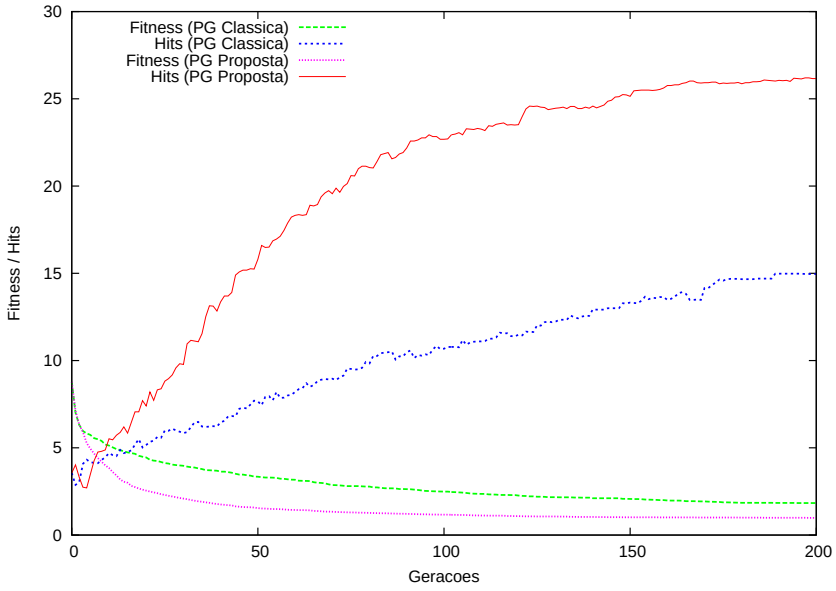

Figura 4.2: Binomial-3 – Evolução do fitness médio no conjunto 6

A partir dos resultados mostrados pode-se concluir que o método proposto obteve excelentes resultados para este problema. A diferença de melhoria, isto é, média de hits, entre os métodos chega a 40% e os resultados do teste t pareado, confirmaram que o método proposto possui um desempenho melhor.

Capítulo 5

Séries Temporais

Como parte do estudo para validação da proposta apresentada, um problema de aplicação real foi escolhido para ser estudado e ser submetido a testes utilizando as duas abordagens de algoritmos. Foi escolhido o problema de Séries Temporais, por se tratar de um problema de aplicabilidade real bastante interessante.

5.1 Fundamentação Teórica

Segundo [Souza, 1989], a classe de fenômenos cujo processo observacional e conseqüente quantificação numérica gera uma seqüência de dados distribuídos no tempo é denominada Série Temporal.

A natureza de uma Série Temporal e a estrutura de seu mecanismo gerador estão relacionados com o intervalo de ocorrência das observações no tempo.

Caso o levantamento das observações da série possa ser feito a qualquer tempo, a série é dita contínua, sendo denotada por $x(t)$ [Granger and Newbold, 1977]. Entretanto, segundo [Granger and Newbold, 1977] e [Nelson, 1973] na maioria das séries, as observações são tomadas em intervalos de tempo discretos e eqüidistantes.

Uma série temporal discreta pode ser representada por $X_t = x_1, x_2, ..., x_t$, sendo que cada observação discreta x_t está associada a um instante de tempo distinto, existindo uma relação de dependência entre essas observações [Souza, 1989].

Como exemplos de Séries Temporais podemos citar:

- Venda mensal de determinado produto.

- Índices diários da Bolsa de Valores.

- Vazão de determinada seção de um rio.

- Quantidade de falhas de determinado *hardware* ou *software*.

Em geral, séries temporais são representadas através de modelos não estacionários nos quais, a tendência e outras características que variam com o tempo podem ser tratadas estatisticamente. A construção de um modelo estatístico adequado para ajustar os dados apresenta um grau de dificuldade razoável. Uma escolha ruim de um modelo de ajuste pode conduzir a uma previsão enganosa e ineficiente [Box and Jenkins, 1976] [Chaves, 1991].

Na Figura 5.1, pode-se visualizar a curva de uma Série Temporal que representa a venda do produto 'X', durante 70 meses [Souza, 1989].

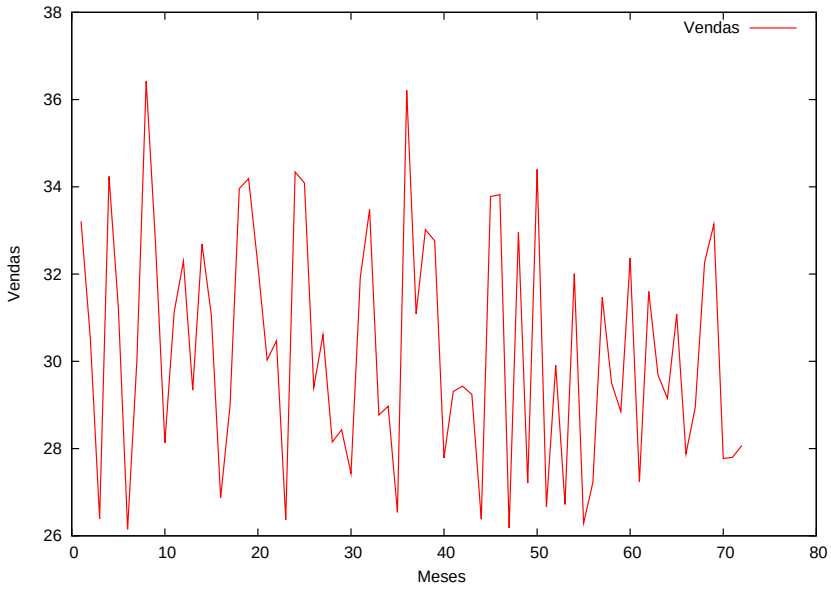

Figura 5.1: Exemplo de Série Temporal.

Na análise de uma série temporal, primeiramente deseja-se modelar o fenômeno estudado para, a partir daí, descrever o comportamento da série, fazer estimativas e, por último, avaliar quais os fatores que influenciaram o comportamento da série, buscando definir relações de causa e efeito entre duas ou mais séries. Para tanto, há um conjunto de técnicas estatísticas disponíveis que dependem do modelo definido (ou estimado, para a série), bem como do tipo da série que será analisada.

5.1.1 Previsão de Séries Temporais

Para [Barbancho, 1970], uma previsão é uma manifestação relativa a sucessos desconhecidos em um futuro determinado. Segundo [Morettin and Toloi, 1981], a previsão não constitui um fim em si, mas um meio de fornecer informações e subsídios para uma conseqüente tomada de decisão, visando atingir determinados objetivos.

[Souza, 1989] e [Wheelwright and Makridakis, 1985] classificam as previsões dos valores futuros de uma série temporal como de curto, médio ou longo prazo. Diante disso, [Refenes, 1993] especifica técnicas distintas para prognosticar os valores futuros de uma série temporal.

- Previsão de múltiplos passos: busca identificar as tendências gerais e pontos de inflexão mais relevantes na série temporal.

- Previsão de simples passo: a previsão é feita apenas para o período de tempo imediatamente posterior ao atual, a partir de observações da série temporal.

5.1.2 Métodos de Previsão

Wheelwright [Wheelwright and Makridakis, 1985] define um método de previsão como sendo o conjunto de procedimentos usados no desenvolvimento de uma determinada previsão. Ainda de acordo com [Wheelwright and Makridakis, 1985], a maioria dos métodos de previsão de séries temporais se baseia na suposição de que observações passadas contêm todas as informações sobre o padrão de comportamento da série temporal, sendo este padrão recorrente no tempo.

O propósito dos métodos de previsão consiste em distingüir qualquer ruído que possa estar contido nas observações e então usar esse padrão para prever os valores futuros da série temporal. Assim, pela identificação desse componente, a previsão para períodos de tempo subseqüentes ao observado pode ser desenvolvida. Dependendo do número de séries temporais envolvidas na modelagem, segundo [Souza, 1989], é possível classificar os métodos de previsão em:

- Univariados: compreendem a maior parte dos métodos de previsão de séries temporais, consideram somente uma única série para a realização de prognósticos.

- Funções de Transferência: nesta metodologia, a série é explicada não só pelo seu passado histórico, como também por outras séries temporais não-correlatas entre si.

- Multivariados: abrangem os procedimentos de previsão que associam mais de uma série temporal na efetivação de previsões, sem qualquer imposição com relação à causalidade entre elas.

Os métodos de previsão baseados exclusivamente em uma única série histórica são classificados em métodos de Decomposição, Métodos Simples e Métodos Avançados de previsão de Séries Temporais, segundo [Souza, 1989] e [Wheelwright and Makridakis, 1985].

5.1.3 Métodos de Decomposição de Séries Temporais

De acordo com [Souza, 1989], os métodos de decomposição assumem que uma série temporal é constituída por um conjunto de componentes não-observáveis. Pela identificação dos componentes individuais presentes no padrão básico da série histórica de dados (tendência, ciclo, sazonalidade e aleatoriedade), a extrapolação para o futuro pode ser realizada [Wheelwright and Makridakis, 1985].

A equação a seguir expressa o relacionamento entre os componentes não-observáveis da série temporal [Wheelwright and Makridakis, 1985]:

$$x_t = f(S_t, T_t, C_t, E_t) \tag{5.1}$$

onde S_t corresponde ao componente sazonal para o período t; T_t é o componente de tendência no período t; C_t é o componente de ciclo de vida em t e E_t é o componente aleatório em t.

Segundo o estudo de [Wheelwright and Makridakis, 1985] vários procedimentos para a decomposição de séries temporais foram desenvolvidos, cada qual tentando isolar os componentes não-observáveis da série o mais acuradamente possível. O objetivo destes procedimentos consiste em remover cada um dos componentes, permitindo que o comportamento da série temporal seja melhor compreendido e, conseqüentemente, prognosticar valores futuros mais apropriados [Mueller, 1996].

5.1.4 Métodos Simples de Previsão de Séries Temporais

Métodos Simples de Previsão efetuam a previsão do valor futuro da série temporal pelo alisamento das observações passadas da série de interesse [Wheelwright and Makridakis, 1985]. Tais métodos são bem populares devido à sua simplicidade, eficiência computacional e razoável previsão obtida [Morettin and Toloi, 1981].

Segundo [Wheelwright and Makridakis, 1985], os principais métodos de previsão simples são:

- Média Móvel - Considera como previsão para o período futuro a média das observações passadas recentes [Wheelwright and Makridakis, 1985]. O termo média móvel é utilizado porque à medida que a próxima observação se torna disponível, a média das observações é recalculada, incluindo essa observação no conjunto de observações e desprezando a observação mais antiga [Morettin and Toloi, 1981].

- Alisamento Exponencial Simples - A princípio, o método conhecido como Alisamento Exponencial Simples se assemelha ao da Média Móvel por extrair das observações da série temporal o comportamento aleatório pelo alisamento dos dados históricos. Entretanto, a inovação introduzida pelo Alisamento Exponencial Simples advém do fato deste método atribuir pesos diferentes a cada observação da série. Enquanto que na Média Móvel as observações usadas para encontrar a previsão do valor futuro contribuem em igual proporção para o cálculo da previsão, no Alisamento Exponencial Simples as informações mais recentes são evidenciadas pela aplicação de um fator que determina esta importância [Wheelwright and Makridakis, 1985].

- Alisamento Exponencial Linear - Enquanto o método Alisamento Exponencial Simples é aplicado na previsão de séries temporais que apresentam tendência entre as observações passadas, os valores prognosticados superestimam (ou subestimam) os valores reais [Morettin and Toloi, 1981]. Desta forma, a acuidade das previsões fica prejudicada. Para evitar este erro sistemático, o método Alisamento Exponencial Linear foi desenvolvido procurando reconhecer a presença de tendência [Wheelwright and Makridakis, 1985].

- Alisamento Exponencial Sazonal e Linear de Winter - Este método produz resultados similares ao Alisamento Exponencial Linear, sendo, no entanto, capaz de manipular

séries temporais que além de apresentarem tendência nos dados, apresentam também sazonalidade [Wheelwright and Makridakis, 1985].

5.1.5 Métodos Avançados de Previsão de Séries Temporais

No universo dos métodos de previsão de séries temporais mais complexos podem ser citados os modelos Autoregressivo e Médias Móveis (AR, MA e ARMA), modelo Autoregressivo Integrado de Médias Móveis (ARIMA), modelos ARMA Multivariáveis (MARMA) [Wheelwright and Makridakis, 1985].

- Modelo Autoregressivo (AR) [Wheelwright and Makridakis, 1985] - é dado pela equação (5.2):

$$x_t = \varphi_1 x_{t-1} + \varphi_2 x_{t-2} + ... + \varphi_p x_{t-p} + e_t \tag{5.2}$$

Onde:

x_t corresponde à observação da série temporal no tempo t; φ_p corresponde ao parâmetro do modelo AR de ordem p e e_t representa o erro de eventos aleatórios que não podem ser explicados pelo modelo.

- Modelo de Médias Móveis [Wheelwright and Makridakis, 1985] (MA) - é definido pela equação (5.3) :

$$x_t = e_t - \theta_1 e_{t-1} + \theta_2 e_{t-2} - ... - \theta_q e_{t-q} \tag{5.3}$$

Onde:

e_t representa o erro de eventos aleatórios que não podem ser explicados pelo modelo e θ_q corresponde ao parâmetro do modelo MA de ordem q.

- Modelo Autoregressivo e de Médias Móveis (ARMA) - Wheelwrigth e Makridakis [Wheelwright and Makridakis, 1985] especificam o modelo misto Autoregressivo e de Médias Móveis (ARMA), como sendo a combinação dos modelos AR e MA.

$$x_t = \varphi_1 x_{t-1} + \varphi_2 x_{t-2} + ... + \varphi_p x_{t-p} + e_t - \theta_1 e_{t-1} + \theta_2 e_{t-2} - ... - \theta_q e_{t-q} \quad (5.4)$$

Analisando a equação, é possível verificar que os modelos ARMA relacionam os valores futuros com as observações passadas, assim como também com os erros passados apurados entre os valores reais e os previstos.

Identificado o modelo, é necessário identificar os parâmetros deste modelo. Segundo [Judge, 1988], os parâmetros do processo AR são estimados através de métodos de regressão, caso exista o processo MA, os parâmetros são estimados através da aplicação de algum algoritmo de otimização não-linear. Tal tarefa não é trivial.

- Modelo de Box e Jenkins

O destaque atribuído ao modelo de Box e Jenkins [Box and Jenkins, 1976], que também pode ser incluído nesta classificação, é devido principalmente a sua fundamentação teórica, sendo, a princípio, capaz de manipular séries temporais de qualquer natureza.

O método de Box e Jenkins consiste na busca de um modelo ARIMA (*AutoRegressive Integrate Moving Average*) que represente o processo estocástico gerador da série temporal, a partir de um modelo ARMA aplicável na descrição de séries temporais estacionárias, estendendo este conceito para séries temporais não-estacionárias [Nelson, 1973]. Um processo $ARIMA(p, d, q)$ pode ser representado:

$$w_t = \varphi_1 x_{t-1} + ... + \varphi_p x_{t-p} + e_t - \theta_1 e_{t-1} + -... - \theta_q e_{t-q} \quad (5.5)$$

Sendo,

$$x_{t-1} = x_{t-1} - x_{(t-1)-d} \quad (5.6)$$

Onde:

φ_p e θ_q são os parâmetros dos processos Autoregressivo e de Média Móvel de ordem p e q (ARMA(p,q)), e_t corresponde ao erro de eventos aleatórios que não podem ser explicados

pelo modelo e d equivale ao grau de homogeneidade não-estacionário.

Entretanto, existem dificuldades com relação aos modelos descritos. Para ajustar um modelo é necessário fornecer o formato da equação a ser obtida, muitas tentativas precisam ser feitas até se conseguir a melhor equação. Para reduzir essas dificuldades, alguns autores propõem o uso de Programação Genética e têm obtido bastante sucesso [Kaboudan, 2000] [Povinelli, 1999].

Considerando este cenário, a PG aparece como um método alternativo para reduzir a dificuldade de previsão de séries temporais. Muitos trabalhos confirmam que a PG é uma abordagem funcional para esta tarefa em diferentes aplicações, tais como previsão de séries temporais financeiras [Povinelli, 1999] [Kaboudan, 2000] [Minglei, 2002] [Hui, 2003].

5.2 Experimentos

Diversos experimentos iniciais foram implementados para construir um algoritmo de PG para a previsão de Séries Temporais. Foram obtidos resultados que comprovam que a PG é uma técnica promissora na previsão de Séries Temporais, apresentando melhores resultados do que os métodos estatísticos tradicionais. Os resultados destes estudos iniciais podem ser encontrados em [Souza et al., 2005a] [Souza et al., 2005b].

O conjunto de funções utilizado para este problema é:

$$\{+, -, \div, \times, exp^x, sqrt, log, sin, cos\}$$

O operador de divisão foi criado como divisão protegida, para evitar divisões por *zero*. O conjunto de terminais é composto por $\{Z_{t-1}, Z_{t-2}, Z_{t-3}, Z_{t-4}, \theta\}$, onde Z_{t-n} é o n-ésimo valor da série e θ é um valor randômico real entre $[0, 1)$.

A função de fitness para este problema é o Raiz do Erro Médio Quadrático (RMSE). O RMSE é uma das medidas mais utilizadas para medir a eficiência de um método de previsão. O RMSE é definido de acordo com a fórmula abaixo. Neste experimento indivíduos com RMSE próximo ou igual a *zero* são os melhores.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(|x_i - \hat{x}_i|)^2}{n}} \tag{5.7}$$

Onde x_i é o valor real da série, $\hat{x}_i$ é o valor previsto e n é o tamanho da série temporal.

Para este problema, um conjunto de 50 séries temporais foi gerada utilizando o software estatístico R [Venables et al., 2005]. Séries temporais dos seguintes modelos foram geradas: AR, MA e ARMA, variando aleatoriamente as constantes e parâmetros dos modelos. Para cada série temporal, 90 rodadas foram realizadas com diferentes sementes para cada abordagem de Programação Genética.

5.3 Resultados

Pela análise dos resultados, pode-se ver que na maioria das séries temporais a média de fitness nos experimentos com a nova abordagem de PG é menor do que com a PG clássica.

Considerando os resultados do teste *t*, somente dois *p–valores* são maiores do que 0.05. Nas Tabelas 5.1 e 5.2, os resultados obtidos são mostrados. A média do *fitness* $\pm$ desvio padrão para cada série temporal e os *p–valores* são apresentados.

Os resultados obtidos confirmam que a PG é uma técnica poderosa para previsão de séries temporais. Com a nova abordagem, pode-se verificar uma melhoria no *fitness* e consequentemente uma redução no RMSE entre o valores real e previsto.

Tabela 5.1: Séries Temporais - Resultados

Modelo	*Série Temporal*	*Modelagem Estatística*	*PG Clássica*	*PG Proposta*	*p–valor*
AR	AR1_100	0.985	0.963 ± 0.01	0.953 ± 0.044	0.083
	AR1_101	1.015	0.992 ± 0.007	0.957 ± 0.049	< **0.001**
	AR1_102	0.929	0.91 ± 0.007	0.877 ± 0.04	< **0.001**
	AR1_103	0.999	0.976 ± 0.011	0.953 ± 0.048	< **0.001**
	AR1_104	1.021	0.981 ± 0.007	0.954 ± 0.052	< **0.001**
	AR1_105	1.051	1.034 ± 0.011	0.98 ± 0.06	< **0.001**
	AR1_106	1.062	1.047 ± 0.011	1.023 ± 0.056	< **0.003**
	AR1_108	0.971	0.944 ± 0.007	0.927 ± 0.028	< **0.001**
	AR1_109	1.032	1.014 ± 0.005	0.959 ± 0.033	< **0.001**
	AR1_110	1.125	1.009 ± 0.008	0.98 ± 0.073	< **0.006**
	AR1_11	1.105	1.079 ± 0.01	1.046 ± 0.051	< **0.001**
	AR2_351	0.951	0.932 ± 0.018	0.897 ± 0.044	< **0.001**
	AR2_352	0.921	0.907 ± 0.006	0.853 ± 0.03	< **0.001**
	AR2_353	1.032	1.024 ± 0.014	0.997 ± 0.039	< **0.001**
	AR2_354	0.971	0.963 ± 0.006	0.916 ± 0.034	< **0.001**
	AR2_355	0.951	0.92 ± 0.018	0.869 ± 0.032	< **0.001**
	AR2_356	0.974	0.957 ± 0.017	0.908 ± 0.037	< **0.001**
	AR2_357	0.951	0.942 ± 0.022	0.891 ± 0.037	< **0.001**
	AR2_358	1.054	1.045 ± 0.012	0.982 ± 0.041	< **0.001**
	AR2_359	1.081	1.065 ± 0.007	0.987 ± 0.044	< **0.001**
	AR2_360	0.989	0.969 ± 0.007	0.919 ± 0.035	< **0.001**
	AR2_362	1.054	1.027 ± 0.008	0.976 ± 0.034	< **0.001**
	AR2_363	0.992	0.971 ± 0.009	0.911 ± 0.044	< **0.001**
	AR2_364	0.987	0.967 ± 0.007	0.893 ± 0.032	< **0.001**
	AR2_36	0.992	0.982 ± 0.006	0.925 ± 0.044	< **0.001**
ARMA	ARMA_100	1.124	1.004 ± 0.009	0.931 ± 0.046	< **0.001**
	ARMA_101	1.074	1.058 ± 0.014	0.963 ± 0.034	< **0.001**
	ARMA_102	1.124	1.077 ± 0.023	0.973 ± 0.051	< **0.001**
	ARMA_10	1.223	1.127 ± 0.017	1.014 ± 0.051	< **0.001**
	ARMA_1	1.102	1.06 ± 0.028	0.977 ± 0.053	< **0.001**
	ARMA_2	0.946	0.922 ± 0.005	0.854 ± 0.027	< **0.001**
	ARMA_230	1.041	1.024 ± 0.011	0.931 ± 0.033	< **0.001**

Tabela 5.2: Séries Temporais - Resultados (continuação)

Modelo	*Série Temporal*	*Modelagem Estatística*	*PG Clássica*	*PG Proposta*	*p–valor*
ARMA	ARMA_231	1.051	1.043 ± 0.009	0.959 ± 0.041	< **0.001**
	ARMA_69	1.167	1.146 ± 0.016	1.104 ± 0.079	< **0.001**
	ARMA_70	1.342	1.332 ± 0.018	1.293 ± 0.074	< **0.001**
	ARMA_71	1.271	1.263 ± 0.018	1.2 ± 0.07	< **0.001**
	ARMA_75	1.291	1.289 ± 0.02	1.222 ± 0.069	< **0.001**
	ARMA_77	1.124	1.02 ± 0.014	0.98 ± 0.069	< **0.001**
MA	MA1_113	0.955	0.926 ± 0.009	0.9 ± 0.046	< **0.001**
	MA1_114	0.984	0.978 ± 0.007	0.945 ± 0.045	< **0.001**
	MA1_115	1.054	1.035 ± 0.013	1.003 ± 0.043	< **0.001**
	MA1_116	0.991	0.962 ± 0.005	0.937 ± 0.053	< **0.001**
	MA1_117	1.032	1.018 ± 0.003	0.972 ± 0.053	< **0.001**
	MA1_118	1.079	1.059 ± 0.007	1.002 ± 0.04	< **0.001**
	MA1_119	0.964	0.946 ± 0.01	0.933 ± 0.052	0.06
	MA1_120	1.061	1.045 ± 0.01	1.003 ± 0.055	< **0.001**
	MA1_121	1.129	1.078 ± 0.007	1.031 ± 0.062	< **0.001**
	MA1_122	1.103	1.021 ± 0.01	1.004 ± 0.038	< **0.001**
	MA1_123	0.956	0.924 ± 0.01	0.901 ± 0.04	< **0.001**
	MA1_12	1.003	0.983 ± 0.006	0.943 ± 0.042	< **0.001**

Capítulo 6

Modelagem da Confiabilidade de *Software*

O problema descrito neste capítulo, que envolve a modelagem da confiabilidade de *software*, baseou-se nos estudos e metodologia propostas em [Souza, 2004].

6.1 Fundamentação Teórica

A confiabilidade de *software* é uma característica que pode ser expressa como a probabilidade de um sistema trabalhar sem falhas, durante um período de tempo, em um determinado ambiente. Modelos de confiabilidade permitem estimar (ou predizer) a falha corrente ou a confiabilidade futura de um sistema. Tais modelos utilizam dados coletados durante a atividade de teste. O uso de tais modelos permite aos gerentes de projeto que estimem o tempo e esforço necessários para testar e disponibilizar o *software* num nível de confiança aceitável.

Diferentes modelos de confiabilidade de *software* têm sido propostos. Eles podem ser classificados de acordo com a hipótese do que cada modelo faz em sua formulação. Alguns deles são baseados no tempo [Moranda, 1975] [Moranda and Jelinski, 1972] [Musa, 1975], ou seja, consideram o tempo entre falhas. Outros consideram a cobertura de um critério de teste [Chen et al., 1996] [Crespo, 1997] [Malaiya et al., 2002] [Pasquine et al., 1996]. Um critério pode ser visto como um predicado a ser satisfeito. Ele é utilizado para selecionar

e avaliar casos de teste [Maldonado et al., 1992] [Rapps and Weyuker, 1985].

Alguns autores propõem outras classificações para os modelos. Os modelos paramétricos e tradicionais assumem e sua formulação analítica um comportamento pré-determinado onde alguns parâmetros precisam ser ajustados, corrigindo a curva dos dados de falha. Tais parametros são explicitamente definidos no modelo e têm uma interpretação física. Para ajustar os parâmetros dos modelos pramétricos, sempre existe um conjunto de hipóteses que pode não ser adequado para a maioria dos casos.

A modelagem não-paramétrica determina modelos e coeficientes. A influência de parâmetros externos e outras peculiariedades de um modelo pode ser eliminada se existir um modelo que seja capaz de se evoluir baseando-se no conjunto de dados coletados durante a fase de teste inicial. Entretanto, tais modelos incluem parâmetros em sua formulação analítica. Eles também são conhecidos como modelos de confiabilidade não-paramétricos porque estes parâmetros não têm uma interpretação física. Estes modelos utilizam técnicas de Aprendizado de Máquina, tais como Redes Neurais Artificiais, para a determinação do modelo de confiabilidade baseado no tempo [Aljahdali et al., 2001] [Karunanithi et al., 1992b] [Sitte, 1999] [Souza and Vergilio, 2006] ou cobertura.

Em [Costa et al., 2005], a Programação Genética foi utilizada como uma abordagem alternativa para determinar a confiabilidade de *software* em modelos baseados no tempo e no teste de cobertura. Os resultados mostraram que os modelos gerados pela Programação Genética se adaptam melhor à curva de confiabilidade, quando comparados aos modelos tradicionais e ao modelo de Redes Neurais Artificiais.

Considerando o que foi exposto, este trabalho pretende testar o comportamento do algoritmo proposto neste tipo de problema. Bons resultados já foram obtidos pelo método clássico de Programação Genética. Acredita-se que, com a abordagem proposta, tais resultados sejam melhores ou pelo menos equivalentes aos encontrados no experimento realizado em [Costa et al., 2005].

6.2 Experimentos

6.2.1 Experimento 1

Este experimento explora modelos de confiabilidade de *software* baseados no tempo entre falhas.

6.2.1.1 Conjuntos de Dados

Os conjuntos de dados utilizados contém falhas obtidas por John Musa nos Laboratórios da empresa Bell Telephone [Musa, 1980]. Ele coletou dados para auxiliar gerentes de projetos em atividades de teste e auxiliar pesquisadores a avaliar modelos de confiabilidade de *software*.

O banco de dados contém 16 conjuntos de dados. Cada conjunto de dados contém dados de falha de um projeto com respeito a diferentes aplicações, tais como comando e controle de tempo real, processadores de texto, aplicações comerciais e militares. Os dados de falha consistem em identificação do projeto, número da falha, tempo entre falhas (TBF) e dia da ocorrência. A Tabela 6.1 apresenta o tamanho dos conjuntos de dados utilizados.

Tabela 6.1: Conjuntos de Dados

Conjunto	*Tamanho*	*Conjunto*	*Tamanho*
1	136	5	831
14C	36	6	73
17	38	SS1A	112
2	54	SS1B	375
27	41	SS1C	277
3	38	SS2	192
4	53	SS3	278
40	101	SS4	196

6.2.1.2 Pré–processamento

Os dados de falhas foram inicialmente gravados em vetores, ordenados por dia de ocorrência para então serem processados.

Dados de falha, tais como TBF, podem apresentar um alto nível de ruído, devido a ativação de um defeito que causa a falha, dependendo de como o sistema do *software* é operado. Esta imprecisão afeta a convergência do modelo. De maneira a reduzir o nível de ruído, o método de médias móveis foi aplicado nos conjuntos de dados. Como exemplo, as Figuras 6.1 e 6.2 apresentam o método aplicado ao conjunto de dados 1.

Dado um conjunto de números $Y_1, Y_2, Y_3, ..., Y_n$. A média móvel da K-ésma ordem é definida como a série:

$$\frac{Y_1, Y_2, ..., Y_K}{K}; \frac{Y_2, Y_3, ..., Y_{K+1}}{K}; \frac{Y_{n+1-k}, Y_{n+2-k}, ..., Y_n}{K} \tag{6.1}$$

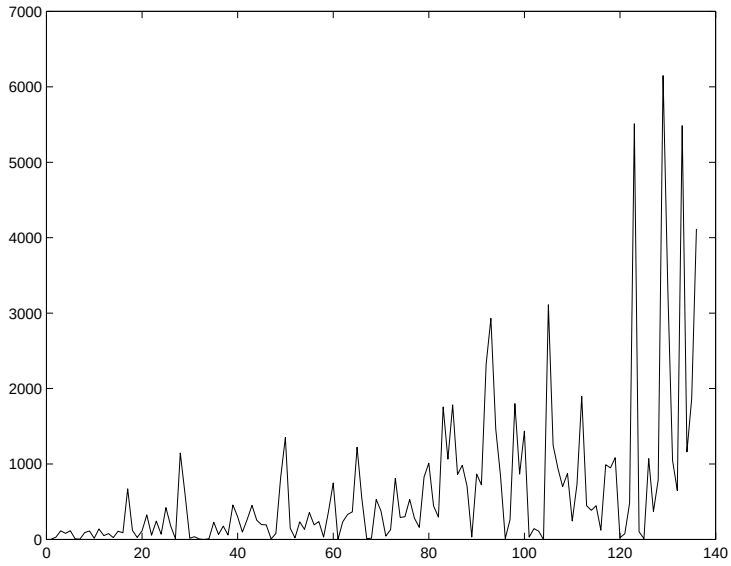

Figura 6.1: Tempo entre falhas (TBF).

Após aplicado o método das médias móveis, o tempo entre falha dos dados foram convertidos para dados de tempo decorrido. O vetor de tempos decorridos contém o tempo entre falhas acumulado e representa o tempo absoluto em que cada falha ocorreu desde o início do teste. Esta unidade tem uma curva suave, criando um modelo mais provável de convergir.

A Figura 6.3 mostra os dois vetores, plotando o tempo entre cada falha ocorrida e o intervalo entre cada uma delas e seu sucessor (TBF).

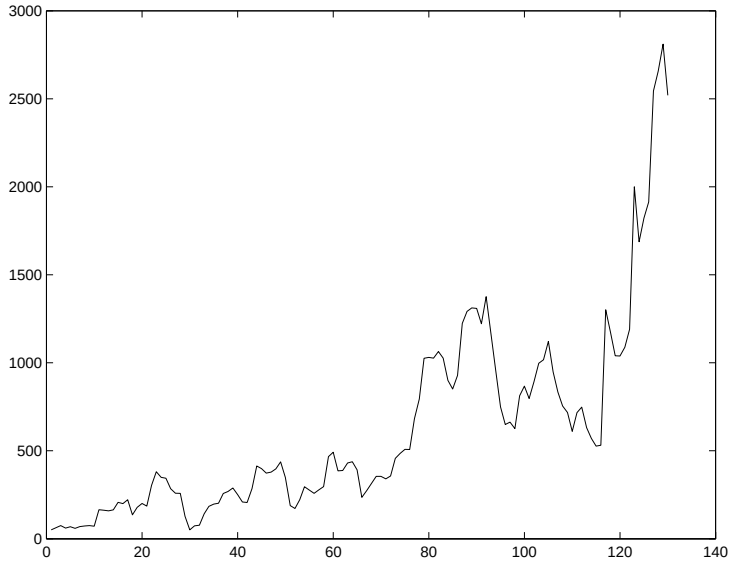

Figura 6.2: Média móvel de 7a. ordem.

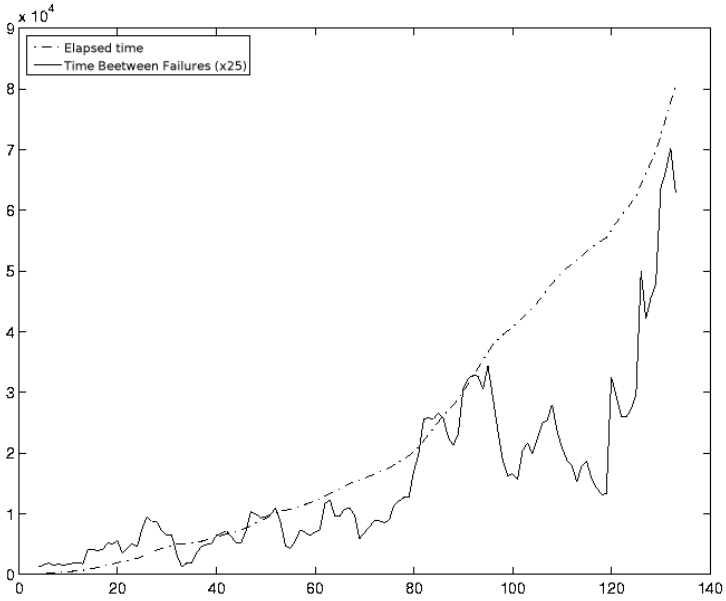

Figura 6.3: Falhas x Tempo

6.2.1.3 Avaliação dos Modelos

Foram utilizadas cinco medidas diferentes, para avaliar os modelos, foram utilizadas. Elas estão descritas a seguir. Na descrição, y e $\hat{y}$ são os valores observados e previstos, respectivamente, e n é o total de estimações realizadas.

Desvio máximo (md): É a máxima diferença entre a estimação do modelo e o valor

observado. Pode ser obtido com a equação:

$$md = max\left(\frac{|y - \hat{y}|}{y}\right) \tag{6.2}$$

Average Bias (ab)[Karunanithi et al., 1992a]: Mede a tendência geral de superestimação ou subestimação do número de falhas.

$$ab = \frac{1}{n}\sum_{i=1}^{n}\frac{y_i - \hat{y}_i}{y_i} \tag{6.3}$$

Average Error (ae) [Karunanithi et al., 1992a]: Mede o quão bem o modelo prediz durante a fase de teste. É dado por:

$$ae = \frac{1}{n}\sum_{i=1}^{n}\frac{|y_i - \hat{y}_i|}{y_i} \tag{6.4}$$

Prediction Error: É o erro do último valor estimado, dado por:

$$pe = \frac{y_n - \hat{y}_n}{y_n} \tag{6.5}$$

Coeficiente de Correlação (cc): Representa o quão bem a estimação corresponde aos valores observados. Um valor de correlação próximo a 1 significa que o modelo representa perfeitamente a relidade. É dado pela equação:

$$cc = \frac{s_{xy}}{\sqrt{s_{xx}}\sqrt{s_{yy}}} \tag{6.6}$$

Onde:

$$s_{xy} = \sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y}) \tag{6.7}$$

6.2.2 Experimento 2

Este experimento explora modelos de confiabilidade de *software* baseados no critério de cobertura de teste. Os modelos obtidos com o método de PG proposto foram compa-

rados com os resultados obtidos em [Costa et al., 2005] e com o modelo *Coverage Based Binomial model* (CBB) proposto por Crespo em [Crespo, 1997].

6.2.2.1 Conjuntos de Dados

Os conjuntos de dados e os critérios de teste utilizados neste experimento são os mesmos utilizados em [Crespo, 1997] [Souza and Vergilio, 2006]. Os dados foram obtidos através de dados de falhas do programa denominado *Space*. Durante o processo de teste e o uso operacional deste programa, 33 defeitos foram descobertos, removidos e registrados em um conjunto. Crespo, reimplantou os defeitos novamente no programa e o utilizou para avaliar a cobertura dos métodos e coletar dados de cobertura para o seu modelo.

Os dados de falha consistem do total de número de defeitos descobertos, da confiabilidade do *software* e da cobertura dos seguintes critérios de teste estruturais: todos-os-nodos (AN), todos-os-arcos (AE), todos-os-potenciais-usos (PU), todos-os-potenciais-DU-caminhos (PDU) e todos-os-potenciais-usos/DU (PUDU) [Maldonado et al., 1992].

6.2.2.2 Avaliação dos Modelos

Juntamente com as medidas *ab, ae* e *cc*, definidas na Seção 6.2.1.3 deste trabalho, foi utilizado o teste de Kolmogorov–Smirnov [Costa et al., 2005]. Este teste é utilizado para determinar se dois conjuntos de dados são significantemente diferentes. Assim, dados coletados em uma ocasião podem ser comparados com dados coletados em outra ou comparados com dados obtidos com um modelo matemático com objetivo de veriricar se duas amostras seguem a mesma distribuição. Em um caso afirmativo, pode ser dito que duas amostras são da mesma população e concluir que elas podem representar a mesma realidade [Trivedi, 2001].

Para o teste Kolmogorov–Smirnov, primeiramente foi obtida a diferença entre duas amostras. Então, foi multiplicado a maior diferença (D_{max}) pelo tamanho da amostra (N). Se o valor é maior do que o valor crítico ($D_{critical}$), com um grau de significância (α), então pode-se dizer que as duas amostras representam a mesma população.

6.3 Resultados

6.3.1 Experimento 1

Em um primeiro momento, foram realizados experimentos com os dados de confiabilidade e a aborgadem de PG clássica. Foram definidos 2 conjuntos de funções, descritos abaixo:

Conjunto de Funções 1 = $\{+, -, \div, \times, exp^x, exp^{-x}, sqrt, log, sin, cos\}$

Conjunto de Funções 2 = $\{+, -, \div, \times, exp^x, sqrt, log\}$

O conjunto 2 foi utilizado para melhorar os resultados dos conjuntos 14C, 27 e 3, como pode ser observado nas Tabelas 6.2 e 6.3. Para cada conjunto de dados, 10 modelos de PG foram treinados utilizando sementes iniciais diferentes. O melhor modelo para cada conjunto foi escolhido.

Os resultados da PG foram comparados com os modelos tradicionais: Jelinksi-Moranda (JAM) e o modelo Geométrico (GEO) e também com Redes Neurais Artificiais (ANN). Os modelos gerados pela PG se adaptaram melhor sob más condições e foram capazes de capturar a tendência da curva mais facilmente do que outros modelos [Costa et al., 2005].

Tabela 6.2: Confiabilidade - PG Clássica - Experimento 1 (Conjunto de Funções 1)

Conjunto	*md*	*pe*	*ab*	*ae*	*cc*
1	**11.45**	**11.45**	**1.72**	**3.96**	**99.82**
14C	128.65	-128.65	-348.86	354.99	16.90
17	**13.55**	**0.97**	**1.12**	**4.31**	**99.44**
2	**9.65**	**4.59**	**0.94**	**3.54**	**99.82**
27	50.30	50.30	8.56	12.80	90.23
3	67.11	67.11	11.63	14.98	70.94
4	**39.05**	**39.02**	**-1.65**	**3.20**	**97.53**
40	**6.43**	**-6.43**	**-7.57**	**9.81**	**99.16**
5	**9.61**	**9.37**	**0.92**	**9.48**	**99.74**
6	**21.79**	**14.82**	**5.36**	**9.14**	**98.31**
SS1A	**22.39**	**-16.45**	**-10.05**	**11.89**	**99.21**
SS1B	**5.27**	**5.27**	**-2.75**	**5.66**	**99.78**
SS1C	**10.80**	**0.57**	**-6.11**	**8.44**	**99.72**
SS2	**9.44**	**5.08**	**12.17**	**16.28**	**99.58**
SS3	**15.61**	**14.78**	**3.42**	**7.51**	**99.31**
SS4	**5.71**	**1.20**	**2.51**	**5.45**	**99.84**

Em um segundo momento, os mesmos conjuntos de dados foram submetidos a experimentos, utilizando a abordagem de PG proposta. Novos testes foram realizados, utili-

Tabela 6.3: Confiabilidade - PG Clássica - Experimento 1 (Conjunto de Funções 2)

Conjunto	*md*	*pe*	*ab*	*ae*	*cc*
1	12.46	12.36	2.07	4.74	99.76
14C	**21.94**	**21.95**	**1.96**	**10.26**	**98.07**
17	54.48	54.58	7.88	10.26	97.46
2	13.30	13.93	3.67	8.25	99.24
27	**7.37**	**0.25**	**1.98**	**13.18**	**99.14**
3	**27.71**	**27.71**	**5.29**	**9.38**	**98.69**
4	201.64	201.64	17.64	20.44	87.38
40	173.00	173.00	51.00	54.49	88.63
5	2.19	2.19	4.58	9.19	99.73
6	23.62	15.11	5.11	10.82	97.67
SS1A	35.12	35.12	9.63	12.18	98.56
SS1B	23.13	23.13	13.52	18.78	98.34
SS1C	9.04	9.04	5.52	17.49	99.29
SS2	12.63	4.36	5.43	11.06	99.40
SS3	27.75	27.75	0.35	10.87	98.48
SS4	17.31	13.92	18.49	26.93	98.64

zando somente o Conjunto de Funções 1. De acordo com os resultados, nota-se que houve uma melhoria nas medidas dos modelos, principalmente nos modelos que apresentaram mau desempenho utilizando a abordagem de PG clássica (14C, 27, 3). Os resultados dos experimentos podem ser vistos na Tabela 6.4.

Tabela 6.4: Confiabilidade - PG Proposta - Experimento 1 (Conjunto de Funções 1)

Conjunto	*md*	*pe*	*ab*	*ae*	*cc*
1	-11.43	-11.43	-2	4.35	99.81
14C	-299.24	-299.24	-33.97	37.06	81.02
17	-13.86	-13.86	-1.84	3.95	99.58
2	9.67	-2.18	1.28	3.32	99.73
27	5.86	5.86	-2	5.25	99.38
3	-17.76	-17.76	-1.73	5.36	99.09
4	-38.51	-38.51	-4.73	7.78	98.59
40	-8.27	-8.27	-4.42	8.18	99.33
5	-6.59	-6.59	-4.23	8.96	99.71
6	20.96	11.83	5.56	8.94	98.43
SS1A	-0.86	-0.86	-2.72	5.31	99.78
SS1B	16.58	16.58	1.07	6.72	99.34
SS1C	7.91	7.91	-7.9	15.68	99.53
SS2	8.84	0.33	-0.22	6.15	99.76
SS3	18.66	16.15	8.36	15.71	98.76
SS4	10.24	7.36	-6.66	14	99.64

Na Tabela 6.5, encontram-se os resultados obtidos com o teste t pareado, aplicado sobre os resultados da medida *ae*, obtidos com a abordagem clássica e a abordagem proposta. Os resultados mostram que em dois dos conjuntos de dados (SS2 e SS3) não houve diferença estatisticamente significativa ($p - valor > 0.05$) entre os dois métodos comparados. Para os demais conjuntos de dados houve diferença estatisticamente significativa ($p - valor < 0.05$).

Tabela 6.5: Confiabilidade (Experimento 1) - p–valores resultando de teste t pareado

Conjunto	*p-valor*
1	$<$ **0.001**
14C	**0.014**
17	**0.001**
2	$<$ **0.001**
27	$<$ **0.001**
3	$<$ **0.001**
4	$<$ **0.001**
40	$<$ **0.001**
5	$<$ **0.001**
6	**0.002**
SS1A	**0.001**
SS1B	**0.006**
SS1C	**0.008**
SS2	0.285
SS3	0.324
SS4	**0.046**

6.3.2 Experimento 2

Nesta primeira parte do experimento os resultados obtidos com a abordagem de PG clássica foram comparados com o modelo CBB (*Coverage Based Binomial*) [Crespo, 1997].

O conjunto de funções utilizado contém os seguintes operadores:

$$\{+, -, /, *, exp^x, sqrt, log, sin, cos\}$$

Os resultados são mostrados nas Tabelas 6.6 e 6.7.

Pelos resultados apresentados, nota-se que a PG clássica apresenta bons resultados perante ao modelo CBB. Em um segundo momento, experimentos com a nova abordagem

Tabela 6.6: Cobertura - Modelo CBB

	Critério				
	AN	AE	PU	PUDU	PDU
D_{max}	0.17	0.16	0.17	0.17	0.17
KD	4.73	4.48	4.74	4.73	4.78
H_0	sim	sim	sim	sim	sim
ae	0.4023	0.4228	0.4098	0.4051	0.4173
ab	0.1795	0.1585	0.1946	0.1811	0.2058
cc	0.9787	0.9761	0.9784	0.9778	0.9783

Tabela 6.7: Cobertura - PG Clássica

	Critério				
	AN	AE	PU	PUDU	PDU
D_{max}	0.094	0.090	0.134	0.075	0.083
KD	2.622	2.521	3.750	2.111	2.334
H_0	sim	sim	sim	sim	sim
ae	0.240	0.215	0.349	0.174	0.996
ab	0.176	0.164	0.023	0.002	0.041
cc	0.994	0.995	0.988	0.996	0.996

de PG foram realizados. Os resultados são mostrados na Tabela 6.8.

Tabela 6.8: Cobertura - PG Proposta

	Critério				
	AN	AE	PU	PUDU	PDU
D_{max}	0.058	0.087	0.086	0.070	0.074
KD	1.624	2.436	2.408	1.960	2.072
H_0	sim	sim	sim	sim	sim
ae	0.019	0.004	0.036	0.040	0.033
ab	0.105	0.095	0.220	0.082	0.216
cc	0.997	0.995	0.994	0.997	0.996

Analogamente ao Experimento 1, na Tabela 6.9, encontram-se os resultados obtidos com o teste t pareado, utilizando a medida ae, aplicado sobre os resultados obtidos com a abordagem clássica e a abordagem proposta. Os resultados mostram que em todos os critérios de teste ocorre uma diferença estatisticamente significativa entre os dois métodos comparados ($p - valor < 0.05$).

Os resultados obtidos com os dois experimentos realizados mostraram que com o mé-

Tabela 6.9: Confiabilidade (Experimento 2) - p–valores resultando de teste t pareado

Critério	*p-valor*
AN	$<$ **0.001**
AE	$<$ **0.001**
PU	$<$ **0.001**
PUDU	$<$ **0.001**
PDU	$<$ **0.001**

todo proposto houve uma melhoria no desempenho. No primeiro experimento, relativo a modelos baseado no tempo entre falhas, foi possível com o método proposto obter bons resultados em todos os conjuntos de dados utilizando um único conjunto de operadores, sendo também observado uma melhoria nas medidas de avaliação (md, ae, ab, cc). No segundo experimento, que contém modelos baseados em cobertura de teste, houve uma melhoria nas medidas de avaliação (ae, ab, cc) e também uma melhoria nos valores de D_{max}.

Capítulo 7

Santa Fe Artificial Ant

7.1 Fundamentação Teórica

O último problema estudado é o problema *Santa Fe Artificial Ant*, comumente chamado de "Problema da Formiga". Este problema foi popularizado por Koza [Koza, 1992], mas originalmente foi desenvolvido pela área da Inteligência Artificial, denominada Vida Artificial. O problema consiste em encontrar a melhor estratégia para obter comida ao longo de uma trilha numa grade. A solução para o problema é um algoritmo para coletar comida.

Geralmente, para este problema é utilizada a trilha Santa Fé, que consiste de uma grade de 32x32 com 89 pontos contendo comida, conforme mostrado na Figura 7.1. A "formiga" deve iniciar no ponto mais ao Noroeste, com face para o Leste da grade.

7.2 Experimentos

Este problema foi implementando de acordo com [Zongker and Punch, 1995]. O conjunto de funções e terminais são executados para mover a "formiga" pela grade durante o processo evolutivo. Os terminais fornecem à "formiga" locomoção e mudança de direção. Na Tabela 7.1 é mostrado, o conjunto de funções e terminais para este problema .

O valor de *fitness* é medido pela quantidade de comida consumida pela "formiga". Para este problema, 600 rodadas de cada abordagem de Programação Genética foram

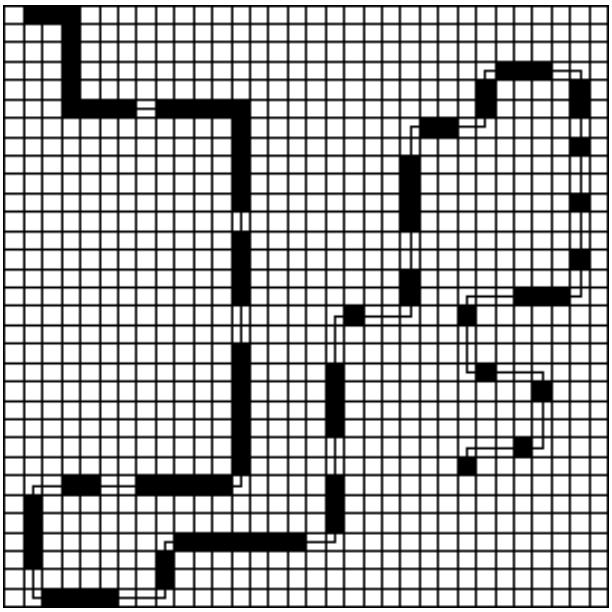

Figura 7.1: Trilha Santa Fe

realizadas, utilizando sementes diferentes.

Tabela 7.1: Funções e Terminais - Problema Santa Fé

Funções	
if_food_ahead(a,b)	se a "formiga" encontra uma comida à sua frente entao avalia a, senão avalia b
progn2(a,b)	avalia a, então b, retorna b
progn3(a,b,c)	avalia a, b então c, retorna c
progn4(a,b,c,d)	avalia a, b, c então d, retorna d – *esta função é opcional*
Terminais	
left	gira a "formiga" para a esquerda
right	gira a "formiga" para a direita
move	move a "formiga" a um passo adiante

7.3 Resultados

Na Tabela 7.2 são mostrados os resultados deste experimento. Os resultados obtidos com o problema da formiga apontam que a abordagem clássica é melhor para este problema, considerando os resultados da média da comida consumida e a média de tempo. A diferença é estatísticamente significativa considerando o *p–valor* , o qual comprova que a abordagem clássica tem um comportamento melhor do que a abordagem proposta.

Tabela 7.2: *Artificial Ant* - Resultados

Método	*Média de Comida Consumida ± Desvio*	*% Sucesso*	*Tempo Médio*	*p–valor*
PG Clássica	87.61 ± 4.64	98.44	562.84	$<$ **0.001**
PG Proposta	70.87 ± 12.32	79.63	599.80	

Na Figura 7.2, o progresso do processo evolutivo é apresentado. Os valores no gráfico são da média da comida consumida de 100 rodadas do processo evolutivo. O gráfico contém a evolução das primeiras 250 gerações.

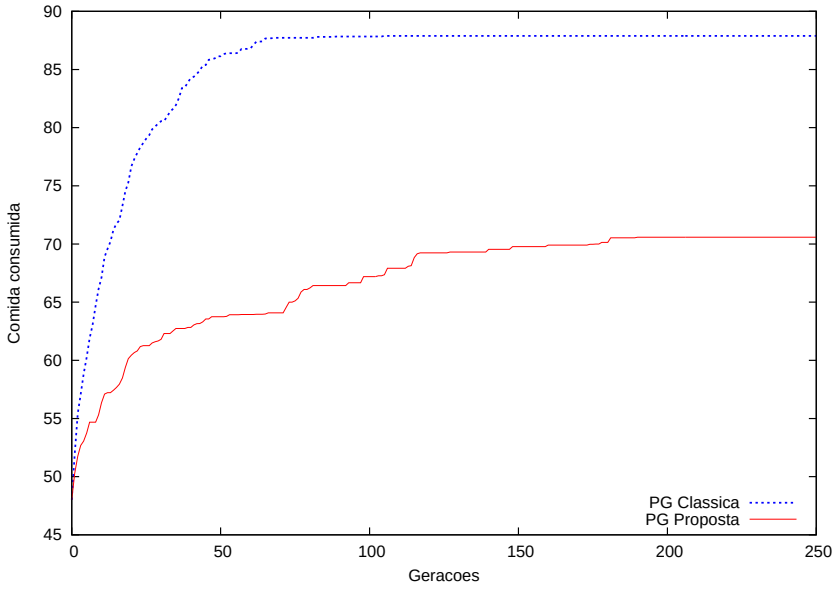

Figura 7.2: *Artificial Ant* – Progresso do processo evolutivo

Analisando os resultados, pode-se verificar que na geração 100, a PG clássica obtém os melhores resultados e acontece uma estagnação no processo evolutivo. Com a nova abordagem isto não acontece. O processo evolutivo faz progressos até perto da geração 200. Entretando, este progresso não é suficiente para encontrar uma solução tão boa como a encontrada pela PG clássica.

Novos experimentos foram realizados na tentativa de encontrar melhores resultados utilizando a abordagem proposta, ou identificar a razão pela qual a abordagem não trabalhou tão bem neste problema.

Considerando esta afirmação, a seleção elitista não foi alterada. O operador de mutação foi retornado a sua forma original e o algoritmo foi executado. Como pode ser visto

na Tabela 7.3, os resultados utilizando o algoritmo com mutação original são próximos aos obtidos com a PG clássica.

Tabela 7.3: *Artificial Ant* - Resultados utilizando a mutação original

Método	*Média de Comida Consumida ± Desvio*	*% Sucesso*	*Tempo Médio*
PG Clássica	87.61 ± 4.64	98.44	562.84
PG Proposta alterada	**78.75 ± 7.83**	**88.49**	**599.39**
PG Proposta	70.87 ± 12.32	79.63	599.80

Conclui-se que o método proposto não obteve bom desempenho para o problema da formiga. Acredita-se que a principal razão deste acontecimento, é o fato de que este problema, em particular, apresenta um domínio bem específico e diferente do domínio dos problemas de Regressão Simbólica. Isto nos leva a crer que para este tipo de problema é necessário estudos mais aprofundados que levariam a adaptação do método proposto, ou até mesmo a proposta de um novo método que conseguisse melhorar tais resultados.

Capítulo 8

Conclusões e Trabalhos Futuros

A presente dissertação apresentou uma proposta de modificação do algoritmo de Programação Genética, baseando-se nos conceitos da Teoria das Estratégias Evolucionárias.

A principal motivação deste trabalho foram diversos estudos propostos em [Daida, 2005], [Stevens et al., 2005], [Daida et al., 2001] e [Silva and Costa, 2005] realizados na área da Programação Genética que apontaram diversas propostas de melhoria (minimização do erro, aumento do número de *hits*) para o desempenho da Programação Genética e também que identificam as causas que afetam a habilidade da PG em encontrar uma solução em determinados casos.

Em vista disto, este trabalho propôs a criação de um algoritmo de Programação Genética, baseado mais fortemente na Teoria das Estratégias Evolucionárias do que nos conceitos dos Algoritmos Genéticos. O algoritmo foi implementado utilizando o sistema Lil-gp [Zongker and Punch, 1995].

Para validar a abordagem proposta, quatro estudos de caso foram realizados, utilizando problemas de dois domínios distintos. No domínio de problemas de Regressão Simbólica, foram estudados: Binomial-3, Séries Temporais e Confiabilidade de Software. O segundo domínio compreende o Problema da Formiga (*Santa Fe Artificial Ant*).

Experimentos foram realizados, utilizando a abordagem clássica da PG e a abordagem proposta, obtendo-se resultados conclusivos a respeito da abordagem proposta:

No problema Binomial–3, obteve-se excelentes resultados com a abordagem proposta.

A melhoria no número de hits chegou a 40% no conjunto com maior grau de dificuldade, comprovando que a abordagem proposta possui um desempenho melhor.

Com o problema de Séries Temporais, verificou-se também uma melhoria com relação ao erro de previsão das mesmas. A melhoria não foi tão significativa como a encontrada no problema Binomial–3, mas pode-se notar que com a abordagem proposta, houve uma redução no RMSE entre os valores real e previsto na maioria das séries temporais estudadas.

Nos experimentos realizados para modelagem da confiabilidade de software obteve-se melhorias. No primeiro experimento, baseado no critério de tempo entre falhas, além da melhoria encontrada nos coeficientes de correlação e medidas *ab, ae, md*, foi possível obter melhores resultados utilizando somente um único conjunto de funções, o que não ocorreu na abordagem clássica. No segundo experimento, baseado no critério de cobertura, uma melhoria nas medidas de avaliação foi encontrada.

No último experimento, com o problema da formiga artificial, não se obteve resultados que pudessem comprovar que a abordagem proposta para este domínio de problema é eficiente. Alterações na abordagem foram realizadas, na tentativa de identificar a razão pela qual para o problema da formiga não ocorreu uma melhoria.

Isto nos leva a concluir que para problemas do domínio de regressão simbólica, a abordagem proposta funciona muito bem, em virtude de suas características, tais como a existência de grande quantidade de soluções possíveis e também pela possibilidade de comutatividade de funções. Para o problema da formiga, que possui características diferentes e um número menor de soluções possíveis, esta abordagem não é válida.

A principal contribuição desta dissertação está na proposta de um algoritmo de Programação Genética contendo conceitos da Teoria das Estratégias Evolucionárias. Os estudos preliminares aqui realizados servem como base para que novos estudos possam ser desenvolvidos na tentativa de melhorar a abordagem proposta e conseqüentemente possibilitar melhorias na performance da técnica de PG.

Trabalhos futuros envolvem o estudo dos motivos pelos quais a abordagem não funcionou bem no problema *Santa Fe Artificial Ant*, a realização de novos experimentos

utilizando outros domínios de problema e também a possibilidade de criação de novos operadores de mutação ou seleção.

Referências Bibliográficas

[Aljahdali et al., 2001] Aljahdali, S. H., Sheta, A., and Rine, D. (2001). Prediction of software reliability: A comparison between regression and neural network non-parametric models. In *ACS/IEEE International Conference on Computer Systems and Aplications*, pages 470–473.

[Banzhaf et al., 1998] Banzhaf, W., Nordin, P., Keller, R. E., and Francone, F. D. (1998). *Genetic Programming: An Introduction*. Morgan Kaufmann.

[Barbancho, 1970] Barbancho, A. G. (1970). *Fundamentos e Possibilidades da Econometria*. Forum Editora.

[Barone and cols., 2003] Barone, D. and cols. (2003). *Sociedades Artificiais. A Nova Fronteira da Inteligência nas Máquinas*. Bookman.

[Bohm and Geyer-Schulz, 1996] Bohm, W. and Geyer-Schulz, A. (1996). An exact uniform initialization for genetic programming. In *Foundations of Genetic Algorithms*. Morgan Kaufmann.

[Box and Jenkins, 1976] Box, G. and Jenkins, G. M. (1976). *Time Series Analysis forecasting and Control*. Holden-Day.

[Bäck et al., 2003] Bäck, T., Rudolph, G., and Schewefel, H. P. (2003). Evolutionary Programming and evolution strategies: similarities and differences. *Annual Conference on Evolutionary Programming*, pages 11–22.

[Chaves, 1991] Chaves, N. A. (1991). *Bootstrap em Séries Temporais*. PhD thesis, Departamento de Engenharia Elétrica – Grupo de Sistemas, PUC-RJ, Rio de Janeiro, Brasil.

[Chellapilla, 1997] Chellapilla, K. (1997). *Evolving computer programs without subtree crossover*. IEEE Press.

[Chen et al., 1996] Chen, N., Lyu, M. R., and Wong, W. E. (1996). An empirical study of the correlation between code coverage and reliability estimation. In *IEEE Proceedings of Metrics*, pages 133–141.

[Costa et al., 2005] Costa, E. O., Vergílio, S. R., Pozo, A., and Souza, G. (2005). Modeling software reliability growth with genetic programming. In *XVI International Symposium of Software Reliability Engineering*. IEEE Computer Society, Chicago, USA.

[Crespo, 1997] Crespo, A. N. (1997). *Modelos de Confiabilidade de Software Baseados em Cobertura de Critérios Estruturais de Teste*. PhD thesis, Universidade Estadual de Campinas, Brasil.

[Daida, 2005] Daida, J. M. (2005). Towards identifying populations that increase the likelihood of success in genetic programming. In *GECCO-2005 - Genetic and Evolutionary Computation Conference*, pages 1627–1634. Association for Computing Machinery, Washington, D.C., USA.

[Daida et al., 1999] Daida, J. M., Bertram, R. R., Polito 2, J. A., and Stanhope, S. A. (1999). Analysis of single-node (building) blocks in genetic programming. In *Advances in Genetic Programming 3*. The MIT Press, Cambridge.

[Daida et al., 2001] Daida, J. M., Bertram, R. R., Stanhope, S. A., Khoo, J. C., Chaudhary, S. A., Chaudhri, O. A., and Polito 2, J. A. (2001). What makes a problem gp-hard? analysis of a tunably difficult problem in genetic programming. *Genetic Programming and Evolvable Machines*, 2:165–191.

[Darwin, 1859] Darwin, C. (1859). *On the origin of species by means of natural selection of the preservation of favored races in the struggle for life.* John Murray, London, UK.

[Dianati et al., 2002] Dianati, M., Song, I., and Treiber, M. (2002). An introduction to genetic algorithms and evolution strategies. Technical report, University of Waterloo, Ontario, Canada.

[Goldberg, 1989] Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley.

[Granger and Newbold, 1977] Granger, C. W. J. and Newbold, P. (1977). *Forecasting Economic Time Series.* Academic Press.

[Holland, 1992] Holland, J. H. (1992). *Adaptation in natural and artificial systems.* MIT Press.

[Hui, 2003] Hui, A. (2003). Using genetic programming to perform time series forecasting of stock prices. In *Genetic Algorithms and Genetic Programming*, pages 83–90. John Koza, Stanford, California, USA.

[Judge, 1988] Judge, G. G. (1988). *Introduction to the Theory and Practice of Econometrics.* John Wiley and Sons Inc. New York, USA.

[Kaboudan, 2000] Kaboudan, M. A. (2000). Genetic Programming Prediction of Stock Prices. *Journal Computational Economics*, 16:207–236.

[Karunanithi et al., 1992a] Karunanithi, N., Verma, P., and Malaiya, Y. (1992a). Predictability of software reliability models. *IEEE Transactions on Reliability*, 41(4):539–546.

[Karunanithi et al., 1992b] Karunanithi, N., Whitley, D., and Malaiya, Y. K. (1992b). Prediction of software reliability using connectionist models. *IEEE Transactions on Software Engineering*, 18(7):563–574.

[Koza, 1992] Koza, J. R. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* MIT Press.

[Kusiak, 2000] Kusiak, A. (2000). Evolutionary computation and data mining. In *SPIE Conference on Intelligent Systems and Advanced Manufaturing*, pages 1–10. Boston, MA, USA.

[Luke and Painait, 2001] Luke, S. and Painait, L. (2001). A survey and comparison of tree generation algorithms. In *Proceedings of the 6th Annual Conference in Genetic Programming (GECCO-2001)*. Springer-Verlag.

[Malaiya et al., 2002] Malaiya, Y. K., Li, M. N., Bieman, J., and Karcich, R. (2002). Software reliability growth with test coverage. *IEEE Trans. on Reliability*, 51(4):420–426.

[Maldonado et al., 1992] Maldonado, J., Chaim, M., and Jino, M. (1992). Briding the gap in the presence of infeasible paths: Potential uses testing criteria. In *XII International Conference of the Chilean Science Computer Society*, pages 323–340. IEEE Computer Society, Santiago, Chile.

[Michalewicz and Schoemauer, 1996] Michalewicz, Z. and Schoemauer, M. (1996). Evolutionary Algorithms for Constrained Parameter Optimization Problems. *Evolutionary Computation*.

[Minglei, 2002] Minglei, B. S. D. (2002). Time series predictability. Master's thesis, Marquette University, Wisconsin.

[Moranda, 1975] Moranda, P. (1975). Predictions of software reliability during debugging. In *Proceddings of the Annual Reliability Maintainability Symposium*.

[Moranda and Jelinski, 1972] Moranda, P. and Jelinski, Z. (1972). Final report on software reliability study. Technical report, McDonnall Douglas Astronautics Company.

[Morettin and Toloi, 1981] Morettin, P. A. and Toloi, C. M. C. (1981). Modelos para previsão de séries temporais. *13o. Colóquio Brasileiro de Matemática*.

[Mueller, 1996] Mueller, A. (1996). Uma aplicação de redes neurais artificiais na previsão do mercado acionário. Master's thesis, Departamento de Engenharia de Produção - Universidade Federal de Santa Catarina (UFSC). Florianópolis, Brasil.

[Musa, 1975] Musa, J. (1975). A theory of software reliability and its application. *IEEE Transactions on Software Engineering*, pages 312–327.

[Musa, 1980] Musa, J. (1980). Software reliability data. Data & Analysis Center for Software.

[Nelson, 1973] Nelson, C. R. (1973). *Applied Time Series Analysis*. Holen-Day.

[Pasquine et al., 1996] Pasquine, A., Crespo, A. N., and Matrella, P. (1996). Sensitivity of reliability growth models to operational profile erros vs testing accuracy. *IEEE Trans. on Reliability*, 45(4):531–540.

[Povinelli, 1999] Povinelli, R. J. (1999). *Time Series Data Mining: Identifying Temporal Patterns for Characterization and Prediction of Time Series Events*. PhD thesis, Marquette University, Milwaukee, USA.

[Rapps and Weyuker, 1985] Rapps, S. and Weyuker, E. (1985). Selecting software test data using data flow information. *IEEE Trans. on Soft. Engineering*, SE-11(4):367–375.

[Rechenberg, 1973] Rechenberg, I. (1973). Evolutionsstrategie: Optimierung technischer systeme nach prinzipien der biologischen evolution.

[Refenes, 1993] Refenes, A. N. (1993). Financial modeling using neural networks: A comparative study with regression models. *Department of Computer Science - University College London*, pages 1–18.

[Schwefel, 1975] Schwefel, H. (1975). *Evolutionsstrategie und numerische Optimierung*. PhD thesis, Technische Universitat Berlin. Berlin, Germany.

[Schwefel, 1977] Schwefel, H. (1977). Numerische optimierung von computer-modellen mittels der evolutionsstrategie. *Interdisciplinary Systems Research*, 26.

[Schwefel, 1981] Schwefel, H. (1981). *Numerical optimization for computer models*. Wiley, Chichester.

[Silva and Almeida, 2003] Silva, S. and Almeida, J. S. (2003). Dynamic maximum tree depth - a simple technique for avoiding bloat in tree-based gp. In *E. Cantú-Paz, J. A. Foster, K. Deb, et. al., editors*. GECCO-2003 - Genetic and Evolutionary Computation Conference, Chicago, IL, USA.

[Silva and Costa, 2005] Silva, S. and Costa, E. (2005). Resource-limited genetic programming: The dynamic approach. In *GECCO-2005 - Genetic and Evolutionary Computation Conference*, pages 1673–1680. Association for Computing Machinery, Washington, D.C., USA.

[Silva et al., 2005] Silva, S., Silva, P. J. N., and Costa, E. (2005). Resource-limited genetic programming: Replacing tree depth limits. In *B. Ribeiro, R. F. Albrecht, A. Dobnikar, et. al., editors*, pages 243–246. ICANNGA-2005, Coimbra, Portugal.

[Sitte, 1999] Sitte, R. (1999). Comparison of software reliability growth predictions: Neural networks vs parametric recalibration. *IEEE Transactions on Software Engineering*, 48(3):285–291.

[Soule et al., 2002] Soule, T., Heckendorn, R., and Shen, J. (2002). Solution stability in evolutionary computation. In *Proceddings of the 17th International Symposium on Computer and Information Systems*, pages 237–241.

[Souza, 2004] Souza, G. (2004). Utilizando redes neurais artificiais para modelar a confiabilidade de software. Master's thesis, Departamento de Informática – Universidade Federal do Paraná (UFPR), Brasil.

[Souza and Vergilio, 2006] Souza, G. and Vergilio, S. (2006). Modeling software reliability growth with artificial neural networks. In *IEEE Latin American Test Workshop*. Buenos Aires, Argentina.

[Souza et al., 2005a] Souza, L. V., Costa, E. O., and Pozo, A. (2005a). Análise da capacidade da programação genética na previsão de séries temporais. In *SEMNI'05 - Congreso de Métodos Numéricos in Ingeniería*, Granada, Espanha.

[Souza et al., 2005b] Souza, L. V., Pozo, A., and Costa, E. O. (2005b). Previsão de séries temporais utilizando programação genética e combinação de preditores. In *XXXVII SBPO - Simpósio Brasileiro de Pesquisa Operacional*, Gramado, Rio Grande do Sul, Brasil.

[Souza, 1989] Souza, R. C. (1989). Modelos estruturais para previsão de séries temporais: Abordagens clássica e bayesiana. In *17o. Colóquio Brasileiro de Matemática*. Rio de Janeiro, Brasil.

[Stevens et al., 2005] Stevens, J., Heckendorn, R. B., and Soule, T. (2005). Exploiting disruption aversion to control code bloat. In *GECCO-2005 - Genetic and Evolutionary Computation Conference*, pages 1605–1612. Association for Computing Machinery, Washington, D.C., USA.

[Trivedi, 2001] Trivedi, K. S. (2001). *Probability and Statistics with Reliability Queuing and Computer Science Applications*. Wiley.

[Venables et al., 2005] Venables, W. N., Smith, D. M., and the R Development Team (2005). *An introduction to R*. R Development Core Team.

[Wagner and Michalewicz, 2001] Wagner, N. and Michalewicz, Z. (2001). Genetic programming with efficient population control for financial time series prediction. In *E. D. Goodman, editor*, pages 458–462. GECCO-2001 - Genetic and Evolutionary Computation Conference, San Francisco, CA, USA.

[Wheelwright and Makridakis, 1985] Wheelwright, S. C. and Makridakis, S. (1985). *Forecasting Methods for Management*. John Wiley and Sons Inc.

[Zongker and Punch, 1995] Zongker, D. and Punch, B. (1995). *Lil-gp 1.0 User's Manual*. Michigan State University.