Notas em Matemática Aplicada e-ISSN 2236-5915

Volume 2, 2012

Editado por

Eliana X.L. de Andrade
Universidade Estadual Paulista - UNESP
São José do Rio Preto, SP, Brasil

Rubens Sampaio
Pontifícia Universidade Católica do Rio de Janeiro -
Rio de Janeiro, RJ, Brasil

Geraldo N. Silva
Universidade Estadual Paulista - UNESP
São José do Rio Preto, SP, Brasil

Sociedade Brasileira de Matemática Aplicada e Computacional

2012

A Sociedade Brasileira de Matemática Aplicada e Computacional - SBMAC publica, desde as primeiras edições do evento, monografias dos cursos que são ministrados nos CNMAC.

Para a comemoração dos 25 anos da SBMAC, que ocorreu durante o XXVI CNMAC em 2003, foi criada a série **Notas em Matemática Aplicada** para publicar as monografias dos minicursos ministrados nos CNMAC, o que permaneceu até o XXXIII CNMAC em 2010.

A partir de 2011, a série passa a publicar, também, livros nas áreas de interesse da SBMAC. Os autores que submeterem textos à série Notas em Matemática Aplicada devem estar cientes de que poderão ser convidados a ministrarem minicursos nos eventos patrocinados pela SBMAC, em especial nos CNMAC, sobre assunto a que se refere o texto.

O livro deve ser preparado em **Latex (compatível com o Miktex versão 2.7), as figuras em eps** e deve ter entre **80 e 150 páginas**. O texto deve ser redigido de forma clara, acompanhado de uma excelente revisão bibliográfica e de **exercícios de verificação de aprendizagem** ao final de cada capítulo.

Veja todos os títulos publicados nesta série na página
http://www.sbmac.org.br/notas.php

FUNDAMENTOS, POTENCIALIDADES E APLICAÇÕES DE ALGORITMOS EVOLUTIVOS

MINICURSO DO XXVI CNMAC

Leandro dos Santos Coelho
Laboratório de Automação e Sistemas, Grupo Produtrônica
Programa de Pós-Graduação em Engenharia de Produção e Sistemas
Pontifícia Universidade Católica do Paraná - PUCPR/CCET/PPGEPS/LAS
Rua Imaculada Conceição, 1155, Prado Velho
CEP 80215-901 - Curitiba - PR, Brasil
http://www.produtronica.pucpr.br/leandro
email: lscoelho@rla01.pucpr.br

Sociedade Brasileira de Matemática Aplicada e Computacional

São Carlos - SP, Brasil
2012

Coordenação Editorial: Geraldo Nunes Silva

Coordenação Editorial da Série: Geraldo Nunes Silva

Editora: SBMAC

Capa: Matheus Botossi Trindade

Patrocínio: SBMAC

Catalogação elaborada pela Biblioteca do IBILCE/UNESP
Bibliotecária: Maria Luiza Fernandes Jardim Froner

Coelho, Leandro dos Santos.
Fundamentos, Potencialidades e Aplicações de Algoritmos Evolutivos - São Carlos, SP: SBMAC, 2012, 103 p., 20.5 cm -(Notas em Matemática Aplicada; v. 2)

e-ISBN 978-85-86883-71-2

1. Otimização 2. Algoritmos Genéticos
3. Matemática Aplicada. 4. Computação Científica.
I. Coelho, Leandro dos Santos. II. Título. III. Série.

CDD - 51

Esta é uma republicação em formato de e-book do livro original do mesmo título publicado em 2003 nesta mesma série pela SBMAC.

PREFÁCIO DA SÉRIE

A Sociedade Brasileira de Matemática Aplicada e Computacional - SBMAC, desde a realização do primeiro CNMAC - Congressso Nacional de Matemática Aplicada e Computacional, publica monografias de cursos que são ministrados durante o Evento. A atual diretoria decidiu fazer a indexação bibliográfica dessas monografias através do ISBN para efeito de catalogação para preservação das mesmas para a memória dos CNMAC.

A coleção recebeu o título de "Notas em Matemática Aplicada"e consistirá das monografias dos cursos ministrados nos CNMACs. O livro correspondente a cada minicurso deve ser preparado em até 100 páginas, para servir como texto introdutório, de modo que é aconselhável que contenha uma boa revisão bibliografica e exercícios de verificação de aprendizagem. A clareza do texto é um dos fatores mais importantes.

A coleção incluirá, gradativamente, os textos dos Encontros Regionais de Matemática Aplicada e Computacional, os ERMACs e de outros eventos patrocindos pela SBMAC.

Além disso, será objeto desta coleção publicar volumes com coletâneas de preprints de trabalhos apresentados em reuniões científicas patrocinadas pela SBMAC.

Esta primeira coleção, composta das monografias dos minicursos do XXVI CNMAC, foi especialmente preparada em comemoração aos seus 25 anos da SBMAC.

E. X. L. de Andrade
R. Sampaio
G. N. Silva

Prefácio

Os métodos de otimização e busca estocástica são baseados nos princípios da evolução biológica natural. Estes métodos têm recebido interesse crescente nas últimas décadas, devido principalmente a sua versatilidade na resolução de problemas complexos, nas áreas de otimização e aprendizado de máquina.

Os algoritmos evolutivos (ou algoritmos evolucionários) — metodologias da área computação evolutiva (ou evolucionária) — não são algoritmos computacionais em seu significado usual, mas formam uma classe de métodos regidos por princípios similares. Estes princípios oriundos do "mundo biológico" são baseados na teoria da evolução Darwiniana. Os algoritmos evolutivos tentam abstrair e imitar alguns dos mecanismos evolutivos à resolução de problemas que requerem adaptação, busca e otimização.

O objetivo deste livro é servir como texto introdutório aos algoritmos evolutivos. Neste contexto, o livro apresenta um apanhado de fundamentos, potencialidades e limitações de diversas abordagens de algoritmos evolutivos. Além disso, é apresentado um panorama do estado da arte das aplicações dos algoritmos evolutivos na academia e no meio industrial.

São José do Rio Preto, 30 de junho de 2003.

Leandro dos Santos Coelho

Agradecimentos

Às pessoas que no desenvolvimento deste trabalho contribuíram de diferentes formas para sua realização, em especial à Profa. Viviana Cocco Mariani, minha esposa, pela paciência, amor e permanente incentivo.

Ao Prof. Antonio Augusto Rodrigues Coelho, da Universidade Federal de Santa Catarina, pela amizade e contínuo incentivo.

Aos organizadores do XXVI CNMAC e aos membros da Sociedade Brasileira de Matemática Aplicada e Computacional - SBMAC por aceitar este minicurso. Em especial aos Prof. Geraldo Nunes Silva, Profa. Cleonice de Fátima Bracciali, Prof. Antonio Leitão e o Prof. Rubens Sampaio pelo apoio e assessoria.

À minha esposa Viviana.
Dedico

Conteúdo

Capítulo 1

Fundamentação biológica e breve histórico dos algoritmos evolutivos

1.1 Introdução

Os métodos de otimização e busca estocástica baseados nos princípios e modelos da evolução biológica natural têm recebido crescente interesse nas últimas décadas, devido principalmente a sua versatilidade para a resolução de problemas complexos, nas áreas de otimização e aprendizado de máquina. O desenvolvimento de modelos computacionais, inspirados nos mecanismos evolutivos, caracteriza-se pela configuração de algoritmos de otimização robustos e sistemas adaptativos [31].

Os algoritmos evolutivos (AEs) — metodologias da área computação evolucionária ou evolutiva (CE) — não são algoritmos computacionais em seu significado usual, mas formam uma classe de métodos regidos por princípios similares. Estes princípios oriundos do "mundo biológico" são baseados na teoria da evolução Darwiniana [35]. Os AEs tentam abstrair e imitar alguns dos mecanismos evolutivos à resolução de problemas que requerem adaptação, busca e otimização.

Nos AEs, os pontos no espaço de busca são modelados através de indivíduos que interagem em um ambiente artificial. Um conjunto de soluções (população) é manipulado a cada iteração, em contraste com outros métodos de otimização, onde apenas uma solução para o problema é utilizada. A chance que um indivíduo da população seja selecionado na próxima geração depende da função de aptidão (existem outros sinônimos, utilizados na literatura, tais como: função de adequação, adequabilidade ou *fitness*) do indivíduo, que consiste, geralmente, de uma função objetivo ou mesmo uma transformação simples desta para geralmente a resolução de um problema de maximização ou minimização de algum(ns) índice(s) de desempenho.

Os AEs não requerem o conhecimento das características do problema de otimização e não dependem de certas propriedades da função objetivo, tais como convexidade ou diferenciabilidade. Os AEs são guiados pela avaliação da função de aptidão dos indivíduos e possuem pouca dependência do tipo de problema que está sendo resolvido.

Os AEs são especialmente úteis às tarefas de otimização global, onde os métodos determinísticos podem levar a soluções de mínimos locais. Por conseguinte, os AEs são aptos à resolução de um amplo espectro de problemas não lineares, descontínuos, discretos, multivariáveis, entre outros.

O ciclo básico dos dados num AE pode ser sintetizado nos seguintes passos:
(i) inicialização aleatória da população de soluções;
(ii) avaliação da função de aptidão;
(iii) seleção dos indivíduos mais aptos de acordo com uma estratégia de seleção;
(iv) aplicação dos operadores de recombinação e mutação;
(v) geração de uma nova população de soluções candidatas;
(vi) repetição dos passos (ii) a (v) até que uma condição de parada seja satisfeita.

1.1.1 Breve descrição dos princípios evolutivos naturais

O propósito desta seção é apresentar uma breve abordagem dos princípios evolutivos naturais, que fundamentam os AEs. Mais detalhes podem ser encontrados na seguinte literatura: [76], [63], [36], [110], [52] e [6].

O conjunto mais aceito de teorias evolutivas é o paradigma neo-Darwiniano. Os argumentos deste paradigma defendem que a maior parte da história da vida animal considera a operação de procedimentos físicos em populações e espécies. Estes procedimentos abordam a evolução como uma interação de quatro procedimentos essenciais: reprodução, competição, mutação e seleção [52], [6].

A reprodução é uma propriedade óbvia das espécies existentes. As espécies, que apresentam maior potencial reprodutivo, têm o crescimento populacional com proporção exponencial, se todos os indivíduos destas espécies reproduzirem-se com sucesso. A reprodução realiza a transferência do código genético do indivíduo (assexuadamente ou sexuadamente) para a progênie. Através da reprodução, os indivíduos com maior aptidão são copiados várias vezes na próxima geração, enquanto os indivíduos com menor aptidão recebem poucas cópias (ou mesmo nenhuma).

A mutação é responsável pelo fato que os erros de replicação, durante a transferência de código genético, necessariamente ocorrem. A competição é uma conseqüência de populações em expansão em um ambiente de recursos finitos. A seleção é o resultado decorrente da competição que influencia a forma que as espécies utilizam os recursos disponíveis [6].

Os indivíduos e as espécies podem ser vistas como uma dualidade de seus códigos genéticos, o *genótipo*, e a expressão de suas características comportamentais, o *fenótipo*. Os resultados das variações genéticas são geralmente imprevisíveis, devido aos efeitos de *pleiotropia* e *poligenia*. A *pleiotropia* é o efeito onde um gene pode afetar simultaneamente diversas características fenotípicas. A *poligenia* é o efeito

onde uma característica fenotípica simples pode ser determinada pela interação simultânea de muitos genes [51].

A evolução é um procedimento que opera em cromossomos a nível genotípico e pode ser vista, do ponto de vista matemático, como um procedimento de otimização. A seleção natural relaciona os cromossomos com o desempenho de suas estruturas codificadas, ou seja, a seleção direciona os fenótipos para um valor tão próximo, quanto possível, do ótimo, dadas as condições iniciais e as restrições ambientais.

A seleção natural influencia os cromossomos, que codificam estruturas bem sucedidas, a reproduzirem mais freqüentemente que aqueles que não as codificam. Entretanto, o ambiente natural está em mudança contínua e nenhum indivíduo pode ser visto como perfeitamente adaptado ao seu ambiente, pois o indivíduo está em constante evolução para um "novo" ótimo.

1.1.2 Inspiração biológica para a evolução e o aprendizado

Os aspectos relacionados à inspiração biológica da evolução e o aprendizado são resumidos nesta seção. Estes aspectos servem de inspiração ao desenvolvimento dos AEs e suas variantes e têm muitos de seus fundamentos baseados em estudos de Charles Darwin, Francis Galton, Jean-Baptiste de Lamarck, Gregor Mendel, James Baldwin, Hugo De Vries, Carl Correns, Erich von Tschermack, Walter Sutton, Thomas Morgan, James Watson, Ernst Mayr, Frederick Sanger, entre outros.

No século XIX, Darwin (1809-1882) da mesma forma que Lamarck (1744-1829) observou a adaptação e a adequação entre a forma e o desempenho dos órgãos dos seres vivos e, sua maneira de viver. A diferença é que Darwin apresentou em suas pesquisas um mecanismo de evolução baseado na seleção natural. Entretanto, a proposta de Lamarck defendia o mecanismo de evolução através da hereditariedade, das características adquiridas diretamente durante a vida, também denominada de lei do uso e desuso [40]. Em síntese, Lamarck propôs que mudanças ambientais através da vida de um organismo causam mudanças estruturais, que são transmitidas ao seu descendente.

Enquanto, até hoje, os biologistas não acreditam na plausibilidade biológica desta teoria, o poder da teoria Lamarckiana pode ser ilustrado pela evolução da nossa sociedade. Neste caso, as idéias e o conhecimento são passados de geração para geração através da estruturação da linguagem e da cultura [85], [169].

No século XX, Mendel (1822-1884) propôs uma teoria de hereditariedade e dominância baseada em genes fatores que se encontravam nos gametas. Atualmente, os fatores mendelianos são denominados *genes*. As duas leis de Mendel procuram explicar a troca de material genético através de operações de cruzamento e mutação. O enunciado das leis de Mendel podem ser apresentados assim [101]:

1^a lei de Mendel: "*Cada caráter é determinado por um par de fatores que se separam na formação dos gametas, indo um fator do par para cada gameta, que é, portanto, puro*".

2^a lei de Mendel: "*Um par de alelos localizados em um par de cromossomos homólogos separa-se independentemente de outro par de alelos localizado em outro par de cromossomos homólogos*".

A palavra *caráter* ou *característica* é utilizada em Genética para designar qualquer particularidade de um indivíduo. Um mesmo caráter pode apresentar duas ou mais variáveis, e a variável de cada caráter é denominada *fenótipo*. O termo *fenótipo* pode ser aplicado tanto ao conjunto das variáveis dos caracteres manifestados em um organismo como à variável de cada caráter em particular, em outras palavras, *genótipo + meio = fenótipo*. O termo *genótipo* pode ser aplicado tanto ao conjunto total de genes de um indivíduo como a cada gene em particular [101].

Outra pesquisa relevante foi a de Baldwin, que acreditava no aprendizado adquirido, através das interações dos indivíduos com o meio, poderia ser transmitido a gerações futuras de forma indireta. A abordagem de Baldwin contraria a proposta de Lamarck, de aprendizado direto.

Os estudos de De Vries (1848-1935) são relevantes quanto a análise da forma como a mutação acontece na natureza. A reunião destas teorias em torno da denominação de *neo-Darwinismo* ou *teoria sintética da evolução* deu-se em uma conferência internacional em Princeton, nos Estados Unidos, em janeiro de 1947, onde estavam geneticistas, naturalistas e paleontólogos [40].

Richard Dawkins [37] apresenta a evolução sob uma perspectiva diferente, onde a unidade fundamental da seleção natural é o gene, mais que o próprio indivíduo. Esta visão não é contrastante com alguns dos princípios da evolução natural, mas providencia uma forma de interpretação alternativa, que é formalizada na teoria do gene egoísta.

Outra contribuição de Dawkins é a utilização do termo *meme*, para referir-se a uma idéia. Dawkins elaborou uma teoria para estudar e explorar os três fenômenos, que são únicos, para a evolução cultural. Segundo Gabora [57], o primeiro é o operador baseado em conhecimento: o cérebro detecta a regularidade e constrói esquemas que este operador utiliza para adaptar-se aos equivalentes mentais da mutação e da recombinação para os seus memes fundamentais. O segundo é a imitação: as idéias se propagam quando os membros de uma sociedade observam e copiam um ao outro. Isto é observado nas sociedades animais e humanas. A imitação habilita os indivíduos a compartilhar as soluções completas ou parciais dos problemas que eles defrontam-se. O terceiro é a simulação mental: os indivíduos podem imaginar o que poderia acontecer se um meme fosse implementado antes que os recursos fossem passados para ele. Este mecanismo provê os indivíduos de uma forma rudimentar de seleção.

1.2 Breve histórico da computação evolutiva

As origens da CE podem ser traçadas por trabalhos pioneiros de R. M. Friedberg, H. J. Bremermann, W. Spendley, F. E. Satterthwaite, entre outros, nos anos 50. Após esta fase, este campo do conhecimento permaneceu relativamente desconhecido ou inexplorado pela maioria da comunidade científica, por mais de três décadas. Este fato deve-se, principalmente, à falta de plataformas computacionais poderosas naquela época, da formalização e caracterização deficiente de cada metodologia evolutiva nos primeiros estudos nesta área [51], [6].

Os trabalhos fundamentais de Holland, Rechenberg, Schwefel e Fogel, serviram à realização de mudanças neste cenário, durante a década de 70. Atualmente, observa-se um relevante e permanente crescimento do número de publicações [3], aplicações e conferências no campo da CE. A maioria das abordagens correntes dos AEs descende dos princípios de diferentes metodologias, principalmente [26]:

(i) *algoritmos genéticos*, desenvolvidos principalmente por A. S. Fraser, H. J. Bremermann, J. Reed e J. H. Holland, entre a década de 50 e 70, com refinamentos posteriores por D. Whitley, D. E. Goldberg, K. De Jong e J. Grefenstette;

(ii) *programação evolutiva*, desenvolvidas por L. J. Fogel, A. J. Owens e M. J. Walsh, nos Estados Unidos, na década de 60, refinada recentemente por D. B. Fogel, G. H. Burgin, P. J. Angeline, V. W. Porto e W. Atmar;

(iii) *estratégias evolutivas*, desenvolvidas na Alemanha, por I. Rechenberg e H. P. Schwefel, na década de 60, com aprimoramentos posteriores de G. Rudolph, H. G. Beyer, F. Kursawe e T. Bäck;

(iv) *programação genética*, abordadas pelos pesquisadores J. R. Koza, J. P. Rice, K. E. Kinnear e P. J. Angeline.

Um fato relevante é que durante muito tempo as metodologias que constituem hoje os AEs foram desenvolvidas independentemente uma das outras. Um esforço organizado, de forma a criar-se um fórum de interação das pesquisas entre as várias comunidades de pesquisadores de AEs, emergiu em 1990. O evento foi o *workshop* internacional intitulado *Parallel Problem Solving from Nature* (PPSN'91) realizado em Dortmund, na Alemanha. Após este evento, outros eventos significativos (ICGA'91, Alife'91, EP'92, PPSN'92 e WCCI'94) e algumas publicações (*Evolutionary Computation*, da MIT Press, 1993, *BioSystems*, da Elsevier, e *IEEE Transactions on Evolutionary Computation*, 1997) levaram a consenso a denominação de CE, a este novo campo do conhecimento.

Entre os anos de 80 e 90, os avanços no desempenho das plataformas computacionais habilitaram a aplicação de AEs à resolução de problemas de otimização do mundo real e em projetos relevantes. Entre os projetos relevantes relativos a aplicação de AEs na indústria deve-se citar a *EvoNet*, uma rede européia de excelência em CE, fundada pela comissão européia ESPRIT IV. A EvoNet é responsável pela disseminação e colaboração ativa entre pesquisadores da comunidade européia e a transferência de tecnologia que utiliza os AEs, para os ambientes industrial e comercial.

As áreas de aplicação abrangem: telecomunicações, escalonamento, robótica móvel, manufatura, reconhecimento de faces, identificação, previsão, controle, sistemas de potência, configuração de hardware, processamento de imagens, otimização de forma e processamento de sinais. Entre os membros da EvoNet estão: Bristish Aerospace, Daimler-Benz, Dassault Aviation, Hewlett Packard Laboratories, Institut Français du Petrole, Rolls-Royce, SGS-Thomson e Siemens [46], [47], [143].

1.3 Por que utilizar os algoritmos evolutivos?

Os AEs são técnicas robustas e eficientes em espaços de procura irregulares, complexos e apresentando múltiplas dimensões. Um AE caracteriza-se por [63]:

(i) operar em uma população de pontos;

(ii) não requerer cálculos de derivadas e informação sobre o gradiente da função objetivo;

(iii) trabalhar com a codificação de seu conjunto de parâmetros, não com os próprios parâmetros (representação binária);

(iv) realizar transições probabilísticas, em vez de regras determinísticas;

(v) necessitar apenas da informação sobre o valor da função objetivo para cada indivíduo da população;

(vi) apresentar simplicidade conceitual;

(vii) ser pouco afetado, quanto à eficiência, quando descontinuidades e ruídos estão presentes nos dados do problema.

As características (iii) a (v) não são comuns a todos os AEs, mas geralmente presentes nos algoritmos genéticos.

1.3.1 Potencialidades dos algoritmos evolutivos

Quanto ao projeto e à configuração de algoritmos de otimização e aprendizado de máquina (*machine learning*), os AEs são empregados com sucesso devido as seguintes características [26]:

(i) tratarem adequadamente os sistemas sujeitos a restrições;

(ii) não requererem as informações relativas a derivadas, estas usualmente necessárias em métodos convencionais de otimização;

(iii) adequarem-se à implementação em paralelo e distribuídas;

(iv) possibilitarem a utilização do conhecimento obtido *a priori* pelo projetista; e

(v) tratarem com sistemas complexos e espaços de busca com múltiplas modas e/ou múltiplos objetivos.

1.3.2 Limitações dos algoritmos evolutivos

Os AEs também apresentam limitações e que devem ser mencionadas. Os AEs tratam-se de métodos estocásticos e seu desempenho varia de execução para execução (a menos que o mesmo gerador de números aleatórios com a mesma semente é utilizado). Devido a isto, a média da convergência sobre diversas execuções do AE é um indicador de desempenho mais útil que uma simples execução.

Os AEs, nas suas configurações usuais, também apresentam dificuldades para a determinação do ótimo global, sem a utilização de uma metodologia de otimização local. Outra limitação é a necessidade de análise de todas as amostras do processo a cada avaliação da função de aptidão. Este aspecto é uma limitação relevante para aplicações de controle em tempo real.

1.3.3 Teorema *no free lunch*

Uma observação relevante quanto aos AEs é apresentada no teorema *no free lunch* (NFL) [171]: não existe algoritmo para a resolução de todos problemas de otimização que seja genericamente (em média) superior que outro algoritmo competidor. Segundo o NFL, a afirmação de que os AEs são inferiores ou superiores a algum método alternativo é "insensata". O que pode ser afirmado somente é que AEs comportam-se melhor que outros métodos com respeito a resolução de uma classe específica de problemas, e como conseqüência comportam-se inadequadamente para outras classes de problemas.

O teorema NFL pode ser confirmado pela análise dos AEs em relação a muitos métodos clássicos de otimização. Os métodos clássicos são mais eficientes para resolução de problemas lineares, quadráticos, fortemente convexos, unimodais, separáveis, e em muitos outros problemas em especial. Por outro lado, os AEs têm sido utilizados nos mais diversos problemas principalmente quando estes são descontínuos, não diferenciáveis, multimodais, ruidosos, e quando superfícies de resposta não convencionais são envolvidas [6], [171].

As conseqüências deste fato são a existência de uma dicotomia entre eficiência e aplicabilidade geral, entre propriedades de convergência e esforço computacional dos algoritmos para a resolução de problemas. Algum conhecimento específico sobre a situação que está sendo tratada pode ser utilizado para especificação de um algoritmo adequado a solução do problema. Entretanto, não existe um método que resolva todos os problemas efetivamente tão bem quanto eficientemente, pois estas metas são contraditórias [26].

1.4 Exercícios

1. O que são algoritmos evolutivos?
2. Qual as diferenças entre os algoritmos evolutivos e outros algoritmos (convencionais) de otimização?
3. Quais são os paradigmas computacionais que compõem a área denominada computação evolutiva (ou evolucionária)?
4. Quais são as potencialidades que caracterizam os algoritmos evolutivos?
5. Cite exemplos de possíveis aplicações de algoritmos evolutivos.

Capítulo 2

Algoritmos genéticos, programação genética e sistemas classificadores

2.1 Introdução

John Holland e alguns de seus colaboradores da University of Michigan estavam interessados em sistemas complexos artificiais, capazes de adaptarem-se a mudanças de condições ambientais. Sob este ponto de vista, a necessidade da estruturação de sistemas com mecanismos de auto-adaptação é enfatizada. Uma população de indivíduos, para adaptar-se coletivamente em um ambiente, deve comportar-se como um sistema natural, onde a sobrevivência é promovida eliminando-se os comportamentos inúteis (ou prejudiciais) e recompensando-se os comportamentos úteis.

Holland [76] compreendeu que os mecanismos biológicos permitiam a adaptação do sistema natural biológico de forma que poderiam ser expressas matematicamente, e simuladas computacionalmente. Esta abstração originou os algoritmos genéticos (AGs). A ligação entre a busca atual, o problema de otimização e o AG é o indivíduo. Cada indivíduo representa uma solução factível em um mesmo espaço de busca, que pode ser verificado através de um mapeamento apropriado. O mapeamento do espaço de busca, para os indivíduos, e o mapeamento reverso foi realizado originalmente através de dígitos binários. Os *strings* de *bits* são formas gerais e permitem a análise de alguns resultados teóricos sobre os AGs. Contudo, a codificação binária não é sempre a melhor escolha e outras representações são possíveis.

Em síntese, Holland [76] formulou originalmente os AGs e providenciou os fundamentos teóricos para a representação binária. Trabalhos relevantes sobre outras representações também são encontrados na literatura em [63], [110], [6].

Os AGs possuem procedimentos probabilísticos de busca, baseados nos princípios decorrentes da dinâmica das populações naturais. Nos procedimentos de busca, uma população de soluções candidatas é aleatoriamente gerada e "evoluem" para

uma solução, através da aplicação de operadores genéticos. Os três operadores usualmente empregados em AGs são: seleção, *crossover* (cruzamento) e mutação.

Um AG é projetado, na sua configuração básica, conforme as seguintes etapas:

(i) criar a população inicial de parâmetros compreendendo *Nind* indivíduos (soluções para o problema). Cada uma das soluções consiste de vetores $x_i \in \{0,1\}$ (representação canônica) ou $x_i \in \Re$ (representação real). Estes parâmetros são inicializados aleatoriamente, de acordo com uma distribuição uniforme;

(ii) classificar cada solução x_i , i=1,...,*Nind*, com relação ao cálculo da função de aptidão, ou seja, avalia-se o grau de adaptação de cada indivíduo da população em relação ao problema;

(iii) selecionar os indivíduos mais aptos de acordo com uma estratégia de seleção;

(iv) aplicar o operador genético de cruzamento (representação canônica) ou recombinação (representação por ponto flutuante);

(v) aplicar o operador genético de mutação;

(vi) gerar uma nova população; e

(vii) repetir as etapas (ii) a (vi) até que um critério de convergência seja satisfeito.

As etapas mencionadas para os AGs podem ser sintetizadas pelo pseudocódigo do seu ciclo evolutivo apresentado a seguir [10].

$t := 0$;
iniciar $P(t)$: $P(0) := \{x_1(0), x_2(0), ..., x_\alpha(0)\}$
avaliar $P(0) := \{\Phi(x_1(0)), \Phi(x_2(0)), ..., \Phi(x_\alpha(0))\}$
enquanto uma condição de parada não é satisfeita
{
realizar *crossover*: $P(t) := c_{\Theta c}(P(t))$
realizar mutação: $P'(t) := m_{\Theta m}(P(t))$
avaliar $P'(t) := \{\Phi(x_1^{'}(t)), \Phi(x_2^{'}(t)), ..., \Phi(x_\alpha^{'}(t))\}$
selecionar $P(t+1) := s_{\Theta s}(P'(t))$
$t := t + 1$;
}

As convenções utilizadas no pseudocódigo apresentado são as seguintes:
x: indivíduo da população antiga;
x': indivíduo da população atual;
α: número de indivíduos da população;
$P(t)$:=$\{x_1(t), x_2(t), ..., x_\alpha(t)\}$: população na geração t;
$P'(t)$:=$\{x'_1(t), x'_2(t), ..., x'_\alpha(t)\}$: população atual na geração t;
$\Phi : I \longrightarrow \Re$: mapeamento da função de aptidão;
$m_{\Theta m}$: operador de mutação com parâmetro de controle Θ_m;
$c_{\Theta c}$: operador de cruzamento (ou recombinação) com parâmetro de controle Θ_c;
$s_{\Theta s}$: operador de seleção $\ni s_{\Theta s} : I^{Nind} \longrightarrow I^{Nind}$;
I: conjunto canônico, isto é, as variáveis com valores 0 ou 1.

O operador genético de *crossover* — ou de forma análoga, o operador de recombinação (para variáveis que pretencem ao domínio dos números reais) — é responsável pela troca de material genético entre os indivíduos, com maior probabilidade de reproduzirem os indivíduos mais aptos ao ambiente.

O operador de mutação modifica o valor dos genes do indivíduo e visa restaurar o material genético perdido ou não explorado em uma população. Este operador, quando projetado de forma apropriada, pode prevenir a convergência prematura do AG para soluções sub-ótimas e manter a diversidade da população.

A definição dos parâmetros intrínsecos aos AGs — parâmetros de controle do AG — geralmente são determinados heuristicamente, tais como: tamanho da população, tamanho da estrutura dos cromossomos, probabilidade de cruzamento, probabilidade de mutação e tipos dos operadores genéticos a serem adotados [52].

2.1.1 Operador de seleção

O operador de seleção emprega o princípio de sobrevivência dos indivíduos mais aptos, através de uma metáfora aos procedimentos de reprodução assexuada e seleção natural, de acordo com o grau de adaptação do indivíduo ao ambiente. O objetivo básico do operador de seleção é enfatizar as melhores soluções que constituem uma população. O operador não cria nenhuma nova solução. Este operador seleciona as soluções relativamente aptas de uma população e remove as soluções remanescentes.

O operador de seleção é uma combinação de dois conceitos diferentes: reprodução e seleção. As múltiplas cópias de uma solução (indivíduo) são alocadas em uma população pela remoção de algumas soluções inferiores. Entretanto, alguns estudos de AEs utilizam ambos conceitos de reprodução e seleção, simultaneamente, em um operador, e outros os utilizam separadamente [6].

A verificação se uma solução é apta (ou não) em uma população é baseada no valor da função de aptidão da referida solução. Para que uma solução tenha maior aptidão deve ter uma alta probabilidade de seleção. Contudo, os operadores de seleção diferem na maneira que as cópias são designadas para serem as melhores soluções.

Os operadores de seleção incluem a reprodução e caracterizam-se por um parâmetro denominado pressão seletiva, relacionado ao tempo de "preenchimento" do operador de seleção. O tempo de "preenchimento" é definido como a velocidade com que a melhor solução, na população inicial, pode ocupar toda a população através da aplicação repetida do operador de seleção [63].

O operador de seleção, quando apresenta uma grande pressão seletiva faz com que a população perca a diversidade rapidamente, ocasionando uma convergência prematura, para uma solução inadequada. Contudo, um operador de seleção com pressão seletiva pequena apresenta uma baixa convergência e permite aos operadores de *crossover* e mutação iterações suficientes à busca no espaço de soluções [64]. As noções de alguns operadores de seleção são apresentadas e discutidas a seguir.

- *seleção proporcional*: este mecanismo utiliza uma distribuição de probabilidade tal que a probabilidade de seleção de um dado indivíduo, para re-

produção, é proporcional à função de aptidão do indivíduo. Assim, obtida a função de aptidão, f_i, de cada indivíduo em uma dada geração, obtém-se

$$F_T = \sum_{i=1}^{Nind} f_i(x) \tag{2.1}$$

que representa a aptidão total da população. Após, a probabilidade de seleção, p_i, é atribuída para cada indivíduo pela equação:

$$p_i(x) = \frac{f_i(x)}{F_T}. \tag{2.2}$$

Finalmente, uma probabilidade acumulada para cada indivíduo é obtida pela soma das funções de aptidão dos membros da população com classificação inferior à sua, ou seja,

$$c_i = \sum_{k=1}^{i} p_k, \quad i = 1, ..., Nind \tag{2.3}$$

Um número r uniformemente distribuído em [0,1] é obtido *Nind* vezes e a cada tempo o i-ésimo *string* é selecionado tal que $c_{i-1} < r \leq c_i$. Quando $r < c_1$, o primeiro indivíduo é selecionado. Este procedimento pode ser visualizado por meio de uma roleta com *Nind* partes, onde cada parte tem tamanho proporcional à aptidão do indivíduo. Para ilustrar este exemplo, tomam-se valores de $p_1 = 0{,}10$; $p_2 = 0{,}40$; $p_3 = 0{,}30$ e $p_4 = 0{,}20$. Assim têm-se: $c_1 = 0{,}10$; $c_2 = 0{,}50$; $c_3 = 0{,}80$ e $c_4 = 1{,}00$. Agora, seja $r = 0{,}07$ o número gerado. Já que $r < c_1$, o indivíduo 1 será selecionado. Se r fosse 0,93 então o indivíduo 4 seria selecionado, pois $c_3 < r \leq c_4$. Um pseudocódigo da seleção proporcional é apresentado a seguir [15].

dados de entrada: a população *P(t)*
dados de saída: a população após a realização da seleção, *P'(t)*
torneio (P_1,...,P_{Nind})
  $c_0 \leftarrow 0$;
  para $i \leftarrow 1$ até *Nind* fazer
    $c_i \leftarrow c_{i-1} + f_i/F_T$;
  fim do para i
para $j \leftarrow 1$ até *Nind* fazer
  $r \leftarrow$ *aleatório* [0,1[
  $P_j^{'} \leftarrow P_l$ tal que $c_{l-1} \leq r \leq c_l$, onde *léé* o comprimento do *string*
fim do para j
retorna {P_1, P_2,...,P_{Nind}}

Uma variante da seleção proporcional é a seleção por *ranking*. Os métodos de *ranking* requerem somente o valor da função de aptidão, para mapear as soluções em um conjunto parcialmente ordenado. Um exemplo desta forma de seleção é a

seleção por *ranking* geométrico normalizado [82], onde a probabilidade de seleção de um indivíduo é dada pelas equações

$$p(x_i) = q'(1-q)^{ra-1} \tag{2.4}$$

$$q' = \frac{q}{1-(1-q)^{Nind}}, \quad i = 1, ..., Nind \tag{2.5}$$

onde q é a probabilidade de selecionar o melhor indivíduo e $ra(i)$ é o *rank* do indivíduo, onde que $ra(\cdot)$=1 é o melhor *rank*.

- *seleção elitista com truncamento*: o mecanismo de seleção elitista com truncamento consiste na seleção, em cada geração, dos b melhores indivíduos da população (b é o coeficiente de truncamento $\in$ [0;1]). Contudo, b tem valores típicos entre 0,1 e 0,5. Entre os b melhores indivíduos são mantidos os indivíduos mais aptos na próxima geração, e destes são selecionados os indivíduos que passarão pelas operações de *crossover* e mutação [39]. Um pseudocódigo da seleção elitista do tipo *breeder* com truncamento é apresentado a seguir.

dados de entrada: a população *P(t)* e $\beta \in [0;1]$
dados de saída: a população após a realização da seleção, *P'(t)*
elistista (β, P_1,...,P_{Nind})
  $\overline{P} \leftarrow$ população classificada de acordo com a aptidão;
  para $i \leftarrow 1$ até *Nind* fazer
    $r \leftarrow$ *aleatório* [0,1[
    $P_i' \leftarrow \overline{P}_r$
  fim do para i
retorna {P_1, P_2,...,P_{Nind}}

- *seleção por torneio*: nesta forma de seleção um grupo de q indivíduos é aleatoriamente escolhido. Esta forma de seleção pode ser projetada da população com ou sem reposição. O grupo ocupa parte de um torneio, onde o indivíduo vencedor é determinado de acordo com o valor da função de aptidão. O melhor indivíduo é escolhido deterministicamente, ainda que uma seleção estocástica possa ser realizada. Em ambos os casos, somente o vencedor é inserido na população da próxima geração e o procedimento é repetido k vezes para formar a nova população. Os torneios são realizados, freqüentemente, entre dois indivíduos (torneio binário). Contudo, isto pode ser generalizado para um grupo com tamanho arbitrário, q, do torneio. Um pseudocódigo da seleção através de torneio é apresentado a seguir [15].

dados de entrada: a população *P(t)* e $q \in \{1, 2, ..., Nind\}$
dados de saída: a população após a realização da seleção, *P'(t)*
torneio (q, P_1,...,P_{Nind})
  para k ← 1 até *Nind* fazer
    $P(x_k) \leftarrow$ indivíduo escolhido de q indivíduos de {P_1,...,P_{Nind}}
    $r \leftarrow$ *aleatório* [0,1[
  fim do para k
retorna {P_1, P_2,...,P_{Nind}}

Alguns operadores de seleção foram mencionados, mas outros tipos de seleção podem ser utilizados em AEs. Outros exemplos de seleção são extensões da seleção por roleta, *steady-state*, amostragem estocástica, Boltzmann, formas de truncamento e modelos elitistas [63], [6]. Os operadores de *crossover* e mutação diferem quanto à estrutura e implementação, de acordo com o tipo de AG adotado, seja este com representação inteira, canônica ou real. Estas representações são brevemente descritas a seguir.

2.1.2 Tipos de representação

Os AGs, com a representação inteira, são usualmente projetados para problemas de escalonamento e do tipo caixeiro viajante [110]. Os AG com a representação canônica e representação real têm sido aplicados nos mais diversos problemas de identificação e controle de processos. Os tradicionais AGs, denominados AGs canônicos, baseiam-se em noções do teorema de esquemas (*schema theorem*) e blocos de construção (*building blocks*). Os indivíduos são representados por vetores binários [76]. Entretanto, esta representação não é universalmente aceita na literatura. Alguns pesquisadores indicam que a representação de ponto flutuante (números reais) apresenta melhor desempenho em relação a representação binária em aplicações que necessitam do tratamento de valores reais, pois apresenta maior compatibilidade, precisão e rapidez de execução. A representação binária é apropriada em aplicações que requeiram o tratamento de valores discretos [36]. Contudo, o procedimento básico de otimização por AGs utiliza os três operadores básicos.

Algoritmo genético canônico

Os AGs canônicos constituem-se de uma população de *strings* de *bits* modificadas por operadores genéticos. Cada cromossomo é composto de N_{df} *substrings* constituídos de valores 0s e 1s. Os strings de comprimento $N_{bit(s)}$ (comprimento igual a 1), $N_{bit(i)}$ e $N_{bit(f)}$, são utilizados para codificar o *bit* de sinal, valores inteiros e fracionários das variáveis, x, da população, respectivamente. A figura 2.1 ilustra um cromossomo que abrange duas variáveis com representação binária e a decodificação de seu valor da base binária para a base decimal.

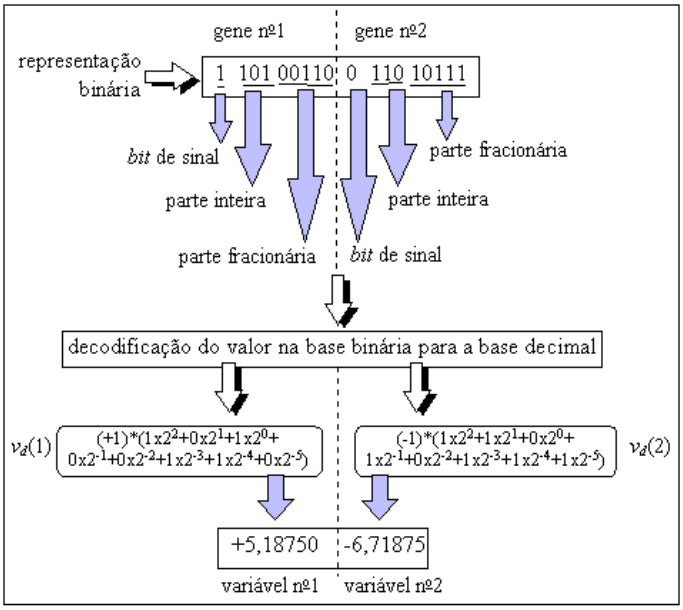

Figura 2.1: Representação de um cromossomo nos AGs canônicos

O cromossomo da figura 2.1 tem N_{df}= 2, $N_{bit(s)} = 1$, $N_{bit(i)} = 3$, e $N_{bit(f)}$= 5. Cada *substring* binário de comprimento $(N_{bit(s)} + N_{bit(i)} + N_{bit(f)})$ representa um número com a representação em ponto flutuante, que pode ser descrito por

$$v_d(k) = sgn(k) \left[\sum_{j=1}^{N_{bit(i)}} \delta_j^{(k)} 2^{N_{bit(i)}-j} + \sum_{i=1}^{N_{bit(f)}} \delta_{i+N_{bit(i)}}^{(k)} 2^{-N_{bit(i)}-N_{bit}(f)+i+2} \right] \quad (2.6)$$

para $k = 1, ..., N_{df}$ e $\delta_j^{(k)}$(0 ou 1) representa o j-ésimo locus (ou posição) do k-ésimo substring binário, *sgn(k)* representa um indicador do *bit* de sinal. Este *bit* pode ser igual 1 (*sgn(k)*=+1), ou caso contrário, quando o *bit* de sinal é igual 0 (*sgn(k)*=-1).

Cada *bit* (ou mesmo grupos de *bits*) representa um valor da mesma variável do problema (gene). No caso do exemplo apresentado na figura 2.1, os *strings* (000000000) e (111111111) representam os limites inferiores e superiores do intervalo de busca. Necessitando aumentar a precisão requerida, um *string* $N_{bit(f)}$ com comprimento maior apresenta maior precisão para representação dos cromossomos. Um caso particular da representação binária é a representação inteira que pode ser configurada quando $N_{bit(f)} = 0$.

Teoria de esquemas e blocos de construção: algumas definições Na representação binária dos indivíduos considera-se um alfabeto de símbolos {0,1, #}, onde {#} é um símbolo especial (*wild card*), que pode ser 0 ou 1. Um esquema é um

string com símbolos variáveis e fixos. Por exemplo, o esquema [01#1#0101] é um máscara que combina os seguintes *strings*: [010100101], [010110101], [011100101] e [011110101]. O símbolo # não é manipulado pelo AG canônico, pois constitui-se somente de um artificio de notação para facilitar a conceituação de grupos de *strings*.

Na proposta de Holland [76], cada *string*, avaliado em uma dada geração, dá uma informação parcial sobre a aptidão do conjunto de possíveis esquemas que o *string* é membro. Isto é a manifestação do denominado paralelismo implícito. Além disso, pode-se analisar a influência dos operadores de seleção, cruzamento e mutação de um número esperado de esquemas, quando passa-se de uma geração para a próxima. Os detalhes desta análise são relatados em [63] e [162]. A seguir são apresentados alguns fundamentos e aspectos relativos a sua importância para a teoria de AGs.

Considerando-se a seleção por roleta, m indivíduos em uma população com um esquema em particular H na geração $(t+1)$ é relacionado ao mesmo número na geração t, através de:

$$m(H, t+1) = m(H, t)\frac{f_H(t)}{\overline{f}(t)} \tag{2.7}$$

onde $f_H(t)$ é o valor da aptidão média do string que representa o esquema H, enquanto $\overline{f}(t)$ é o valor da aptidão média de todos *strings* da população. Assume-se que um esquema em particular permanece acima da média por uma determinada quantidade $\overline{cf}(t)$ para t gerações. Logo, a solução recorrente da equação 2.7 é apresentada na forma de uma equação de crescimento exponencial, isto é,

$$m(H, t) = m(H, 0)\,[1 + c]^t \tag{2.8}$$

onde abrange o número de esquemas H da população na geração 0, c é uma constante positiva e $t \geq 0$. A relevância deste resultado é que a reprodução proporcional a aptidão, aloca um número de tentativas com crescimento exponencial em média sobre a média do esquema.

Os operadores de cruzamento e mutação têm influência neste sentido. O efeito do cruzamento é diminuir o crescimento exponencial dos esquemas por uma quantidade que é proporcional a taxa de cruzamento, p_c, e depende do comprimento definido δ de um esquema e l do *string*:

$$pc\frac{\delta(H)}{l-1} \tag{2.9}$$

O comprimento definido d de um esquema é a distância entre a primeira e a última posição fixa de um *string*. Por exemplo, para o esquema [**0**1#1#0**1**##], δ = 7 -1 = 3 e para [###**1**00#**1**#], δ = 8 - 4 = 4. Intuitivamente, nota-se que um esquema com menor comprimento tem maior probabilidade de ser menos interrompido por uma operação de cruzamento, com um ponto de corte. Logo, o esquema superior a média, com comprimentos definidos pequenos, tem provavelmente uma taxa de crescimento exponencial. Estes esquemas com definição de

comprimento pequeno, acima da média, são denominados de blocos de construção e são um aspecto essencial na teoria que regem os AGs canônicos.

O efeito da mutação é simples de descrever. Se a probabilidade de mutação de um *bit* é p_m, então a probabilidade de sobrevivência de um simples *bit* é 1-p_m. Desde que mutações de bit são independentes, a probabilidade total é então $(1\text{-}p_m)^l$. Este número constitui-se na ordem *o(H)* de um esquema H e é igual a l menos o número de símbolos # do esquema.

Por exemplo, o esquema [**0**1#1#0**1**##] tem *o(H)* = 9 - 4 = 5 e [###**1**00#**1**#], *o(H)* = 9 - 5 = 4. Então a probabilidade de sobrevivência a uma mutação de um esquema H é $(1\text{-}p_m)^{o(H)}$, que para $p_m << 1$ pode ser aproximado por 1 - *o(H)*p_m.

Assim, colocando-se juntos os efeitos dos operadores de seleção proporcional, cruzamento e mutação obtém-se o teorema dos esquemas, proposto por Holland [76], através da relação:

$$m(H, t+1) \geq m(H, t)\frac{f_H(t)}{\overline{f}(t)}\left[1 - p_c\frac{\delta(H)}{l-1} - o(H)p_m\right] \tag{2.10}$$

Operador de cruzamento O operador de cruzamento é o operador principal dos AGs, e consiste na troca de partes dos cromossomos entre os indivíduos. O(s) ponto(s) de cruzamento são escolhidos aleatoriamente. As partes de dois cromossomos ancestrais, à direita do ponto de cruzamento são trocadas, para a formação dos cromossomos descendentes.

O cruzamento não é realizado em cada par de indivíduos, pois sua freqüência de utilização é controlada através da sua probabilidade (geralmente tem valor próximo de 1, ou seja, 100%). O operador de cruzamento com um e dois pontos de corte são representados na figura 2.2.

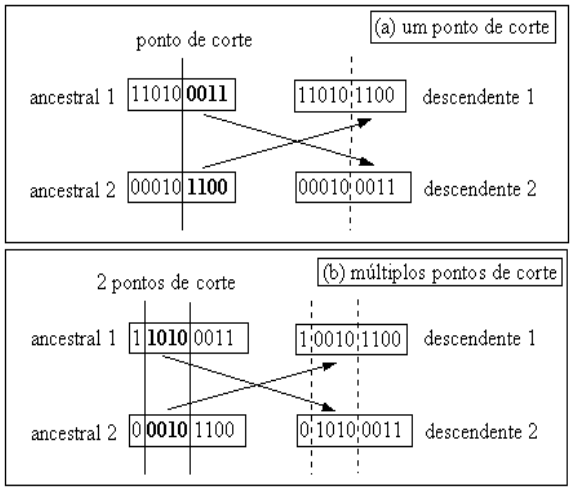

Figura 2.2: Operador de cruzamento com um ou dois pontos de corte

Outro operador de cruzamento é o cruzamento uniforme. No cruzamento uniforme não seleciona-se um conjunto de pontos de corte, mas considera-se cada posição dos *bits* dos dois ancestrais e troca-se dois *bits* com uma probabilidade de 50% (podem ser considerados outros valores). Assim, supondo-se que as posições 2 e 5 dos ancestrais são trocadas, conforme apresentado na figura 2.3.

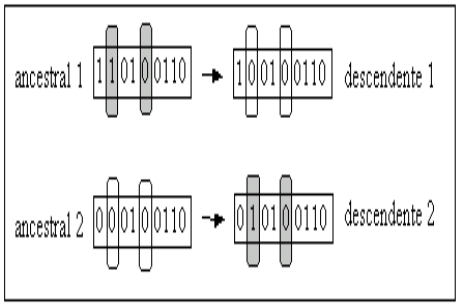

Figura 2.3: Operador de cruzamento uniforme

Operador de mutação O operador de mutação nos AGs é introduzido para prevenir a convergência prematura para ótimos locais, através da amostragem de novos pontos no espaço de busca. O operador de mutação é aplicado, a cada descendente individualmente, após a aplicação do operador de cruzamento. Consiste na mudança aleatória de uma parte do *string*, que representa o indivíduo (normalmente um *bit*), apresentando pequena probabilidade. O operador de mutação é representado na figura 2.4.

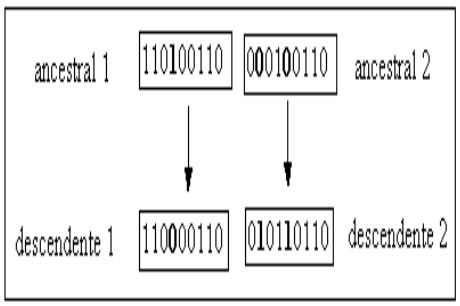

Figura 2.4: Operador de mutação nos AGs canônicos

Exemplo de aplicação do AG canônico: Otimização de uma função simples [110] A função é definida como $f(x) = x \cdot sen(10\pi \cdot x) + 1,0$ e apresentada graficamente a seguir na figura 2.5. O problema é encontrar x no intervalo [-1; 2] que maximiza a função $f(x)$, isto é, que encontra x_0 tal que

$$f(x_0) \geq f(x), \text{ para todo } x \in [-1, 2].$$

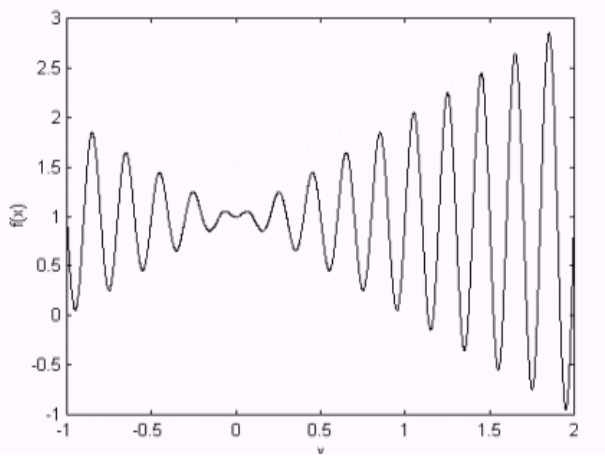

Figura 2.5: Gráfico da função $f(x) = x \cdot sen(10\pi \cdot x) + 1,0$

É relativamente fácil analisar a função $f(x)$. Os zeros das primeiras derivadas f' podem ser determinados por

$$f'(x) = sen(10\pi \cdot x) + 10\pi x \cdot \cos(10\pi x) = 0. \tag{2.11}$$

A fórmula 2.11 é equivalente a $\tan(10\pi \cdot x) = -10\pi x$ e possui um número infinito de soluções,

$$x_i = \frac{2i-1}{20} + \varepsilon_i, \quad \text{para } i\text{=1,2,...} \tag{2.12}$$

$$x_0 = 0 \tag{2.13}$$

$$x_i = \frac{2i+1}{20} + \varepsilon_i, \quad \text{para } i\text{=-1,-2,...} \tag{2.14}$$

onde os termos representam as seqüências decrescentes de números reais (para i=1,2,... e para i=-1,-2,...) que aproximam-se de zero.

Observa-se também que a função *f(x)* alcança seu **máximo local** para x_i se i é um inteiro ímpar e seu mínimo local para x_i se i é um inteiro par.

Desde que o **domínio do problema** é $x \in [-1, 2]$, a função alcança seu máximo para $x_{19} = \frac{2*19-1}{20} + \varepsilon_{19}$, ou seja, $x_{19} = \frac{37}{20} + \varepsilon_{19} = 1,85 + \varepsilon_{19}$, onde $f(x_{19}) = f(1,85) = 1,85 \cdot sen\left(18\pi + \frac{\pi}{2}\right) + 1,0 = 2,85$.

Assume-se que deseja-se construir um algoritmo genético para resolver o problema mencionado, isto é, **maximizar a função** *f(x)*. A seguir apresenta-se os passos para a resolução deste problema.

- *Representação das soluções*

Neste caso é utilizado um vetor binário como um cromossomo que representa os valores reais da variável x. O comprimento do vetor depende da precisão requerida. Neste exemplo supondo-se que são adotadas **6 casas** depois da vírgula decimal.

O domínio da variável x tem **comprimento 3**: a precisão requerida implica que o intervalo [-1; 2] pode ser dividido em pelo menos 3·1000000= 3000000 intervalos de tamanhos iguais. Isto significa que **22 *bits*** são **requeridos** como um vetor binário (cromossomo):

$$2097152 = 2^{21} < 3000000 \leq 2^{22} = 4194304 \tag{2.15}$$

O mapeamento do *string* binário $<b_{21}b_{20}\cdots b_0>$ em um número real x no intervalo [-1; 2] realizada em dois passos:

· `Passo 1`: Conversão do *string* binário da base 2 para a base 10:

$$(< b_{21}b_{20}\cdots b_0 >)_2 = \left(\sum_{i=1}^{21} b_i \cdot 2^i\right)_{10} = x' \tag{2.16}$$

· `Passo 2`: Determinar um número real correspondente a x:

$$x = (\text{fronteira inferior}) + x' \frac{(\text{comprimento do domínio})}{2^{nbits} - 1} \tag{2.17}$$

$$x = -1,0 + x' \frac{3}{2^{22} - 1} \tag{2.18}$$

Por exemplo, um cromossomo (de 22 *bits*):
x=(1000101110110101000111) representa o número 0,637197, pois
x'=$(1000101110110101000111)_2$ = 2288967 e
Em síntese, os cromossomos:
(0000000000000000000000) e (1111111111111111111111) representam as **fronteiras do domínio** -1,0 e 2,0 respectivamente.

- *População inicial*

O procedimento de inicialização é muito simples: gera-se um população de cromossomos, onde cada cromossomo é um vetor binário de 22 *bits*. Todos 22 *bits* para cromossomo são gerados de forma aleatória e geralmente com distribuição uniforme.

- *Avaliação da função de avaliação*

A função de avaliação *aval* para vetores binários v é equivalente a função f:

$$aval(v) = f(x), \tag{2.19}$$

onde o cromossomo v representa o valor real x. A função de avaliação desempenha o papel do ambiente taxando as soluções potenciais para o problema em termos de seu *fitness* (adequação, aptidão).

Por exemplo, três cromossomos:
v_1=(10001011101101010001 11),
v_2=(00000011100000000010000), e
v_3=(11100000001111110001 01),
correspondem aos valores x_1=0,637197, x_2=-0,958973 e x_3=1,627888, respectivamente.

Consequentemente, a função de avaliação taxará os cromossomos conforme segue:
$aval(v_1)$=$f(x_1)$=1,586345
$aval(v_2)$=$f(x_2)$=0,078878
$aval(v_3)$=$f(x_3)$=2,250650
Nota-se que o cromossomo v_3 é o **melhor** dos três cromossomos apresentados, deste sua avaliação retorna o valor de $f(x)$ mais alto.

- *Operadores genéticos*

Durante a fase de alterações nas soluções obtidas pelo algoritmo genético são utilizados dois operadores genéticos clássicos: mutação e cruzamento.

`Operação de mutação`: A mutação altera um ou mais genes (posições em um cromossomo) com um probabilidade igual a taxa de mutação.
Assumindo-se que o 5° gene do cromossomo v_3=(1110 **0** 00000111111000101) foi selecionado para mutação. O 5° gene do cromossomo v_3 é **0** e é modificado para **1**.
Assim o cromossomo v_3 depois da mutação deve ser:
v'_3=(1110 **1** 00000111111000101).

Este cromossomo representa o valor x'_3=1,721638 e $f(x'_3)$=-0,082257. Isto significa que esta mutação em particular resulta em um **decréscimo** significativo do valor do cromossomo v_3.
De outra forma, se o 10° gene (em vez do 5° gene) fosse selecionado para mutação no cromossomo v_3 então
v_3=(111000000 **0** 111111000101) $\longrightarrow$ $v_3^{"}$=(111000000 **1** 111111000101).

O valor correspondente o valor $x_3^{"}$=1,630818 e $f(x_3^{"})$=2,343555. Isto significa que esta mutação em particular resulta no **acréscimo** do valor original $f(x_3)$=2,250650.

`Operação de cruzamento(com um ponto de corte)`: Para ilustrar o operador de cruzamento nos cromossomos v_2 e v_3, assume-se que o ponto de corte foi aleatoriamente selecionado para depois do 5° gene:
v_2=(00000 | **01110000000010000**),
v_3=(11100 | **00000111111000101**).

Número de gerações	função de avaliação
1	1,441942
10	2,250363
120	2,301290
150	2,850227

Tabela 2.1: Resultados para 150 gerações.

Os dois descendentes resultantes são:
v_2=(00000 | **00000111111000101**),
v_3=(11100 | **01110000000010000**).

Estes descendentes são avaliados como:
$f(v'_2)$=f(-0,998113) = 0,940865
$f(v'_3)$=f(1,666028) = 2,459245

Nota-se que o segundo cromossomo descendente (filho) tem uma melhor avaliação que ambos cromossomos antecessores (pais).

- *Parâmetros de controle do algoritmo genético*

Para este problema em particular foi utilizado os seguintes parâmetros:
tamanho da população = 50
probabilidade de cruzamento, p_c = 0,25
probabilidade de mutação, p_m = 0,01

- *Resultados experimentais*

Na tabela apresentada a seguir é apresentado o número da geração e a função de avaliação para o valor da função $f(x)$. Após 150 gerações, o melhor cromossomo foi v_{maximo}=(11110011010000100000101), que corresponde ao valor de $x_{\text{máximo}}$ = 1,850773. Conforme esperado, $x_{maximo}= 1,850773+\varepsilon$ e $f(x_{maximo})$ são ligeiramente maiores que 2,85. A tabela 2.1 apresenta resumidamente a evolução para a solução do problema por algoritmos genéticos.

Algoritmo genético com representação em ponto flutuante (ou real)

A seguir, apresenta-se uma descrição dos operadores de mutação e recombinação (ou seja, cruzamento para a representação em ponto flutuante ou real) para configuração de AGs com representação em ponto flutuante.

Operador de recombinação Os tipos de operadores de recombinação, para a representação em ponto flutuante, propostos por Michalewicz [110], são os seguintes:

- *recombinação simples*: é selecionado um par de indivíduos,

x=(x_1,$\cdots$, x_k, x_{k+1}, $\cdots$, x_q) e y=(y_1,$\cdots$, y_k, y_{k+1}, $\cdots$, y_q),
combinados a partir do k-ésimo parâmetro. Os indivíduos descendentes resultantes desta recombinação são:

$$x = (x_1, \cdots, x_k, y_{k+1}, \cdots, y_q) \text{ e } y = (y_1, \cdots, y_k, x_{k+1}, \cdots, x_q) \tag{2.20}$$

- *recombinação aritmética*: define a recombinação como uma combinação linear de dois vetores, x=(x_1,$\cdots$, x_k, x_{k+1}, $\cdots$, x_q) e y=(y_1,$\cdots$, y_k, y_{k+1}, $\cdots$, y_q). Após esta operação tem-se os descendentes descritos por:

$$x = \alpha_{ra} \cdot x + (1 + \alpha_{ra}) \cdot y \tag{2.21}$$

$$y = \alpha_{ra} \cdot y + (1 + \alpha_{ra}) \cdot x \tag{2.22}$$

  Neste operador pode-se ter um parâmetro $\alpha_{ra} \in [0;1]$, que é uma constante (*recombinação aritmética uniforme*) ou uma variável que depende do tamanho da população (*recombinação aritmética não-uniforme*).

- *recombinação heurística*: gera apenas um (ou mesmo nenhum) novo indivíduo, através da extrapolação linear de dois indivíduos. Considerando-se x=(x_1,$\cdots$, x_k, x_{k+1}, $\cdots$, x_q) e y=(y_1,$\cdots$, y_k, y_{k+1}, $\cdots$, y_q), após esta operação tem-se um indivíduo resultante factível regido pela expressão:

$$z = ra \cdot (y - x) + y \tag{2.23}$$

  onde ra é um número aleatório, tal que $ra \in [0;1]$. Caso a solução gerada não seja factível, deve-se gerar um novo número aleatório para a obtenção de um novo indivíduo.

Operador de mutação Os tipos de mutação, propostos por Michalewicz [110], são os seguintes:

- *mutação uniforme*: Seja x=(x_1,$\cdots$, x_k, $\cdots$, x_q), um indivíduo, a mutação uniforme gera um novo indivíduo a partir de outro. Após ser selecionado um indivíduo x, escolhe-se aleatoriamente um componente k, deste indivíduo para obtenção de um novo valor, tal que

$$x' = (x_1, \cdots, x'_k, \cdots, x_q) \tag{2.24}$$

  onde x'_k é um valor aleatório selecionado dentro dos limites do parâmetro k.

- *mutação não-uniforme*: o componente x_k é selecionado para mutação, resultando num vetor $x' = (x_1, \cdots, x'_k, \cdots, x_q)$, tal que:

$$x'_k = \left\{ \begin{array}{l} x_k + \Delta(t, \textit{limite maximo}(k)\text{-}x_k), \text{ se } z = 0, \\ x_k + \Delta(t, x_k - \textit{limite minimo}(k)), \text{ se } z = 1 \end{array} \right\} \tag{2.25}$$

onde z é um digito binário aleatório (0 ou 1), *limite máximo*(k) e *limite mínimo*(k) são os valores mínimo e máximo dos limites do parâmetro x'_k, respectivamente. A função $\Delta(t,y)$ retorna um valor no intervalo $[0,y]$ tal que a probabilidade de $\Delta(t,y)$ inicia em zero e é incrementada de acordo com o número de gerações t, tal que:

$$\Delta(t,y) = y \cdot ra \cdot \left[1 - \frac{t}{Nger}\right]^b \tag{2.26}$$

onde ra é um número gerado aleatoriamente no intervalo [0;1], N_{ger} é o número máximo de gerações e b é um parâmetro escolhido pelo usuário, que determina o grau de dependência com o número de gerações. Esta propriedade leva o operador a efetuar uma busca uniforme no espaço inicial, quando t é pequeno e, mais localmente nas gerações posteriores.

- *mutação de contorno*: nesta mutação seleciona-se aleatoriamente uma variável j com distribuição uniforme, em [0;1], e após atribui-se a variável j o limite máximo ou mínimo do parâmetro. O procedimento é resumido em:

$$x'_k = \left\{ \begin{array}{l} \textit{limite } \max \textit{imo}(k), \text{ se } k{=}j \text{ e } ra{<}0{,}5 \\ \textit{limite minimo}(k), \text{ se } k{=}j \text{ e } ra \geq 0{,}5 \\ x_k, \text{ para os outros valores} \end{array} \right\} \tag{2.27}$$

Algoritmo genético com representação inteira

Existem várias formas possíveis de representação inteira para a utilização de algoritmos genéticos em problemas de caixeiro viajante e roteamento. As mais conhecidas são seqüências de inteiros, onde o cromossomo é formado pela seqüência de nós no ciclo. Entre os operadores de cruzamento para esta representação tem-se:

- *partially-mapped crossover* (PMX);
- *order crossover* (OX);
- *cycle crossover* (CX); e
- *edge recombination crossover* (ERX).

  Os detalhes de implementação destes operadores e de como tratar soluções não factíveis podem ser encontrados em [62] e [6].

2.1.3 Configuração dos operadores, sintonia e controle dos parâmetros nos AEs

Éiben et al. [45] definem duas formas de configurar os valores dos parâmetros dos AEs (incluindo-se os AGs):

(i) sintonia de parâmetros: parâmetros configurados antes de executar o AE; e

(ii) controle de parâmetros: modificação dos valores dos parâmetros durante a execução do AE.

- As técnicas de controle de parâmetros levam em conta diversos questionamentos, tais como:
- *o que é modificado* ? (exemplo: representação, operadores, procedimento de seleção, mutação, ...);
- *de que forma a modificação é realizada* ? (exemplo: heurística determinística, heurística baseada em realimentação ou auto-adaptativa);
- *qual o escopo da modificação* ? (exemplo: nível populacional, nível individual,...);
- *qual a evidência de que a modificação foi realizada* ? (exemplo: monitoramento do desempenho dos operadores, diversidade da população, ...).

Apesar de intensas pesquisas na área, não existe uma regra única para estipular o tamanho da população, nem mesmo para a probabilidade de aplicação dos operadores genéticos. A escolha da probabilidade de cruzamento e probabilidade de mutação é um problema compexo de otimização não-linear. Entretanto, as suas configurações são criticamente dependentes da natureza da função objetivo [103].

A literatura menciona que as configurações adotadas em AGs (representação binária) utilizam usualmente a seguinte sintonia de parâmetros do AG: tamanho da população entre 30 e 200, probabilidade de cruzamento entre 0,5 e 1,0 e probabilidade de mutação entre 0,001 e 0,05 [36], [153].

Numa população pequena é relativamente fácil a ocorrência de uma convergência prematura, devido à ocorrência de cromossomos muito dominantes ou recessivos, na busca pelo espaço de solução. Muitas vezes, diversos experimentos ou mesmo procedimentos de tentativa e erro, são realizados para obtenção de valores adequados para os parâmetros de probabilidade e tamanho de população.

Em síntese, pode-se dizer que a sintonia de parâmetros, através de tentativa e erro, é uma prática usual em AEs. Um parâmetro de controle sintonizado pode ocasionar escolhas sub-ótimas, já que os parâmetros interagem freqüentemente de uma forma complexa. A sintonia simultânea de mais parâmetros, contudo, leva a uma quantidade excessiva de experimentos para obter uma relação adequada entre eles. Segundo Éiben et al. [45], as desvantagens da sintonia de parâmetros baseada em experimentação são resumidas pelos seguintes aspectos:

(i) os parâmetros de controle não são independentes, mas a obtenção sistemática de todas as combinações é praticamente impossível;

(ii) o procedimento de sintonia de parâmetros consome tempo; e

(iii) os valores dos parâmetros selecionados não são necessariamente ótimos.

A literatura é rica quanto a abordagens adaptativas de configuração de parâmetros em AEs [1], [163], [172], [161]. A seguir, algumas propostas de mecanismos adaptativos das probabilidades de cruzamento e mutação, e um operador baseado em gradiente são apresentadas. As abordagens apresentadas podem ser combinadas para abranger a vantagem de cada técnica em um AG eficiente.

2.1.4 Abordagem de ajuste automático das probabilidades de cruzamento e mutação

A configuração das probabilidades de cruzamento, p_c, e mutação, p_m, de forma adaptativa é realizada com dois objetivos: manter a diversidade da população e a capacidade de convergência apropriada dos AGs. A proposta de Srinivas & Patnaik [154] é a determinação das probabilidades dos operadores de cruzamento e mutação, que variam de acordo com os valores da função de aptidão dos indivíduos da população. Neste caso, as expressões para p_c e p_m são regidas pelas seguintes equações:

$$p_c(t) = \begin{array}{l} k_1 \frac{fmaximo(t) - f'(t)}{fmaximo(t) - \overline{f}(t)}, \text{para } f'(t) \geq \overline{f}(t) \\ k_3, \text{para } f'(t) < \overline{f}(t) \end{array} \tag{2.28}$$

$$p_m(t) = \begin{array}{l} k_2 \frac{fmaximo(t) - f'(t)}{fmaximo(t) - \overline{f}(t)}, \text{para } f'(t) \geq \overline{f}(t) \\ k_4, \text{para } f'(t) < \overline{f}(t) \end{array} \tag{2.29}$$

onde k_1, k_2, k_3 e k_4 são parâmetros de projeto, $\overline{f}(t)$ é o valor de aptidão média da população em relação ao valor de aptidão máximo, f_{maximo}, da população, $f'(t)$ é o maior dos valores de aptidão dos dois indivíduos a serem combinados ou o indivíduo a ser realizada mutação. Segundo Srinivas & Patnaik [154], os valores de $k_1 = k_3 = 1{,}0$ e $k_2 = k_4 = 0{,}5$ são escolhas adequadas.

Abordagem com probabilidade de mutação variável

A abordagem de Kim et al. [88] adota uma probabilidade de mutação variável, durante o ciclo evolutivo regida por uma função,

$$p_m(t) = \alpha_{p_m} \cdot e^{-\beta p_m t}, \text{ onde } \alpha_{p_m} \leq 0,1 \text{ e } \beta_{p_m} \leq 0,1, \tag{2.30}$$

onde α_{p_m} e β_{p_m} são parâmetros de projeto. Esta abordagem visa um aprimoramento da qualidade da solução, através um operador de mutação com taxa de aplicação que varia com a geração corrente.

Abordagem com operador de reprodução baseado em gradiente

Os pesquisadores Pham & Jin [126] propuseram um novo operador, denominado pelos autores, de operador de reprodução baseado em gradiente. Este operador visa acrescentar aos AGs uma natureza combinando características determinísticas

e estocásticas. O novo operador é empregado para superar as limitações usualmente associadas ao método de seleção por roleta.

Estas limitações incluem: perda do indivíduo mais apto da população, dominância excessiva de indivíduos muito aptos (elite dominante) e a habilidade limitada de explorar novos pontos no espaço de busca. Para a implementação deste operador um novo vetor de soluções é calculado pela equação:

$$x'(t) = x(t) + \psi i \frac{[f_{maximo}(t) - f_i(t)]}{f_{maximo}(t)} \left[x_{melhor}(t) - x_i(t)\right], \text{ onde } i = 1, ..., Nind, \tag{2.31}$$

onde ψ_i constitui-se de um coeficiente positivo, $x_{melhor}(t)$ é o vetor de parâmetros com maior aptidão, $f_{maximo}(t)$. Pham & Jin [126] recomendam ψ_i com valores entre 0 e 2 para o operador de reprodução. Este operador pode ser utilizado para aprimorar a qualidade da solução obtida, a cada geração, após aplicar-se os operadores de cruzamento e mutação.

2.2 Programação genética

A programação genética (PG) é uma forma de AE que distingue-se por seu particular conjunto de escolhas para a representação, projeto de operadores genéticos e avaliação da função de adequação. A PG constitui-se também de uma extensão dos AGs no tratamento da complexidade de estruturas computacionais, visando a obtenção de soluções potenciais em um ambiente que imite o processo de Darwin [91], [77].

A PG utiliza um desenvolvimento eficiente de estrutura de dados para a geração de expressões simbólicas e executa regressões simbólicas direcionando a determinação simultânea da estrutura e complexidade requerida pelo modelo durante o processo evolutivo. A resolução de um problema através de PG pode ser tratado como uma busca através de possíveis combinações de expressões simbólicas definidas pelo projetista. Cada expressão é codificada em uma estrutura em árvore, também denominada de programa computacional, apresentando um comprimento variável e subdividida em nós.

Os elementos da PG são tipicamente conjuntos fixos de símbolos selecionados no tratamento da solução de problemas em domínio de interesse, permitindo a otimização de uma estrutura em árvore de maneira mais apropriada que apenas por parâmetros numéricos.

Os elementos da PG são divididos em dois alfabetos, um funcional e outro terminal. O alfabeto funcional (não-terminal) é constituído, por exemplo, por um conjunto {*+, -, *, /, $\sqrt{}$, log, exp, ln, abs, e, ou*}, que inclui operações aritméticas, funções matemáticas, operações lógicas condicionais. O alfabeto terminal é um conjunto que inclui entradas (variáveis) apropriados para o domínio do problema, valores constantes e números.

O espaço de busca é um hiperespaço de todas as possíveis composições de funções que podem ser recursivamente compostas pelo alfabeto funcional e terminal. As ex-

pressões simbólicas (*S-expressions*) de uma linguagem de programação LISP (*List Processing*) são uma maneira especialmente conveniente de criar e manipular as composições de funções e terminais. Estes expressões simbólicas em LISP correspondem diretamente ao *parse tree* que é internamente gerado por muitos compiladores [91], [92]. A figura 2.6 apresenta um diagrama de blocos típico de uma árvore em PG.

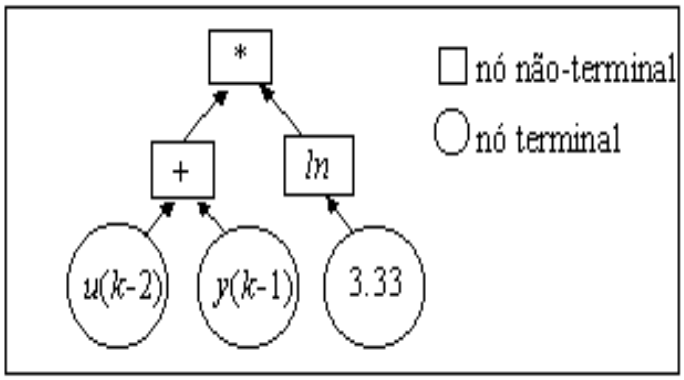

Figura 2.6: Diagrama de blocos típico de árvore em PG

A operação de recombinação é implementada por meio de sub-árvores de indivíduos aleatoriamente selecionados e pela execução de permutações de termos das árvores. As operações de mutação consistem, geralmente, da troca de genes com imposição de restrições pelo projetista [106], [66], [127].

O procedimento de otimização por PG pode ser dividido em um número seqüencial de passos [6]:

(i) criar aleatoriamente com distribuição uniforme uma população de árvores, provendo expressões simbólicas;

(ii) avaliar cada árvore atribuindo-lhe o valor da função de adequação;

(iii) utilizar uma técnica de reprodução pré-definida e assim copiar as árvores existentes para uma nova geração;

(iv) aplicar o operador de cruzamento em um conjunto de árvores ancestrais escolhidas aleatoriamente;

(v) aplicar o operador de mutação;

(vi) repetir os passos (ii) a (v) até que uma condição de parada seja satisfeita.

Durante a operação de cruzamento de duas árvores são escolhidas utilizando-se um dos métodos de seleção de forma similar aos métodos de seleção tratados pelos AGs. A operação de cruzamento deve preservar a sintaxe das expressões simbólicas. Em outras palavras, a aplicação dos operadores genéticos deve produzir uma equação que possa ser avaliada.

Por exemplo, uma sub-árvore é selecionada aleatoriamente de uma árvore ancestral (pai) no 1 é então trocada com uma sub-árvore aleatoriamente escolhida da árvore ancestral (pai) no 2.

As árvores criadas são então inseridas no *mating pool* que será utilizado para formar a próxima geração. Uma operação de recombinação válida é apresentada a seguir, onde duas árvores ancestrais, conforme apresentadas na figura 2.7, são tratadas.

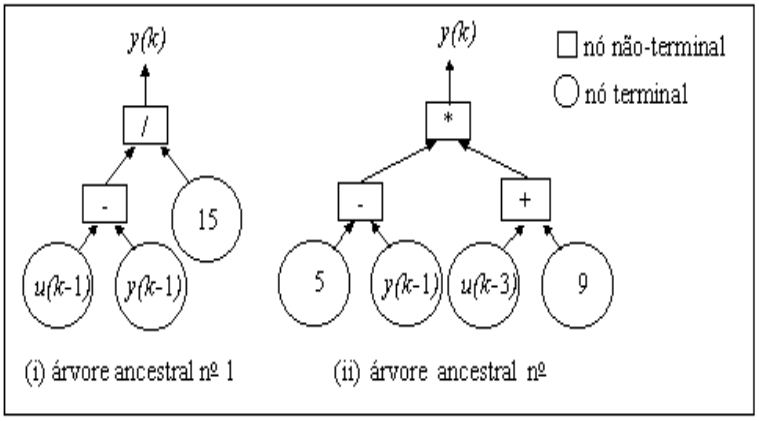

Figura 2.7: Representação das estruturas árvores ancestrais em PG

Após a operação de cruzamento em sub-árvores resulta na criação de duas árvores descendentes (filhos), conforme apresentadas na figura 2.8.

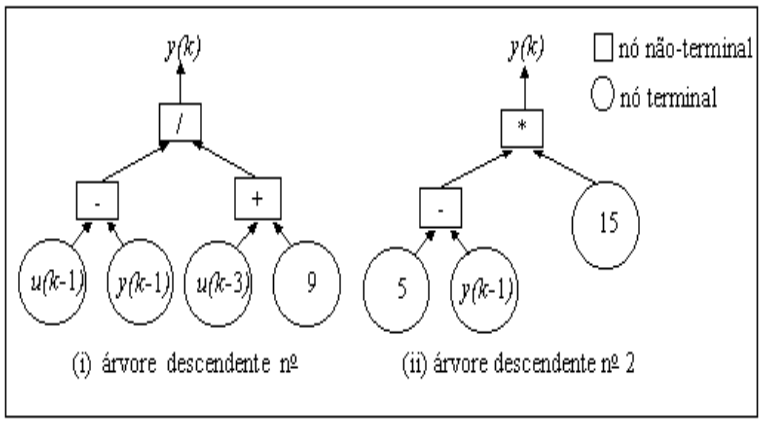

Figura 2.8: Representação das estruturas árvores descendentes resultantes da aplicação do operador de cruzamento em PG

O operador de mutação visa facilitar a exploração de diferentes partes do espaço de busca. A mutação cria um indivíduo modificado que é copiado para a próxima geração da população. A mutação consiste em síntese de uma mudança aleatória de uma função, uma entrada ou uma constante em uma expressão simbólica de uma população.

Um exemplo da aplicação do operador de mutação é a mudança do nó terminal de valor constante 15 da árvore apresentada na figura 2.8 (ii) para o valor 21, sendo esta operação mostrada na figura 2.9.

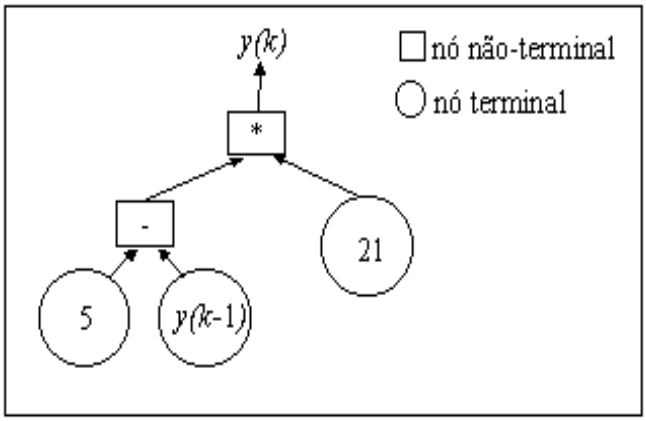

Figura 2.9: Exemplo da operação de mutação em PG.

A PG tem sido utilizada em diversas áreas do conhecimento, tais como: desenvolvimento de sistemas de classificação de imagens, projeto de filtros, equações apropriadas de sistemas de equações não-lineares de sistemas caóticos, indução de árvore de decisão, detecção de características, otimização e evolução de redes neurais artificiais [6], [106]. Mais informações referentes a base teórica e aplicações da PG podem ser encontrados em [6], [91], [92] e [152].

2.3 Sistemas classificadores

Outra metodologia, que é considerada um AE por alguns autores e por outros uma derivação dos AGs, são os sistemas classificadores. Um sistema classificador é munido de aprendizagem, que serve para aprender os strings de regras sintaticamente simples (chamadas de classificadores) objetivando a melhoria do desempenho em um ambiente arbitrário. Um sistema classificador é constituído de três componentes básicos:

- regras e sistema de mensagem;
- partilha de créditos;

- algoritmo genético.

As abordagens de sistemas classificadores tratadas na literatura dividem-se em abordagem Michigan e Pittsburgh, descritas em 1978 e 1980, respectivamente [18].

Na abordagem Michigan, um sistema simples constitui-se de um conjunto de regras (ou parâmetros). Os operadores genéticos são aplicados as regras existentes para descobrir-se novas regras. Nesta abordagem atribui-se um valor para cada regra, expressando a aptidão da regra por alcançar um *payoff*. As regras merecem altos valores por alcançarem o *payoff* diretamente no ambiente de tarefa, ou pela configuração das regras em estágio posterior.

Na abordagem Pittsburgh, muitos sistemas competem, cada um com seu conjunto de regras. A abordagem Michigan é mais prática para ambientes em tempo real, devido a sua carga computacional reduzida. Enquanto a abordagem Pittsburgh é apropriada para ambiente *off-line*, onde uma exploração mais lenta é aceitável.

2.4 Exercícios

1. O que é um algoritmo genético?
2. Quais as potencialidades dos algoritmos genéticos?
3. Quais são os tipos de representações dos algoritmos genéticos mais utilizadas?
4. Explique os operadores de seleção, cruzamento e mutação (estes usuais nos algoritmos genéticos).
5. Mencionar as principais características da programação genética.
6. O que é um sistema classificador?

Capítulo 3

Estratégias evolutivas e programação evolutiva

3.1 Introdução

As estratégias evolutivas [131], [143] foram inicialmente propostas com o intuito de solucionar problemas de otimização de parâmetros, tanto discretos quanto contínuos. Em virtude de originalmente somente utilizarem operadores de mutação, grandes contribuições em relação a análise e síntese destes operadores foram elaboradas, principalmente a análise de convergência e o desenvolvimento de mecanismos de auto-adaptação de parâmetros [138].

A programação evolutiva [50] foi originalmente proposta como uma metodologia para aprendizado de máquina através da evolução de máquinas de estado finito. A programação evolutiva originalmente também só emprega a operação de mutação.

As estratégias evolutivas e a programação evolutiva são representações da CE com muitas características similares. Entretanto estas abordagens apresentam diferenças que são comentadas nas seções apresentadas a seguir.

3.2 Fundamentos das estratégias evolutivas

As estratégias evolutivas (EEs), inicialmente, foram desenvolvidas para a resolução de problemas de otimização em engenharia. A primeira EE desenvolvida foi a EE-(1+1), proposta por I. Rechenberg e H. P. Schwefel, nos anos 60, no *Hermann Föttinger Institute for Hydrodynamics* (Universidade Técnica de Berlin, Alemanha), em experimentos com um processo túnel de vento. A EE-(1+1) original utiliza somente o operador de mutação, onde apenas uma solução ancestral produz um único descendente [6].

A EE-(1+1) foi progressivamente generalizada em variantes do número de ancestrais (pais), $\mu>1$, e número de descendentes (filhos), $\lambda>1$, por geração. As EEs

com múltiplos membros têm o embasamento biológico relacionado às características de poligenia e pleiotropia. Estas EEs são divididas de acordo com o mecanismo de seleção em:

- estratégia soma (*plus strategy*) ou EE-(μ+λ): Os μ ancestrais geram λ descendentes. Após os μ ancestrais e os λ descendentes competem pela sobrevivência;
- estratégia vírgula (*comma strategy*) ou EE-(μ, λ): Os λ descendentes competem para sobreviver e os μ ancestrais são completamente substituídos a cada geração [152].

A EE-(μ, λ) tem a tendência de manter uma maior diversidade de indivíduos na população, o que pode ser uma vantagem de forma a evitar-se mínimos locais. A desvantagem da EE-(μ, λ) é a possibilidade de uma solução "ótima" obtida, em uma dada geração, não "sobreviver" até o final do procedimento evolutivo.

A EE-(μ+λ), ao contrário, da EE-(μ, λ), dependendo de sua configuração pode ser mais susceptível a mínimos locais, ou seja, o domínio de uma elite de ancestrais (soluções similares) antes da obtenção de um valor adequado. Entretanto, uma solução ótima obtida, durante o procedimento evolutivo, não é perdida e "sobrevive" até que o final do procedimento evolutivo.

Os indivíduos (soluções), (x_i, σ_i), são diretamente representados por vetores de valores reais, $x_i \in \Re^n$ e vetores de desvio padrão, $\sigma_i \in \Re^n$. Na criação da população inicial os vetores x_i e σ_i são gerados aleatoriamente dentro de intervalos definidos pelo usuário.

O operador de mutação atua em cada *variável objeto*, x_i, pela adição de números aleatórios normalmente distribuídos (distribuição normal ou Gaussiana) com média zero e variância σ_i^2, regidos pela notação $N(0,\sigma_i^2)$. Um novo vetor solução pode ser criado através de uma regra de atualização com distribuição lognormal, tal que:

$$x_i'(t) = x_i(t) + \sigma_i N(0,1) \tag{3.1}$$

$$\sigma_i'(t) = \sigma_i(t) \cdot e^{\tau' \cdot N(0,1) + \tau \cdot N_i(0,1)}, \text{ onde } i = 1, ..., Nind, \tag{3.2}$$

onde $N_i(0,1)$ é uma distribuição variável Gaussiana aleatória para cada componente i. A mutação de σ_i é baseada em um fator de busca global $\tau' \cdot N(0,1)$ e um fator de busca local $\tau \cdot N_i(0,1)$. No caso do fator de busca global é gerado apenas um valor de $N(0,1)$, a ser utilizado por todos os σ_i dos indivíduos na geração atual. Entretanto, no caso do fator de busca local, $\tau \cdot N_i(0,1)$, é gerado um valor para cada indivíduo da população, com distribuição normal, média zero e variância σ_i^2. Estes fatores são regidos pelas seguintes equações [10]:

$$\tau = \frac{1}{\sqrt{2\sqrt{n}}} \tag{3.3}$$

$$\tau' = \frac{1}{\sqrt{2 \cdot n}} \tag{3.4}$$

onde n é o número de dimensões da função que está sendo otimizada. Os operadores de recombinação são similares aos implementados em representação por ponto flutuante nos AGs [110]. Entre as opções têm-se: a recombinação discreta, a intermediária (local e global) ou mesmo a não realização da operação de recombinação [10], [9].

A implementação de EEs apresenta diversas variantes, destacando-se as EEs contemporâneas e as EEs com mecanismos de auto-adaptação, com a realização de mutações correlacionadas [10], [13].

3.2.1 Mecanismo de auto-adaptação através de mutações correlacionadas

O desempenho das EEs, comparadas com outras técnicas de otimização, depende da configuração adequada dos seus parâmetros internos de controle. As EEs apresentam facilidades no ajuste de tais parâmetros através da utilização de auto-adaptação. Nos AGs, os parâmetros de controle são ajustados, usualmente, através de métodos heurísticos de tentativa e erro.

O princípio da auto-adaptação é facilitar o controle implícito dos parâmetros da EE, pela sua incorporação na representação do indivíduo com a evolução usual das variáveis objeto. O termo parâmetros da estratégia (ou parâmetros de controle) refere-se aos parâmetros que controlam o procedimento de busca evolutiva, tais como: taxa de mutação, variância da mutação e taxa de cruzamento [6].

Muitas das pesquisas dos princípios de auto-adaptação em AEs tratam de parâmetros relacionados com operador de mutação. A técnica de auto-adaptação é muito utilizada para as variâncias e as covariâncias de uma distribuição normal n-dimensional [51].

Segundo Angeline [5] é possível adaptar dinamicamente os aspectos de processamento de um AE, antecipando as regularidades do ambiente, aprimorando o procedimento de otimização e enfatizando a rapidez na busca dos parâmetros. Os AEs que apresentam mecanismos adaptativos (AEMAs) distinguem-se pela configuração dinâmica dos parâmetros selecionados ou operadores durante o ciclo evolutivo.

Os AEMAs têm uma vantagem sobre os AEs básicos, são mais reativos em antecipar as particularidades do problema e, em algumas formulações, podem dinamicamente adquirir informação sobre as regularidades no problema e explorá-las. Os AEMAs podem ser separados em três níveis, em que os parâmetros adaptativos atuam:

- *nível populacional*: os métodos adaptativos ajustam dinamicamente os parâmetros, que são globais à população inteira;
- *nível individual*: os métodos adaptativos modificam a maneira que um indivíduo da população é afetado pelos operadores de mutação;
- *nível de componente*: os métodos adaptativos alteram a forma pela qual os componentes de cada indivíduo são manipulados, independentemente dos outros indivíduos.

Os mecanismos de auto-adaptação, no nível de componente dos parâmetros da estratégia, providenciam uma das características principais do sucesso das EEs. As EEs utilizam princípios de busca no espaço de variáveis objeto e estratégia interna de controle dos parâmetros, simultaneamente [13]. A figura 3.1 apresenta o efeito da distribuição de tentativas com mutações Gaussianas descorrelacionadas (esquerda) e correlacionadas (direita) [8].

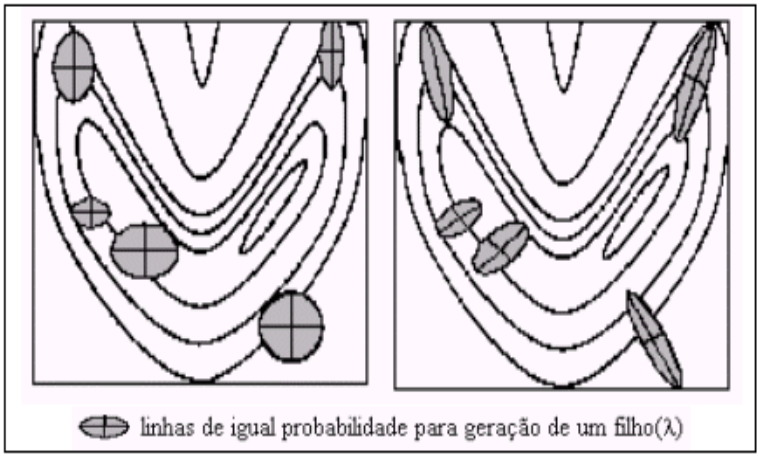

Figura 3.1: Efeito da operação de mutação descorrelacionada e correlacionada em EEs

As mutações Gaussianas geram contornos provavelmente elípticos em cada dimensão. Se as mutações são independentes através de cada dimensão, estas elipses estão alinhadas com os eixos coordenados. Se as mutações são correlacionadas, então as elipses podem realizar rotações arbitrárias. Desta forma as curvas subjacentes indicam contornos de erro igual na superfície de resposta. A figura 3.1 indica a utilidade potencial da incorporação de mutações correlacionadas devido ao contorno da superfície não estar alinhado com os eixos coordenados.

O poder de uma EE é baseado principalmente na habilidade de executar uma otimização de parâmetros de segundo nível. Este procedimento adapta o poder de mutação de tal forma que a EE apresente desempenho "próximo" do ótimo.

Existem diferentes possibilidades de construir tais estratégias. A regra mais simples é a de 1/5 de probabilidade de sucesso [131], onde a proporção de mutações bem sucedidas de todas mutações deve ser de 1/5. Esta forma de configuração da mutação opera satisfatoriamente em muitos problemas de otimização, mas depende da aplicabilidade de um modelo externo da topologia do espaço de parâmetros e isto somente habilita a tamanho de passo geral, não um tamanho de passo individual [120].

O operador de mutação em uma EE realiza um tipo de procedimento de subida de encosta, quando considera sua combinação com o operador de seleção. Com o

desvio padrão, σ, para cada variável objeto configura-se as direções de busca, que podem ser estabelecidas somente ao longo dos eixos do sistema de coordenadas, adotado. Em geral, a melhor direção de busca, ou seja, o gradiente, não é alinhado com estes eixos. Caso contrário, a trajetória da população, através do espaço de busca apresenta, um comportamento do tipo *ziguezague* ao longo do gradiente [7], [8].

Para evitar a redução da taxa de convergência, o operador de mutação pode ser estendido para manipular mutações correlacionadas. Schwefel [143] propõe uma extensão do operador de mutação deste tipo, que requer um vetor de estratégia adicional, aproximando a inversa da Hessiana de forma probabilística, simultaneamente, ao procedimento de otimização. Rudolph [136] mostrou que o procedimento de aproximação probabilística de [143] pode ser utilizado para a construção de um vetor aleatório correlacionado multinormal válido. Contudo, os resultados numéricos indicam que a convergência da aproximação ainda não é satisfatória. O motivo principal é que o procedimento de adaptação do tamanho de passo afeta o procedimento de adaptação do ângulo.

3.3 Programação evolutiva

A programação evolutiva (PE) foi desenvolvida inicialmente por L. J. Fogel, na década de 1960, teve enfoque na evolução de máquinas de estado finito. A transformação de seqüências de símbolos de entrada em seqüências de símbolos de saída, pelas máquinas de estado finito, visava a previsão de séries temporais baseada nas informações disponíveis *a priori*. A PE foi estendida, posteriormente, a problemas de otimização de parâmetros.

A PE, de forma análoga aos outros AEs apresentados, utiliza os conceitos de evolução, para gerar progressivamente soluções apropriadas para ambientes estáticos ou que mudam dinamicamente. A PE, de forma similar as EEs, difere dos AGs, pois são metodologias que simulam a evolução, enfatizando mais a ligação comportamental (relação fenotípica) entre as populações geradas (ancestrais e descendentes) que a ligação genética.

Entretanto, existem algumas diferenças entre as características da EE em relação a PE. A seleção na PE é tipicamente probabilística, usualmente, realizada através de torneio (*tournament*), enquanto na EE é determinística. A operação de mutação na PE, usualmente, utiliza a distribuição Gaussiana, enquanto a EE utiliza o operador de mutação com distribuição lognormal. Atualmente, existem alguns estudos comparativos da adoção de outras distribuições de probabilidade nos operadores de mutação de ambas, PE e EE [9], [7].

A operação de seleção na PE caracteriza-se por abstrair a evolução da população, onde diversas espécies competem para obter parte dos recursos, enquanto na EE a reprodução realiza-se no nível de comportamento individual [10], [71]. O procedimento de otimização via PE é implementado conforme os seguintes passos [52]:

(i) a criação da população inicial de parâmetros compreendendo *Nind* soluções.

Cada um dos indivíduos (x_i, σ_i) onde $x_i \in \Re^n$ consiste de vetores de soluções, e $\sigma_i \in \Re^n$ são desvios padrões, i=1,...,*Nind*, com as n dimensões correspondendo ao número de parâmetros a serem otimizados. Os componentes de cada x_i e σ_i são gerados aleatoriamente de acordo com uma distribuição uniforme em um intervalo *a priori* especificado;

(ii) cada solução x_i é classificada com relação a sua aptidão;

(iii) cada vetor de solução ancestral (x_i, σ_i) gera somente um vetor solução descendente (x'_i, σ'_i), de acordo com as seguintes equações:

$$x'_i(t) = x_i(t) + \sigma_i N(0,1) \tag{3.5}$$

$$\sigma'_i(t) = \sigma_i(t) + \alpha\sigma_i N(0,1), \text{ onde } i = 1, ..., Nind, \tag{3.6}$$

onde α é um fator de escala e o termo $N(0,1)$ é usado para gerar um número com distribuição Gaussiana com média 0 e desvio padrão 1;

(iv) cada vetor solução descendente x'_i é avaliado com relação à função de aptidão;

(v) comparações são conduzidas sobre todas as soluções x_i e x'_i. Para cada solução, k oponentes são selecionados, aleatoriamente, e após são escolhidos dos vetores solução ancestrais e descendentes, com igual probabilidade. A seleção realizada, neste caso, é através de competição por meio da seleção por torneio. Na seleção por torneio, em cada comparação, se a solução considerada oferece pelo menos um desempenho tão adequado quanto o oponente selecionado aleatoriamente, ela recebe uma "vitória";

(vi) os vetores soluções, x_i e x'_i, que apresentam mais "vitórias" são selecionados para serem ancestrais na próxima geração, sendo que os vetores σ'_i e σ_i a elas associados são também incluídos;

(vii) repetir os passos (ii) a (vi) até que uma condição de parada seja satisfeita.

3.3.1 Programação evolutiva com operador de mutação baseado em distribuição de Cauchy

Os descendentes na PE são gerados de seus ancestrais através de mutações aleatórias. Tipicamente, cada indivíduo, composto de uma variável objeto (vetor solução) acompanhado do seu respectivo desvio padrão, tem seu valor modificado por uma variável aleatória com distribuição Gaussiana, com média zero e valor de variância adaptável. A técnica de PE convencional possui operador de mutação lognormal e obedece ao método de Box-Muller [128], para a geração de valores aleatórios com uma distribuição normal.

Uma variante do operador convencional de mutação é com distribuição de Cauchy e apresenta uma função densidade de probabilidade centrada na origem, definida por [174]:

$$f_t(z) = \frac{1}{\pi}\frac{t}{t^2 + z^2}, \quad \text{para } -\infty < z < \infty \tag{3.7}$$

onde $t > 0$ é um parâmetro de escala. A função de distribuição correspondente é:

$$F_t(z) = \frac{1}{2} + \frac{1}{\pi} \arctan\left(\frac{z}{t}\right), \quad \text{para } -\infty < z < \infty \tag{3.8}$$

A forma de $f_t(z)$ parece-se com a função de densidade Gaussiana, mas nas proximidades o eixo $f_t(z)$ decresce mais vagarosamente. Como um resultado disto, a variância da distribuição de Cauchy é infinita. A figura 3.2 mostra a diferença entre as funções densidade de Cauchy e Gaussiana.

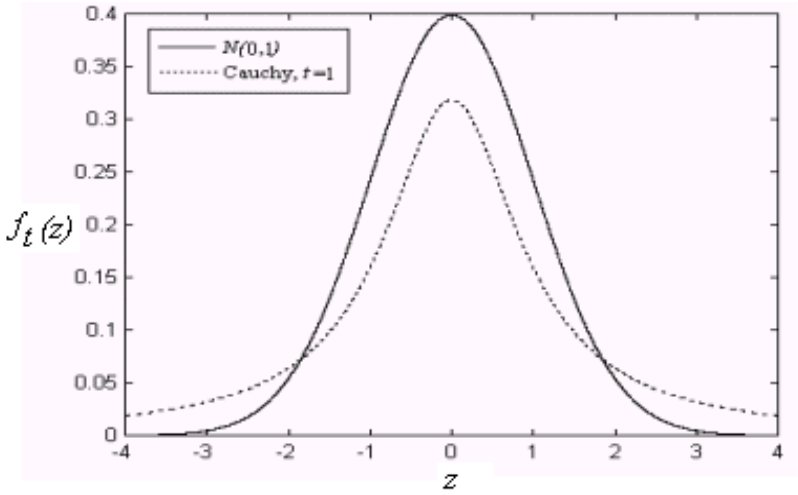

Figura 3.2: Diferença entre as funções densidade de Cauchy e Gaussiana

Alguns estudos têm indicado o benefício do incremento da variância em algoritmos de Monte Carlo. Um exemplo é a implementação de algoritmos rápidos de *simulated annealing* [79].

O operador de mutação, com distribuição de Cauchy [22], é útil para escapar de ótimos locais, enquanto o operador lognormal providencia a convergência local mais rápida em funções convexas. A estratégia de mutação, combinando estes dois operadores de mutação, pode explorar as propriedades desejadas de convergência da PE.

3.4 Exercícios

1. Explique o funcionamento do operador de mutação nas estratégias evolutivas.
2. Explique o funcionamento do operador de mutação na programação evolutiva.
3. Quais são as diferenças entre as estratégias evolutivas e a programação evolutiva?

Capítulo 4

Abordagens emergentes de algoritmos evolutivos

4.1 Introdução

Nos últimos anos, o campo da computação evolutiva tem visto o resurgimento de novas idéias, muitas delas sedimentadas em novas inspirações biológicas. Este capítulo apresenta em linhas gerais algumas abordagens emergentes de algoritmos evolutivos.

4.2 Algoritmos com busca local (evolução Lamarckiana, métodos híbridos de busca local ou algoritmos meméticos)

Os métodos de otimização têm duas formas de configuração: os métodos determinísticos e os métodos estocásticos. As técnicas determinísticas buscam um ponto de mínimo, baseadas na informação dada pelo negativo do gradiente da função objetivo. A eficiência destas técnicas depende de diversos fatores, tais como: o ponto (solução) inicial, a precisão da avaliação da direção descendente e o método utilizado para executar a busca em linha e critério de parada. A solução obtida é geralmente um ponto de mínimo local, que pode ser mínimo global se a função apresentar apenas uma moda. As duas desvantagens principais são a necessidade de avaliações do gradiente e falta da garantia do mínimo global.

Os métodos estocásticos, dos quais os AEs fazem parte, não necessitam do cálculo do gradiente e são aptos a encontrar a solução global. Contudo, o número de avaliações da função objetivo, necessárias para encontrar a solução, é normalmente maior que o número requerido pelos métodos determinísticos [164].

A solução de problemas com muitos ótimos locais defronta-se com o conflito fundamental entre precisão, integridade e esforço computacional. O compromisso entre *exploitation* (convergência) e *exploration* (diversidade da população) é uma

constante no AE e deve ser considerada na configuração de uma metodologia de otimização eficiente.

A configuração de abordagens compostas por técnicas determinísticas, estas hibridizadas, com técnicas estocásticas, é uma alternativa promissora se bem projetada. Para obter os benefícios da configuração híbrida, uma forma eficiente é executar, inicialmente, um AE para localizar a região de ótimo global e após aplicar-se outra metodologia de otimização para a busca local. A vantagem da utilização de um método de busca local está na melhoria da velocidade de convergência pela avaliação da função objetivo. O valor final obtido pelo método de busca local provavelmente será mais preciso que o obtido pelo AE.

Segundo Vasconcelos et al. [164], a maior dificuldade no uso de metodologias híbridas, também conhecidos por *algoritmos meméticos* [112], [108] (apesar das controvérias em adotar este nome para este tipo de método) ou evolução Lamarckiana [169], [85], [118] de natureza determinística e estocástica, é encontrar o momento de parar o procedimento de busca estocástico para iniciar o procedimento de busca determinística.

Os princípios da evolução Lamarckiana implementados computacionalmente baseiam-se que Jean Baptiste Lamarck (1744-1829) acreditava na herança das características adquiridas. O Lamarckianismo requer um mapeamento inverso do fenótipo e ambiente para o genótipo. No contexto da biologia, este mapeamento inverso não é plausível. Por outro lado, embora o Lamarckianismo não seja correto biologicamente, ele é útil para o desenvolvimento de AEs, pois organismos vivos não modificam seu DNA (ácido desoxirribonucléico) baseados em sua experiência, mas pode-se simular a evolução Lamarckiana em um computador.

Em síntese, a idéia principal é começar o método determinístico quando a região de mínimo global foi obtida. Contudo, não se sabe quando esta região é obtida. Assim, é impossível garantir que a solução é a global. Vasconcelos et al. [164] propõem os seguintes critérios para comutação entre AE e métodos de busca local que podem ser definidos pelo(a):

(i) *número de gerações*: No caso mais simples, o procedimento de otimização do AE é finalizado em um número especificado de gerações e o melhor resultado é transferido para o método determinístico. A principal desvantagem deste critério é que nem todas as funções objetivo apresentam o mesmo comportamento;

(ii) *diferença entre os melhores valores da função objetivo em um conjunto de gerações*: Este procedimento é menos sensível ao tipo de problema que a do item (ii);

(iii) *diferença entre os melhores valores da função objetivo em uma mesma geração*: Neste caso analisa-se a similaridade dos indivíduos que compõem a população.

Na literatura tem sido apresentadas vátrias abordagens de algoritmos de evolução Lamarckiana usando AEs combinados com método simplex de Nelder & Mead [149], quase-Newton [132], Hooke & Jeeves [27], *simulated annealing* [17], entre outros.

Em síntese, os AEs convencionais geralmente são eficientes para a busca global, mas são relativamente lentos em sintonia local, o que conduz à natural integração com métodos numéricos eficientes de descida de encosta, em buscas locais.

A seguir são apresentadas algumas abordagens de AEs híbridos. Estas podem ser consideradas formas de aprendizado Lamarckiano. As abordagens são: (i) AE híbrido com *simulated annealing,* (ii) AE híbrido com método simplex de Nelder & Mead , e (iii) AE híbrido com busca tabu.

4.2.1 Algoritmo evolutivo híbrido com *simulated annealing*

O algoritmo *simulated annealing* (SA) (ou têmpera simulada) é uma variação de algoritmos de subida de encosta, onde o objetivo é a minimização de uma função objetivo (nível de energia). Segundo Sun [156], o SA difere dos algoritmos de otimização convencionais devido ao fato que ele pode:

(i) processar funções objetivo que possuem graus arbitrários de não-linearidades, descontinuidades e estocasticidade;

(ii) tratar restrições impostas pela função objetivo;

(iii) ser implementado facilmente (poucas linhas de código) em relação a outros algoritmos de otimização não-linear;

(iv) garantir estatisticamente que encontra a solução ótima.

O SA baseia-se em uma analogia com a mecânica estatística de materiais com resfriamento, onde substâncias físicas tais como os metais são levados a altos níveis de temperatura e posteriormente são gradualmente resfriadas até alcançar um estado mínimo de energia. Sob outras condições, menos cuidadosas de resfriamento, o material se cristalizaria com uma energia "localmente mínima", o que freqüentemente se traduz em imperfeições estruturais. A esse processo cuidadoso de resfriamento denomina-se annealing.

A abordagem do SA é probabilística. O SA não requer informação de derivadas e não é afetado por descontinuidades e não-linearidades. O SA é um procedimento de descida de encosta, mas configurado de forma modificada. O SA realiza pequenos passos de busca em uma topografia local; se o passo resulta em uma melhora na solução, a nova solução é aceita, caso contrário, esta nova solução é aceita com probabilidade que inicialmente é configurada para 1. Contudo, com o progresso das iterações o SA é reduzida a probabilidade de aceitar uma nova solução que não apresenta aprimoramento em relação àquela obtida na iteração anterior.

A idéia do SA origina-se numa combinação das observações sobre a física dos materiais, com procedimento computacional tendo por finalidade a simulação do comportamento de uma coleção de átomos (variáveis do problema) em condições de temperatura fixa. Kirkpatrick et al. [90] foram os primeiros pesquisadores a propor e demonstrar a aplicação de técnicas de simulação da física dos materiais para problemas de otimização, especificamente em problemas de projeto de *hardware*.

O SA é muito aplicado em problemas de otimização combinatória, em sistemas híbridos com outros métodos numéricos de otimização e AEs. Neste último caso, o procedimento de *annealing* pode ser aplicado a um AE, a cada geração, após as operações de cruzamento e mutação, para sintonia fina (busca local) dos valores dos indivíduos de através de uma pequena perturbação no valor dos indivíduos.

4.2.2 Algoritmo evolutivo híbrido com método simplex

O método simplex de Nelder & Mead [115] − também conhecido como método do politopo ou método ameba − é uma técnica simples de busca direta que é utilizada em problemas de otimização. Uma busca direta significa que o método é guiado somente pelo cálculo do valor da função em vários pontos e não necessita da avaliação da primeira e segunda derivadas (parcial) da função a ser otimizada. O método simplex "mantém" diversos pontos diferentes. Esta é uma característica similar aos AEs que, no caso, mantêm uma população de pontos.

A maior diferença entre os AEs e o método simplex, é que o método simplex tem escolha de pontos de forma determinística, configurando um politopo de forma a "repelir" soluções inadequadas. Em contrapartida, os AEs têm desempenho vinculado à geração de novos pontos, através dos indivíduos com função de aptidão mais apta ao ambiente, durante o ciclo evolutivo.

O método simplex tem sua configuração básica na definição de um simplex. Um simplex é uma estrutura formada por (número de pontos + 1) pontos, não no mesmo plano, em um espaço multidimensional. A essência do algoritmo é a seguinte: a função é avaliada em cada ponto (vértice) do simplex. Em um problema de minimização, o vértice que têm maior valor de função objetivo é substituído por um novo ponto com valor da função objetivo menor. Existem quatro operações que são realizadas em um simplex: reflexão, expansão, redução e contração.

O simplex inicial se move, expande e contrai, de tal maneira que se adapta ao panorama da função e, finalmente, aproxima-se do ótimo. Para determinar a transformação apropriada, o método usa só a ordem relativa entre os desempenhos (valor da função a ser otimizada) do ponto considerado. Depois de cada transformação, o pior ponto é trocado pelo melhor dos já existentes, deste modo o algoritmo sempre força a convergência da seqüência de iterações.

Esta concepção de método simplex pode ser utilizada em diversas variantes possíveis de AEs híbridos [139], [175], [133].

4.2.3 Algoritmo evolutivo híbrido com busca tabu

A busca tabu (BT) é um método heurístico aplicado, com sucesso, a diversos problemas de otimização combinatória [61]. A BT foi proposta inicialmente por Glover [58], [59] para resolução de problemas de otimização. A BT é um procedimento adaptativo que pode guiar um algoritmo de busca local na exploração contínua do espaço de busca, sem ser encerrado prematuramente pela ausência de vizinhos que melhorem a solução corrente.

A BT explora a vizinhança de uma dada solução e seleciona a melhor solução encontrada nesta vizinhança mesmo que esta piore a solução corrente. Esta estratégia permite que a busca escape de um ótimo local e explore outra parcela do espaço de soluções. Se ocorrer retornos a um ótimo local previamente visitado (condição desejada, mas não necessária), a BT, através de seus mecanismos de controle, permite que a exploração do espaço de soluções prossiga evitando o efeito de "ciclagem". Se um movimento está presente na lista tabu, ele poderá ser aceito somente se minimizar o valor da função objetivo.

O processo no qual a BT transcende a otimalidade local se baseia em uma função de avaliação, que escolhe a cada iteração, o movimento com o maior valor de avaliação na vizinhança. Para tornar a busca mais flexível classifica-se um movimento de tabu para não tabu por algum critério de aspiração. Este critério libera um movimento do seu estado tabu antes que seu tempo tabu termine. Um movimento será aceito quando suas restrições tabus não forem violadas ou quando algum critério de aspiração retirar seu estado tabu. Os parâmetros básicos da BT são:

(i) a estrutura de memória,

(ii) o critério de aspiração, e

(iii) o critério de terminação.

Os passos genéricos da otimização baseada em BT são os seguintes:

(i) partir de uma solução inicial D.

(ii) iniciar os parâmetros lista tabu $\leftarrow 0$ e MelhorSolução $\leftarrow$ D.

(iii) enquanto um critério de terminação não for satisfeito faça: Avaliar a lista de candidatos a movimento (candidato = {D'$\in$ N(D), D' $\notin$ a lista tabu ou D' satisfaz a um critério de aspiração}: Selecionar a melhor solução admissível, D* $\in$ soluções candidatas.

(iv) atualizar a lista tabu; atualizar a solução corrente D $\leftarrow$ D*; se F(D) $<$ F(MelhorSolução) então MelhorSolução $\leftarrow$ D.

(v) retornar a melhor solução encontrada.

Segundo Müller [113], a BT tem-se mostrado eficiente na solução de problemas de otimização combinatorial, encontrando soluções, algumas vezes até melhores do que as encontradas por outras técnicas. A hibridização de AEs com busca tabu tem sido reportada na literatura em diversos trabalhos [105], [81], [87].

4.3 Efeito Baldwin

Na virada do século passado, não era claro se a teoria da evolução de Darwin ou evolução na ótica de Lamarck explicava melhor os mecanismos da evolução natural. Lamarck acreditava na transmissão direta das características adquiridas por

indivíduos durante sua vida. Enquanto Darwin propôs que a seleção natural aliada a diversidade poderia explicar a evolução.

O Lamarckismo foi uma teoria viável até que o trabalho de August Weismann [168], foi vastamente aceito. Weismann arguiu que uns organismos maiores têm dois tipos de células, as células germe que passam a informação genetica à prole e as céluals somáticas que não têm nenhum papel direto na reprodução. Weismann também comenta que não há nenhuma maneira da informação adquirida pelas células somáticas seja transmitida às células germe.

Neste contexto, James Mark Baldwin [11] propôs um "novo fator na evolução", onde as características adquiridas poderiam ser indiretamente adquiridas. Morgan [111] e Osborn [119] propuseram independentemente idéias similares. O "novo fator" foi a plasticidade fenotípica, isto é, a habilidade de um organismo de adaptar-se a seu meio ambiente durante sua vida. A abilidade de aprender é o exemplo o mais óbvio do plasticidade fenotípica, mas outros exemplos são a habilidade de bronzear-se com exposição ao sol ou aumento da força de um músculo com realização de exercícios. Baldwin [11] indicou que, entre outras coisas, o fator novo poderia explicar equilíbrios pontuais.

O efeito Baldwin opera através da realização de duas etapas. Primeiro, o plasticidade fenotípica permite que um indivíduo adapte-se a uma mutação "parcialmente" bem sucedida, que de outra maneira possa ser inútil ao indivíduo. Se esta mutação aumentar inclusive a aptidão, esta tenderá a se proliferar na população. Entretanto, o plasticidade fenotípica é tipicamente cara computacionalmente para um indivíduo. Por exemplo, aprender requer a energia e o tempo, e envolve às vezes erros perigosos.

Na segunda etapa do efeito Baldwin: é dado um tempo considerado suficiente e o mecanismo de evolução adotado tenta encontrar um mecanismo rígido que possa substituir o mecanismo plástico. Assim um comportamento que seja aprendido uma vez (na primeira etapa) pode eventualmente tornar-se instintivo (na segunda etapa). Este mecanismo possui características similares a evolução de Lamarckiana, mas não há nenhuma alteração direta do genótipo, baseada na experiência do fenótipo.

O efeito Baldwin chamou a atenção dos cientistas computacionais com o trabalho de Hinton & Nowlan [73]. O efeito Baldwin pode ser empregado em AEs quando um algoritmo genético é usado evoluir uma população dos indivíduos que empregam também um algoritmo de busca local. A busca local é o "análogo" computacional do plasticidade fenotípica na evolução biológica.

Em termos computacionais, na primeira etapa do efeito de Baldwin, a busca local "alisa" a superfície da aptidão (*fitness landscape*) para facilitar a busca evolutiva. Na segunda etapa, como genótipos melhores provavelmente estão presentes na população, existe uma pressão seletiva para a redução na busca local, principalmente devido aos custos computacionais intrínsecos para realização de várias avaliações do *fitness* pelo mecanismo de busca local.

Neste contexto, na literatura têm sido apresentados várias pesquisas empíricas sobre os benefícios da incorporação de aprendizado por efeito Baldwin em AEs, destacando-se [169], [100], [122], [117], [4], entre outros.

4.4 Algoritmos evolutivos interativos

A computação evolutiva interativa (*interactive evolutionary computing*) se relaciona à avaliação humana parcial (ou completa) das soluções geradas de um procedimento de otimização evolutivo. Esta forma de AE foi introduzida para problemas onde a avaliação quantitativa se não for impossível é difícil de conseguir.

Os exemplos da aplicação desta abordagem incluem as artes plásticas e problemas de animação gráfica, projeto automóveis, engenharia de alimentos e recuperação de base de dados. Tais aplicações confiam em uma avaliação da aptidão centrada em um projeto particular, em uma imagem, em um gosto humano, critérios subjetivos, entre outros, ao contrário de uma avaliação desenvolvida a partir de algum modelo matemático analítico.

A interação parcial pode também ser útil para o aprimoramento do procedimento de busca através da introdução de novas configurações de projeto obtidas a partir do conhecimento do projetista. Um apanhado de tais procedimentos evolutivos interativos podem ser encontrados na literatura em [160].

4.5 Agentes inteligentes e teoria de jogos

Os agentes inteligentes constituem uma classe de programas de computador que atuam a favor do usuário executando tarefas como buscar e/ou filtrar informação, negociar serviços, simplificar e automatizar tarefas complexas ou ainda colaborar com outros agentes para solucionar problemas mais complexos. É, na realidade, um poderosa abstração para visualizar e estruturar *softwares* de alta complexidade. Os desenvolvedores de *software* utilizam diariamente abstrações como procedimentos, funções, métodos e objetos, mas o desenvolvimento de agentes necessita, fundamentalmente, de novos paradigmas, ainda não muito familiares aos desenvolvedores. Desenvolver agentes inteligentes requer conhecimento especializado, envolve alta complexidade e longos períodos de tempo para a realização de testes, correção de erros e análise de resultados.

Em síntese, um agente inteligente possui tarefas de (ver figura 4.1): (i) percepção captada por sensores; (ii) atuadores que executam uma ação que é realimentada ao ambiente; e (iii) ações que são realizadas baseadas em algum mecanismo de inferência ou raciocínio, de acordo com as percepções captadas.

Neste contexto, diversas pesquisas têm sido desenvolvidas recentemente usando agentes evolutivos inteligentes [34], ambientes de projeto virtual baseado em agentes evolutivos [159], algoritmos evolutivos combinados a teoria de jogos [23], [144].

Uma vertente de pesquisa interessante vinculada a teoria de jogos é a utilização de mecanismos de coevolução, ou seja, populações que evoluem simultaneamente, tanto para cooperarem entre si como para competirem. Essas abordagens são promissoras para aplicações de alguns tipos de jogos, uma vez que pode-se ter agentes em constante competição com outros agentes com o objetivo de superá-lo e, ao mesmo tempo, cooperar entre si com a mesma finalidade (ou seja, obter um melhor desempenho na resolução de problemas).

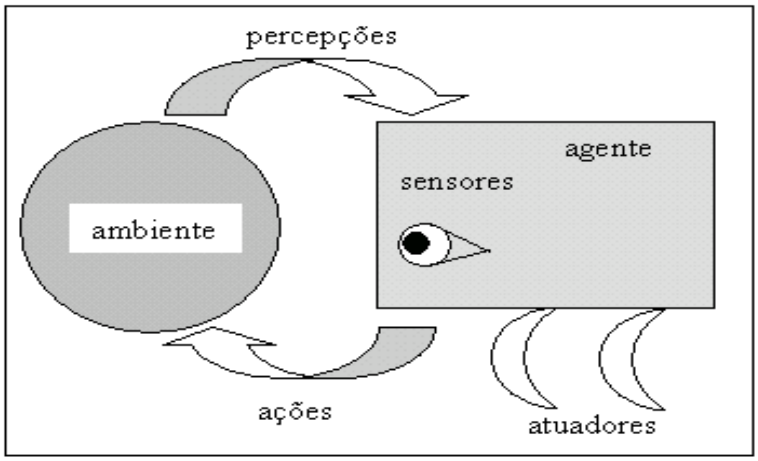

Figura 4.1: Concepção de um agente inteligente

Entre os exemplos de aplicações tem-se modelos de cooperação para aprimoramento de operadores dos AGs [2] métodos de coevolução entre AEs [170], análise de cooperação em problemas do tipo *prisoner's dilemma* [145].

4.6 Evolução diferencial

A evolução diferencial é um AE proposto por Rainer Storn e Kenneth Price [158] para problemas de otimização. Na evolução diferencial, o valor de cada variável é representada por um valor real (ponto flutuante) e o seu procedimento de otimização é regido pelas seguintes etapas:

(i) gerar uma população inicial aleatória, com distribuição uniforme, de soluções factíveis à resolução do problema em questão, onde é garantido por regras de "reparo" que garantem que os valores atribuídos as variáveis estão dentro das fronteiras delimitadas pelo projetista;

(ii) um indivíduo é selecionado, de forma aleatória, para ser substituído e três diferentes indivíduos são selecionados com genitores (pais);

(iii) um destes três indivíduos é selecionado como genitor principal;

(iv) com alguma probabilidade, cada variável do genitor principal é modificada. Entretanto, no mínimo uma variável deve ter seu valor alterado;

(v) a modificação é realizada pela adição do valor atual da variável de uma taxa, F, da diferença entre dois valores desta variável nos dois outros genitores. Em outras palavras, o vetor denominado genitor principal é modificado baseado no vetor de variáveis de dois outros genitores. Este procedimento representa o operador de cruzamento na evolução diferencial;

(vi) se o vetor resultante apresenta uma função de aptidão melhor que o escolhido à substituição, ele o substitui; caso contrário, o vetor escolhido para ser substituído é mantido na população.

Em outras palavras, adotando-se um formalismo matemático, na evolução diferencial uma solução, l, na geração w é um vetor multidimensional $x^l_{G=w} = (x^l_1, \cdots, x^{lT}_N)$. Uma população, $P_{G=k}$, na geração $G = k$ é um vetor de M soluções , onde $M > 4$. A população inicial $P_{G=0} = (x^l_{i,G=0}, \cdots, x^M_{i,G=0})$, é gerada inicialmente, com distribuição uniforme, adotando-se

$$x^l_{i,G=0} = \textit{limite inferior}(x_i) + \textit{rand}_i[0,1] * [\textit{limite superior}(x_i) - \textit{limite inferior}(x_i)] \tag{4.1}$$

onde *limite inferior*(x_i) e *limite superior*(x_i) são os limites inferior e superior de valores admissíveis para a variável x_i, respectivamente; M é o tamanho da população; N é a dimensão da solução e $rand_i[0,1]$ gera um número aleatório, com distribuição uniforme, no intervalo entre 0 e 1. A seleção é realizada para selecionar quatro diferentes índices de soluções r_1, r_2, r_3 e $j \in [1, M]$. Os valores de cada variável, na solução descendente (filha), são modificados com uma mesma probabilidade de cruzamento, p_m, para $\forall i \leq N$,

$$x^l_{i,G=k} = \left\{ \begin{array}{c} x^{r3}_{i,G=k-1} + F \cdot \left(x^{r1}_{i,G=k-1} - x^{r2}_{i,G=k-1} \right), \text{ se } (rand[0,1\} < p_c \wedge i = i_{rand}) \\ x^j_{i,G=k-1}, \text{ nos outros casos} \end{array} \right\} \tag{4.2}$$

onde $F \in (0, 1)$ é uma taxa de "perturbação" a ser adicionada a uma solução escolhida aleatoriamente denominada genitor (ancestral) principal. A nova solução substitui a solução anterior (antiga) se ela é melhor que ela e pelo menos uma das variáveis é modificada, esta solução é representada na evolução diferencial pela seleção aleatória de uma variável, $i_{rand} \in (1, N)$. Depois da operação de cruzamento, se uma ou mais variáveis na nova solução estão fora das fronteiras (limites) uma regra de "reparo" é aplicada, ou seja,

$$x^l_{i,G=k} = \left\{ \begin{array}{l} \left[x^j_{i,G} + \textit{limite inferior}(x_i) \right] /2, \text{ se } x^j_{i,G+1} < \textit{limite inferior}(x_i) \\ \\ \textit{limite superior}(x_i) + \\ \left[x^j_{i,G} - \textit{limite superior}(x_i) \right] /2, \text{ se } x^j_{i,G+1} > \textit{limite superior}(x_i) \\ \\ x^j_{i,G+1}, \text{ nos outros casos} \end{array} \right\} \tag{4.3}$$

4.7 Sistema imunológico artificial

O sistema imunológico gera interesse pela pesquisa dado o seu poder de processar informação. Ele executa processos complexos de forma paralela, distribuída, e possui a propriedade de interagir localmente. O sistema imunológico age como um "segundo cérebro" pois pode armazenar as experiências passadas, fortalecer as interações de suas células componentes e gerar uma nova resposta para novos padrões de antígenos [19].

A principal caraterística do sistema imunológico é reconhecer todas as células (ou moléculas) no corpo e categorizá-las como sendo próprias ou externas. As células externas são, por sua vez, identificadas de forma a estimular um ou mais sistemas defensivos. O sistema imunológico aprende, através da evolução, a distinguir entre antígenos (bactéria, vírus, etc.) e anticorpos. Na figura 4.2 é apresentada a estrutura multicamadas do sistema imunológico [150].

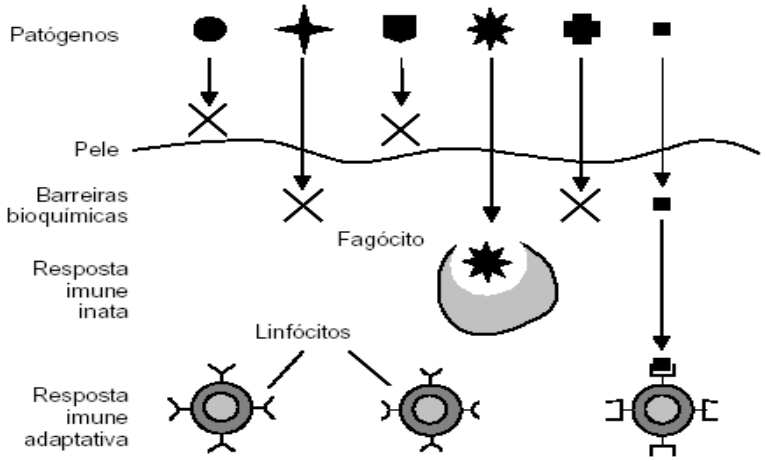

Figura 4.2: Estrutura multicamadas do sistema imunológico

O sistema imunológico artificial é constituído de dois operadores: hipermutação e expansão clonal. O conceito de hipermutação consiste no fato de que o mesmo anticorpo pode gerar n clones, e que cada gene do clone pode ser mutado. Na próxima geração, cada um destes n clones poderá gerar outros n clones, e assim sucessivamente. Isto significa que cada gene de um anticorpo pode sofrer mutação diversas vezes entre a primeira iteração e a última. Um anticorpo, gerado a partir de clonagem, só é adicionado à população de anticorpos se, e somente se, o valor da sua respectiva função de avaliação for maior ou igual a um limiar (*threshold*) de expansão clonal [19].

O mecanismo de expansão clonal pode prover a regulação do processo de hipermutação, a qual é dependente da afinidade do anticorpo. Um anticorpo com baixo valor de avaliação deve ser eliminado. Em anticorpos com alto valor de avaliação, o

processo de seleção clonal é iniciado. Alguns detalhes do esquema do princípio da seleção clonal são apresentados na figura 4.3 [150].

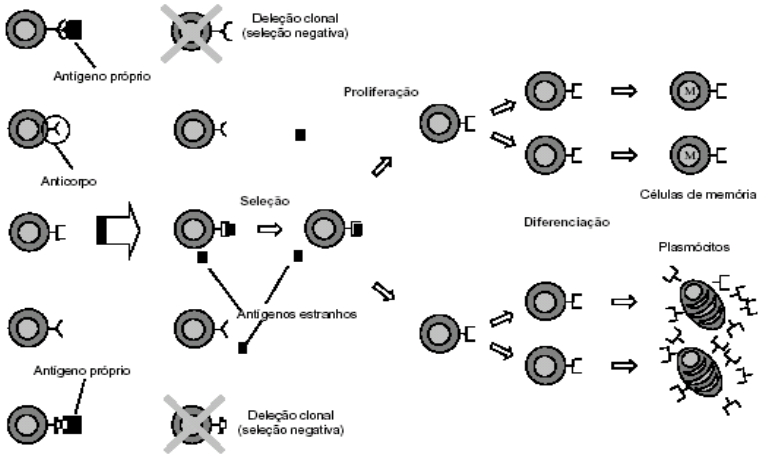

Figura 4.3: Esquema do princípio da seleção clonal

O valor de *match* é importante para o procedimento de expansão clonal. No "algoritmo imunológico", citado em [19], o valor de *match* é obtido através da função de avaliação. Um anticorpo só é clonado se este anticorpo tem seu valor de avaliação maior que um limiar pré-determinado. A cada iteração do algoritmo de seleção clonal, se um anticorpo obtém o limiar da expansão clonal, o sistema gera clones deste anticorpo. Estes clones são submetidos ao processo de hipermutação. Neste caso, tanto o limiar quanto o número de clones são estabelecidos previamente.

Existem várias experiências de desenvolvimento de metodologias, baseadas no sistema imune, para tratar de problemas de interesse comercial ou industrial. Algumas organizações financeiras estão utilizando técnicas de imunização para prevenir contra fraudes em processos de hipotecas e empréstimos [78].

Da mesma forma que o sistema imune pode detectar uma infecção viral no organismo, é possível desenvolver uma aplicação capaz de detectar a presença de vírus em computadores. Os vírus, em geral, infectam programas, setores de *boot* e arquivos em geral. A partir deste ponto de vista o problema de proteção é semelhante ao apresentado nos organismos, detecção *self-nonself*, onde *self* são dados não corrompidos e *nonself* é interpretado como alguma alteração do *self* [151].

4.8 Colônias de partículas (*particle swarm optimization*)

A otimização por colônia de partículas (OCP) é uma abordagem de AE baseada em uma população de indivíduos e motivada pela simulação de comportamento social

em vez da sobrevivência do indivíduo mais apto e da evolução da natureza como em outros AEs, tais como algoritmos genéticos, programação evolutiva, estratégias evolutivas e programação genética. A OCP é uma metodologia baseada em população de soluções.(ver uma representação na figura 4.4).

Figura 4.4: Representação de uma colônia de partículas

Na OCP, cada solução candidata (denominada partícula) possui associada uma velocidade. A velocidade é ajustada através de uma equação de atualização que considera a experiência da partícula correspondente e a experiência das outras partículas da população.

A OCP é uma metodologia proposta originalmente por Kennedy & Eberhart [43], [84] baseada em uma população de soluções. De forma similar a outros AEs, a OCP é iniciada com uma população de soluções gerada aleatoriamente.

De forma diferente dos AEs, cada solução potencial (indivíduo) na OCP é também atribuída uma velocidade aleatória. As soluções potenciais denominadas partículas são então "movimentadas" pelo espaço de busca do problema.

Cada partícula conserva o conhecimento do seu melhor valor da função de aptidão denotada por *pbest* (versão local). Um outro melhor valor é "seguido" pela versão global, *gbest*, do otimizador por colônia de partícula e sua localização obtida de alguma partícula que compõe a população.

O conceito da OCP consiste de, a cada passo iterativo, mudar a velocidade (acelerando) de cada partícula em direção as localizações do *pbest* e do *gbest*. A aceleração desta busca é ponderada através de um termo gerado de forma aleatória vinculados estes de forma separada as localizações do *pbest* e do *gbest*. O procedimento para implementação da OCP é regido pelas seguintes etapas [16]:

(i) iniciar uma população (matriz) de partículas, com posições e velocidades em um espaço de problema n dimensional, de forma aleatória com distribuição uniforme.

(ii) para cada particular, avaliar a função de aptidão.

(iii) comparar a avaliação da função de aptidão da partícula com o *pbest* da partícula. Se o valor corrente é melhor que *pbest*, então o valor de *pbest* passa a ser igual ao valor da função de aptidão da partícula, e a localização do *pbest* passa a ser igual a localização atual no espaço n dimensional.

(iv) comparar a avaliação da função de aptidão com o prévio melhor valor de aptidão da população. Se o valor atual é melhor que o *gbest*, atualizar o valor de *gbest* para o índice e valor da partícula atual.

(v) modificar a velocidade e a posição da partícula de acordo com as equações 4.4 e 4.5 , respectivamente [146], [147]:

$$v_i'(t) = w \cdot v_i(t) + c_1 \cdot ud() \cdot [p_i(t) - x_i(t)] + c_2 \cdot Ud() \cdot [p_g(t) - v_i(t)] \quad (4.4)$$

$$x_i'(t) = x_i(t) + v_i'(t) \quad (4.5)$$

(vi) ir para a etapa (ii) até que um critério de parada seja encontrado, usualmente uma função de aptidão suficientemente boa ou um número máximo de iterações (gerações).

As notações usadas são: x_i=$[x_{i1}, x_{i2}, \cdots, x_{in}]^T$ armazena a posição da i-ésima partícula, v_i=$[v_{i1}, v_{i2}, \cdots, v_{in}]^T$ armazena a velocidade da i-ésima partícula. O valor de p_i=$[p_{i1}, p_{i2}, \cdots, p_{in}]^T$ representa a posição do melhor valor de aptidão da i-ésima partícula. O índice g representa o índice da melhor particular entre todas as partículas do grupo. A variável w é a ponderação de inércia; c_1 e c_2 são constantes positivas; $ud()$ e $Ud()$ são duas funções para geração de números aleatórios com distribuição uniforme no intervalo [0,1], respectivamente.

As velocidades das partículas em cada dimensão são limitadas a um valor máximo de velocidade, V_{max}. O V_{max} é um parâmetro importante, pois determina a resolução que a região próxima a soluções atuais são procuradas. Se V_{max} é muito alto, a OCP facilita a busca global, enquanto um valor V_{max} pequeno enfatiza as buscas locais.

A primeira parte na equação 4.4 é um termo de momento da partícula. A ponderação de inércia w representa o grau de momento da partícula. A segunda parte consiste da parte "cognitiva", que representa o "conhecimento" da partícula independentemente. A terceira parte é a parte "social", que representa a colaboração entre as partículas.

As constantes c_1 e c_2 representam a ponderação das partes de "cognição" e "social" que influenciam cada particular em direção as *pbest* e *gbest*. Estes parâmetros são usualmente ajustados por heurísticas de tentativa e erro. O tamanho da população também é selecionado dependendo do problema.

4.9 Otimização por colônia de formigas (*ant colony*)

Cada criatura viva tem a missão de preservar sua espécie sob efeito dinâmico de uma variedade de mudanças ambientais. É conhecido que muitos grupos de criaturas estão aptas a resolução de problemas através de atividades exercidas de forma cooperativa.

Neste contexto, um sistema distribuído pode melhorar seu desempenho geral através da interação entre diversos agentes autonômos. Um exemplo deste tipo de resolução de problema por cooperação é a utilização de um meta-heurística denominada colônia de formigas artificial [96].

4.9.1 Inspiração biológica

As formigas reais são capazes de encontrar o caminho mais curto para uma fonte de alimento do formigueiro ou ninho sem a utilização de dados visuais. Enquanto se movimentam, as formigas depositam no solo uma substância denominada de feromônio (designação genérica de substâncias secretadas pelas formigas que servem de meio de comunicação entre elas), e seguem seu deslocamento baseadas em feromônios previamente depositados por outras formigas. Estas trilhas de feromônios pode ser observadas por outras formigas e motivar elas em seguir determinado caminho, isto é, um movimento aleatório das formigas seguirá com maior pribablidade uma trilha de feromônio. Esta é uma maneira de como as trilhas são reforçadas e mais e mais formigas seguirão aquela trilha.

Uma formiga trabalha da seguinte forma: Primeiro, quando as formigas chegam a um ponto de decisão em que elas tem que decidir mover-se à direita ou à esquerda, as formigas selecionam aleatoriamente o próximo caminho e depositam feromônio no solo, sem ter a noção de qual é a melhor escolha. Depois de um pequeno período de tempo a diferença entre a quantidade de feromônio entre dois caminhos é suficientemente grande que influencia a decisão de novas formigas estão no impasse da tomada de decisão por qual caminho seguir. Neste caso, as novas formigas escolhem o caminho com maior quantidade de feromônio. Consequentemente, as formigas podem cheirar o feromônio e escolher, com dada probabilidade, os caminhos marcados com concentrações mais acentuadas de feromônios. Em síntese, este princípo da natureza pode ser útil na configuração de uma colônia de formigas artificiais para resolução de problemas do tipo caixeiro viajante, escalonamento, problema de atribuição quadrática (*quadratic assignment problem*) , entre outros.

4.9.2 Características e fundamentos matemáticos da colônia de formigas artificial

O algoritmo de colônia de formigas artificial ou *ant colony systems* (ACS) é inspirado pelo comportamento de colônias de formigas reais, em particular, pelo seu comportamento de procura de alimento. Uma das idéias centrais do ACS proposto originalmente por Dorigo et al. [42] é a comunicação indireta entre uma colônia de agentes ou formigas (*ants*) baseada em trilhas (caminhos ou trajetos) de feromônios.

As trilhas de feromônios são um tipo de informação numérica distribuída que é modificada pelas formigas para refletir sua experiência quando da resolução de um problema em particular [155], [104].

O ACS é uma meta-heurística baseada em uma população de agentes (formigas) que pode ser utilizada para resolução de problemas de otimização combinatória. O ACS apresenta as seguintes características:

(i) é um algoritmo não-determinístico baseado em mecanismos presentes na natureza desde que ele é baseado no comportamento de formigas para a determinação de caminhos através de suas colônias para procura eficiente de fontes de comida;

(ii) é um algoritmo paralelo e adaptativo, pois uma população de agentes movém-se simultaneamente, de forma independente e sem um supervisor (não há controle central);

(iii) é um algoritmo cooperativo, pois cada agente (formiga) escolhe um caminho com base na informação (trilhas de feromônios) depositadas por outros agentes que tenham selecionado previamente o mesmo caminho. Este comportamento cooperativo tem ingredientes de autocátalise (catálise provocada por um substância — feromônio — que se forma no próprio sistema reacional), isto é, o ACS providencia uma realimentação positiva, desde que a probabilidade de um agente escolher o caminho aumenta com o número de agentes que escolheu previamente aquele caminho.

Nas figuras 4.5 a 4.14 é apresentada uma evolução do comportamento cooperativo das formigas de sua movimentação do ninho ou formigueiro (*nest*) até o alimento (*food*) baseada na atualização do feromônio [93].

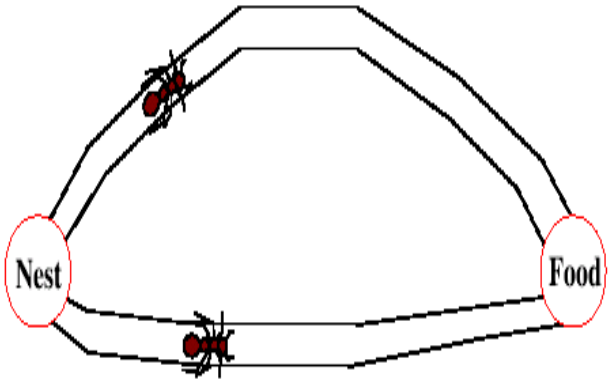

Figura 4.5: Movimentação inicial das formigas em direção ao alimento

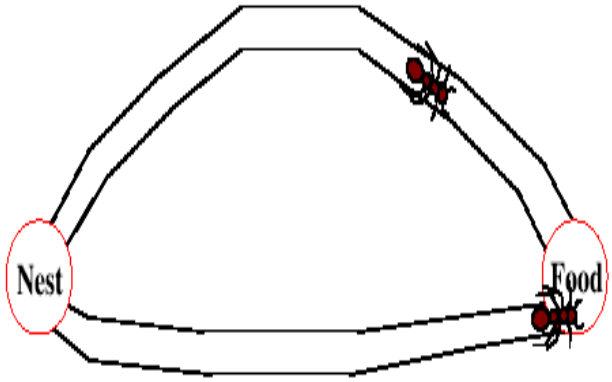

Figura 4.6: Uma formiga acha o alimento

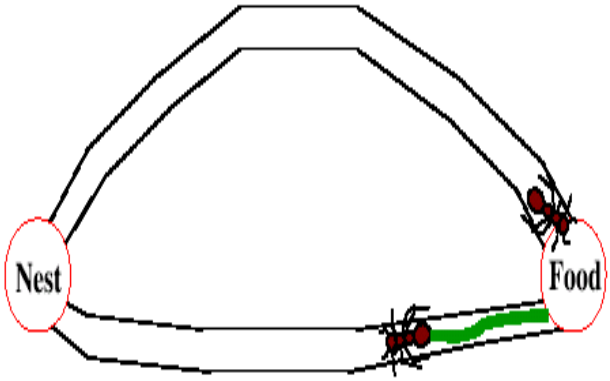

Figura 4.7: Uma formiga está retornando ao formigueiro, enquanto a outra está se movimentando em direção ao alimento

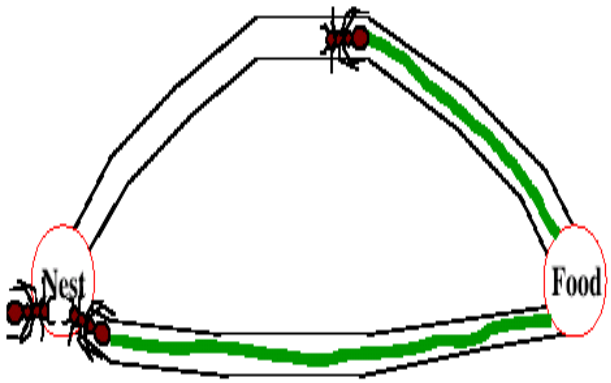

Figura 4.8: Uma formiga chegou ao formigueiro e outra saiu em direção ao alimento

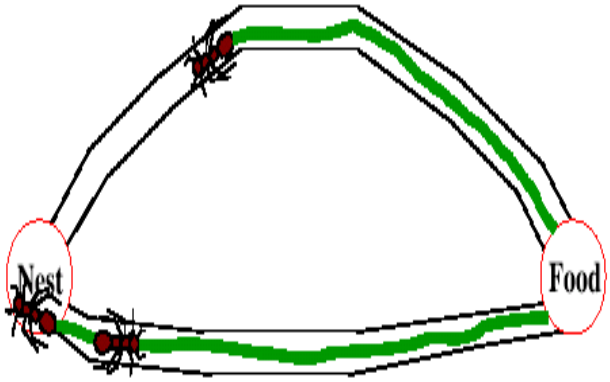

Figura 4.9: Uma formiga chegou ao formigueiro e outra saiu em direção ao alimento

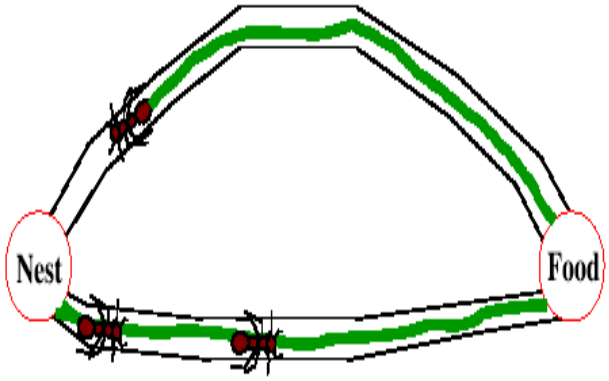

Figura 4.10: Outras formigas começam a seguir o feromônio depositado pela primeira formiga a retornar ao formigueiro

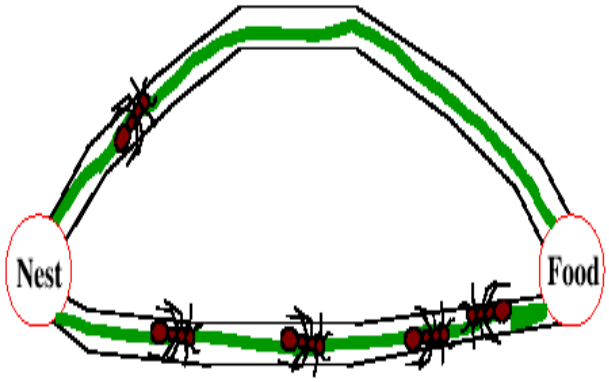

Figura 4.11: Várias formigas começam a se movimentar em direção ao formigueiro

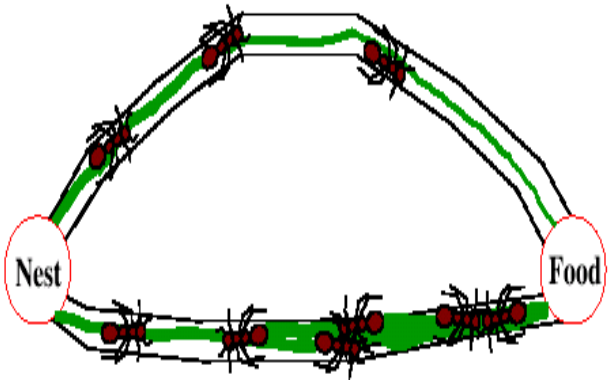

Figura 4.12: Nota-se que um dos dois caminhos começa a ter um depósito bem mais acentuado de feromônio que o outro

Figura 4.13: A maioria das formigas começam a seguir o melhor caminho (com mais feromônio) em direção ao alimento

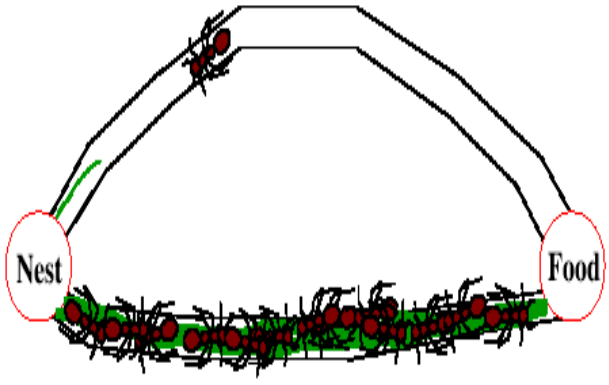

Figura 4.14: A colônia de formigas tende a seguir o caminho com mais feromônio depositado

4.9.3 Pseudocódigo e formulação matemática do ACS

Uma população tem m formigas, onde k é uma formiga genérica (k=1,...,m). A probabilidade de que a k-ésima formiga atribua a facilidade j para localização i é dada por:

$$p_{ij}^{k}(t) = \left\{ \begin{array}{l} \frac{[\tau_{ij}(t)]^{\alpha} \cdot \left[\eta_{ij}(t)\right]^{\beta}}{\sum_{k \in k_{permitidos}} [\tau_{ik}(t)]^{\alpha} \cdot [\eta_{ik}(t)]^{\beta}} \\ 0, \text{ outros casos} \end{array} \right\} \tag{4.6}$$

Na equação 4.6 α e β são parâmetros com ajuste usualmente heurístico. A variável

α é a ponderação do feromônio ($0 \leq \alpha \leq 1$) e β é a ponderação da informação heurística ($0 \leq \beta \leq 1$), $\eta_{ij} = 1/d_{ij}$ é a visibilidade entre a variável j até a variável i e vice-versa; d_{ij} é a distância Euclidiana entre i e j; τ_{ij} é a intensidade da trilha no caminho *(i,j)* no tempo t (quando t=0 a intensidade da trilha é gerada aleatoriamente com distribuição uniforme).

Ao longo da trilha de i até j a formiga deposita na trilha uma substância (feromônio) definida por:

$$\Delta\tau_{ij}^{k} = \left\{ \begin{array}{l} \frac{Q}{L_k}, \text{ se a } k\text{-ésima formiga usa a trilha } (i,j) \text{ no seu } tour \\ 0, \text{ outros casos} \end{array} \right\} \tag{4.7}$$

onde Q é uma constante de projeto e L_k é o comprimento do *tour* da k-ésima

formiga. Este valor, avaliado quando a formiga completa um *tour* no tempo [t_0, $t_0 + n$] e consiste de um ciclo de n iterações, é então utilizado para atualizar a quantidiade e substância depositada previamente na trilha, com base em

$$\tau_{ij}^{k}(t+n) = \rho \cdot \tau_{ij}^{k}(t) + \Delta\tau_{ij}^{k} \tag{4.8}$$

onde r é um coeficiente que representa a persistência da trilha durante o ciclo (entre o tempo t e t+n), usualmente definido heuristicamente; o valor de *(1-r)* representa a evaporação da trilha entre o tempo t e t+n, onde

$$\Delta\tau_{ij} = \sum_{k=1}^{m} \tau_{ij}^{k} \tag{4.9}$$

4.10 Sistemas híbridos inteligentes

O progresso da tecnologia dos sistemas inteligentes é motivado pela necessidade do desenvolvimento de estratégias flexíveis e eficientes com o propósito de tratar aplicações complexas do mundo real. Cada metodologia da inteligência computacional (*soft computing*) possui potencialidades e limitações que as tornam mais adequadas para algumas aplicações particulares. Por exemplo, enquanto as redes neurais são aptas às tarefas de reconhecimento de padrões, não são eficientes para

interpretação lingüística. Os sistemas nebulosos ou *fuzzy systems* podem tratar informação imprecisa e são adequados em explanar suas decisões. Os AEs, conforme comentado anteriormente, são métodos de otimização para o tratamento de projetos complexos, onde, usualmente, empregam-se procedimentos heurísticos do tipo tentativa e erro, mas demandam uma elevada complexidade computacional.

Estas limitações são a principal motivação da criação de sistemas híbridos inteligentes (SHI), onde duas (ou mais) técnicas são combinadas para superar as limitações das técnicas quando tratadas individualmente. Os SHIs são relevantes quando considera-se domínios complexos que apresentam problemas com componentes diferentes, os quais requerem diversos tipos de processamento [65].

Os pesquisadores da inteligência computacional objetivam atender a estas necessidades com o desenvolvimento de sistemas que combinam as vantagens de algumas metodologias através da configuração de SHIs. Os SHIs combinando sistemas nebulosos, redes neurais, algoritmos evolutivos e sistemas especialistas são eficientes em uma ampla variedade de problemas reais. Os SHIs são relevantes quando se considera a natureza variada das aplicações. As formas dos sistemas híbridos inteligentes analisados incluem:

- sistemas evolutivo-nebulosos;
- sistemas evolutivo-neurais;
- sistemas neuro-nebulosos; e
- sistemas neuro-nebuloso-evolutivos.

A utilização de SHIs está crescendo rapidamente com aplicações em áreas, tais como: controle de processos, projetos de engenharia, mercado financeiro, controle de qualidade, diagnóstico de falhas, avaliação de crédito, diagnóstico médico e simulação cognitiva [86].

4.10.1 Classificação dos sistemas híbridos inteligentes

Na literatura são mencionadas as formas de classificação para os SHIs. Quanto à funcionalidade, Medsker & Bailey [109] classifica-os em:

(i) *modelo independente*: composto por módulos independentes, não existindo nenhuma interação entre os módulos. Este modelo não é uma proposta de integração propriamente dita, mas objetiva a comparação do desempenho entre as técnicas aplicadas a um problema específico;

(ii) *modelo com transformação*: é similar ao modelo independente. Entretanto, a diferença é que este inicia o processamento com uma metodologia e termina com outra;

(iii) *modelo com acoplamento livre*: é uma primeira forma de integração verdadeira. O problema é decomposto em sistemas separados que se comunicam através de arquivos de dados. Entre as variações deste modelo têm-se: pré-processadores, pós-processadores e co-processadores;

(iv) *modelo com acoplamento rígido*: é similar ao modelo com acoplamento livre. Neste modelo os dados são transferidos de um sistema para outro e ficam residentes na memória; e

(v) *modelo de integração completa*: este modelo compartilha diferentes componentes, tais como: representação e estruturas de dados.

Outra classificação, quanto à funcionalidade, é de Goonatilake & Khebbal [65] em relação a:

(i) *substituição de funções*: está relacionada à composição funcional de uma única metodologia inteligente. A função principal de uma metodologia é substituída por outra metodologia de processamento inteligente. Exemplo: otimização de um sistema nebuloso (ou uma rede neural) através de algoritmos evolutivos;

(ii) *intercomunicação*: são módulos independentes de processamento inteligente que trocam informações. Neste caso um problema pode ser decomposto em diversas tarefas menores, onde as metodologias são aplicadas independentemente da resolução de cada parte do problema;

(iii) *polimorfismo*: são sistemas que utilizam uma arquitetura de processamento única para a realização da funcionalidade de metodologias de diferentes processamentos inteligentes. Exemplo: redes neurais funcionando como se estivessem realizando uma otimização genética.

Sob outro ponto de vista, Schaffer [140] classifica os SHIs em duas classes:

(i) *combinações de suporte*: um algoritmo ajuda o outro. Exemplo: preparação dos dados através de um algoritmo para ser utilizado por outro;

(ii) *combinações colaborativas*: dois (ou mais) algoritmos cooperam para a obtenção de uma conclusão.

A seguir são apresentadas algumas possíveis combinações de algoritmos evolutivos com redes neurais e sistemas nebulosos.

4.10.2 Redes neurais artificiais

As redes neurais artificiais consistem em elementos de processamento altamente interconectados denominados neurônios. Cada um tendo um número de entradas e uma saída. A saída de cada neurônio é determinada como uma função não-linear de um soma ponderada das entradas, embora operações matemáticas mais complexas poderiam ser incluídas. Os neurônios se interconectam através de pesos, os quais são ajustados durante o período de treinamento. Entre as características relevantes das redes neurais têm-se: processamento paralelo, aprendizado, memória associativa e distribuída. Estas características são inspiradas nas redes neurais biológicas, mesmo que rudimentarmente.

As redes neurais e os AEs são metodologias úteis para o aprendizado e otimização fundamentadas em inspirações diferenciadas dos sistemas biológicos. Estas metodologias são de auto-aprendizado, mas utilizam estrutura e base matemática diferentes.

As redes neurais são métodos de aprendizado indutivo, enquanto os algoritmos evolutivos utilizam aprendizado dedutivo e requerem uma função de aptidão para avaliação das soluções do problema [98]. O procedimento de desenvolvimento de redes neurais para uma aplicação inclui estágios de projeto, a citar:

(i) a seleção do domínio do problema;

(ii) a configuração da rede neural (número de unidades a serem utilizadas, conexões entre os neurônios e outros parâmetros estruturais);

(iii) a escolha do algoritmo de treinamento da rede neural; e

(iv) a análise das propriedades de convergência, a habilidade de resolver o problema e a capacidade de generalização [142].

A evolução pode ser introduzida nas redes neurais em diversos níveis, incluindo-se: (i) evolução das regras de aprendizado, (ii) evolução de arquiteturas e (iii) evolução de pesos. Existem discussões se a evolução das regras de aprendizado estariam no nível mais alto dos três níveis de evolução apresentados. Um estudo sobre as possíveis combinações é apresentado em [173], onde foram referenciados 319 artigos relevantes sobre os SHI neuro-evolutivos.

4.10.3 Sistemas nebulosos (*fuzzy systems*)

Os fundamentos teóricos dos conjuntos nebulosos foram propostos por Zadeh [176], [177], professor da Universidade de Califórnia (Berkeley, Estados Unidos), como uma forma alternativa de modelar os sistemas complexos, se diferenciando das técnicas convencionais. Os sistemas nebulosos possuem um formalismo para a representação do conhecimento e inferência de novos conhecimentos que é similar à maneira utilizada pelos seres humanos para expressarem o conhecimento e raciocínio, ao contrário dos sistemas baseados em lógica clássica.

Existe um crescente interesse na integração destes tópicos. A classificação bibliográfica de 562 referências tratando a combinação de sistemas nebulosos e algoritmos evolutivos nas mais diversas áreas foi realizada por [32], [33]. Em muitos problemas de identificação e controle de processos, é possível encontrar um sistema nebuloso através de conhecimento heurístico e pela utilização de heurísticas de tentativa e erro para ajustes no projeto. Entretanto, esta forma de projeto consiste indubitavelmente de uma tarefa tediosa, pois existem diversos parâmetros a serem sintonizados.

Os sistemas nebulosos e os algoritmos evolutivos têm diferentes vantagens. Por exemplo, os sistemas nebulosos são aptos para a representação de diferentes formas de conhecimento, modelam as interações e as relações entre as variáveis do sistema e são apropriados à representação de problemas de natureza qualitativa do conhecimento.

Sob outro ponto de vista, os algoritmos evolutivos apresentam características promissoras quanto à capacidade de aprendizado e otimização de busca global. Os algoritmos evolutivos são ferramentas de otimização de propósito geral e, desta forma, não requerem conhecimento de "como" resolver o problema. Geralmente deve ser dotado de apenas uma função de aptidão para que cada solução potencial possa ser avaliada. Os sistemas nebulosos e algoritmos evolutivos podem ser combinados de diversas formas à obtenção de diferentes tipos de SHIs, por exemplo:

(i) *SHIs evolutivo-nebulosos*: visam a utilização de sistemas nebulosos para melhorar o comportamento ou mesmo a configuração de componentes dos algoritmos evolutivos. Trabalham com problemas de gerenciamento, em ambientes imprecisos, onde a imprecisão é modelada por sistemas nebulosos; e

(ii) *SHIs nebuloso-evolutivos*: constitui-se da aplicação de algoritmos evolutivos em problemas envolvendo sistemas nebulosos.

4.11 Outras abordagens relevantes

Outras abordagens relevantes que são tendências em algoritmos evolutivos são [6], [12]:

- sistemas evolutivos "criativos";
- embriologia computacional;
- teoria de autômatos celulares evolutivos;
- concepções de vida artificial combinada a algoritmos evolutivos; e
- algoritmos culturais.

4.12 Exercícios

1. Comente sobre abordagens emergentes de algoritmos evolutivos e suas potencialidades.
2. Comente sobre possíveis aplicações de sistemas híbridos inteligentes combinando algoritmos evolutivos com sistemas nebulosos e redes neurais.
3. Pesquise e cite outras abordagens emergentes de algoritmos evolutivos diferentes das mencionadas neste capítulo.

Capítulo 5

Aplicações de algoritmos evolutivos na academia e indústria

5.1 Introdução

Entre os exemplos de tipo de aplicações bem sucedidas de algoritmos evolutivos tem-se: sistemas de controle de processos, mineração de dados, sistemas tolerantes a falhas, aprendizado de máquina, problemas de escalonamento, identificação de sistemas, otimização não-linear, análise e otimização de sistemas com múltiplos objetivos, projeto de *hardware*, entre outras. A seguir são apresentados breves comentários sobre algumas destas aplicações.

5.2 Aplicaçoes de algoritmos evolutivos na academia

5.2.1 Aplicações em previsão de séries temporais e identificação de sistemas

A tentativa de explicar ou reproduzir os comportamentos dos sistemas físicos é algo que há tempo desperta o interesse de pesquisadores. Com o desenvolvimento dos processos industriais e a necessidade de controlá-los, é preciso desenvolver modelos que reproduzam suas características estáticas e dinâmicas. A previsão de séries temporais e a identificação de processos tem relevância, pois pode-se prever o que acontece a um processo, conhecendo a(s) entrada(s) e/ou a(s) saídas anteriores, disponíveis do processo.

A identificação de processos é o procedimento de identificar um modelo de um processo desconhecido, para propósitos de previsão e/ou compreensão do comportamento do processo. A complexidade inerente de muitos processos reais (multi-

variáveis, não lineares e variantes no tempo) dificulta a aplicação de técnicas convencionais de identificação e modelagem de sistemas.

A identificação de processos é, muitas vezes, como um problema de otimização, envolvendo algumas medidas para a adequação dos modelos candidatos a representar um processo. A escolha dos modelos matemáticos e o ajuste dos parâmetros são influenciados por fatores, entre os quais:

(i) o conhecimento *a priori* do sistema (linearidade, grau de não-linearidade, atraso de transporte);

(ii) as propriedades do modelo do sistema identificado (complexidade);

(iii) a escolha da medida de erro a ser minimizada; e

(iv) a presença de ruídos.

A identificação de sistemas é um exercício que envolve múltiplos e conflitantes objetivos, tipicamente a complexidade do modelo, o(s) critério(s) de desempenho e a validação que influenciam a seleção da estrutura do modelo matemático. Existem diversas razões para manter a ordem do modelo tão baixa quanto possível. Os critérios de informação podem ser introduzidos para combinar a adequação e os princípios fundamentais de construção de modelos, tais como:

- *princípio da redução de dados*: o menor número de variáveis deve ser utilizado para explicar uma quantidade máxima de informação;
- *princípio da parcimônia* (ou razão de Occam): os melhores modelos são obtidos utilizando-se as estruturas aceitáveis simples, contendo o menor número de parâmetros.

Entre os critérios utilizados destacam-se: informação Bayesiana, Akaike ou *minimum description length*, que combinam a variância residual e a ordem do modelo. O objetivo do algoritmo de otimização é a minimização de um critério de desempenho. Se todas as restrições e as condições forem atendidas, o modelo encontrado é aceito. Caso contrário, se uma das condições impostas é violada, o procedimento de determinação do modelo, de estimação de parâmetros e diagnóstico do modelo deve ser repetido até que seja encontrado um modelo apropriado.

Na maior parte das vezes as técnicas clássicas de identificação de processos, são, em essência, técnicas de busca local guiada por gradiente e necessitam de um espaço de busca regular ou uma função objetivo diferenciável. Estes métodos convencionais podem falhar na obtenção de um ótimo global, se o espaço de busca do modelo não é diferenciável. Adicionalmente, os métodos convencionais de identificação apresentam as seguintes desvantagens:

(i) alguma informação inicial dos parâmetros do processo é necessária à convergência;

(ii) os parâmetros estimados podem ser tendenciosos, se o ruído é correlacionado;

(iii) a dificuldade na identificação do atraso de transporte; e

(iv) a aplicação em sistemas não-lineares requer algumas alterações no algoritmo.

Neste contexto os algoritmos evolutivos podem ser úteis em procedimentos de seleção de estrutura, estimação de parâmetros e determinação de ordem de polinômios vinculados a pólos e zeros de sistemas dinâmicos [49], [25], [175], [94].

5.2.2 Aplicações em controle de processos industriais

Muitos dos processos industriais a serem controlados são complexos, em larga escala, não-lineares, não-estacionários e estocásticos. Os métodos analíticos clássicos, tais como diagramas de Nyquist, lugar das raízes e H_∞ tem sido utilizados para o tratamento de tais processos através das inerentes propriedades de robustez dos algoritmos. Tais métodos são aplicáveis para processos que podem apresentar uma invariância linear no tempo relativamente pequena e com limites conhecidos. Entretanto, muitos processos com tais complexidades não seguem a representação de modelos matemáticos analíticos, lineares e invariantes no tempo, o que dificulta o controle de processos complexos.

Segundo Henson & Seborg [72], nos últimos anos, existe um ressurgimento de interesse no desenvolvimento de estratégias de controle aprimoradas e estratégias de identificação de sistemas não lineares motivado por diversos fatores, tais como:

- avanços da teoria de sistemas não lineares, ocasionando metodologias de projeto aplicáveis a uma extensão de problemas de controle não linear;
- desenvolvimento de métodos de identificação eficientes, para modelos não lineares empíricos, e vasta aplicabilidade em pacotes computacionais, disponíveis comercialmente;
- desenvolvimento continuado das capacidades de *software* e *hardware*, tornando possível a incorporação de modelos não lineares complexos aos sistemas de controle.

As estratégias de controle avançadas permitem o aprimoramento do desempenho dos sistemas de controle, se comparada com as estratégias convencionais. Mas, usualmente, o projetista necessita configurar um grande número de parâmetros que, em alguns casos, pode dificultar o domínio do conhecimento de usuários, que não são especialistas na utilização destas técnicas de controle. Também, é necessária a utilização de diferentes pacotes ou mesmo o desenvolvimento de um sistema para a implementação de diferentes estratégias de controle [114]. Entretanto, o projeto adequado de metodologias de controle avançadas, considerando um compromisso entre desempenho e complexidade, pode oferecer uma ferramenta acessível e eficiente para a comunidade de controle atuante no meio industrial.

Em muitos destes casos justifica-se a utilização de projetos de otimização de controladores avançados baseados em AEs. O projeto e a configuração em sistemas de controle utilizando-se de AEs têm abrangido uma variada gama de aplicações. Os

AEs são uma ferramenta robusta em configuração e projeto de sistemas de controle, todavia geralmente é empregada através de um procedimento *off-line*.

A sintonia do controlador de três parâmetros PID (proporcional, integral e derivativo) [97], controlador por alocação de pólos [94], controle de estrutura variável, controle robusto, controle adaptativo [83] e controle preditivo [130], [102] têm sido descritas e analisadas por vários pesquisadores e aplicada em diversos processos do meio industrial [89], [80], [24].

5.2.3 Aplicações em robótica

A área de robótica tem evoluído rapidamente desde a década passada, devido ao acentuado aumento do poder computacional e disponibilidade de uma grande variedade de sensores. Isto pode ser observado no fato dos seres humanos terem enviado robôs para Marte, para o fundo dos oceanos, aplicado robôs dentro de reatores nucleares, linhas de manufatura, inspeção de dutos, além da comercialização de brinquedos infantis (brinquedos da Sony, por exemplo, Aibo) e educacionais (por exemplo, kits LEGO e Khepera).

Outro indicador deste fato é que as vendas anuais de robôs industriais têm crescido nos Estados Unidos à taxa de aproximadamente 25% ao ano. Além disso, outro aspecto relevante é que os robôs cada vez mais têm sido dotados da capacidade de aprender, atuar autonomamente, e interagir com os humanos e seu ambiente.

A área de robótica pode ser dividida em duas abordagens: (i) a robótica de manipuladores e (ii) a robótica móvel. Os AEs podem ser utilizados para o projeto de sistemas de controle de todos estas categorias, principalmente através de simulação, devido a complexidade computacional dos AEs para problemas de controle de manipuladores em tempo real.

As pesquisas com AEs na área de robótica móvel (área denominada de robótica evolutiva) têm evoluído quanto a aplicações para: (i) construção de mapas do ambiente em tempo de execução, (ii) configuração de métodos reativos e híbridos para a navegação de robôs, (iii) tratamento de imprecisão e fusão de sensores, (iv) interfaces amigáveis, (v) cognição, aprendizado e coevolução, (vi) arquiteturas de controle para robôs autônomos , (vii) cooperação entre robôs, e (viii) automação industrial e manufatura [21], [75], [116], [107], [123], [167].

5.2.4 Aplicações em sistemas de manufatura

Os tipos de problemas em sistemas de manufatura onde os AEs tem sido utilizados podem ser agrupados em problemas de escalonamento e otimização [41]. A figura 5.1 ilustra as áreas de sistemas produtivos onde as abordagens utilizando AEs têm sido aplicadas.

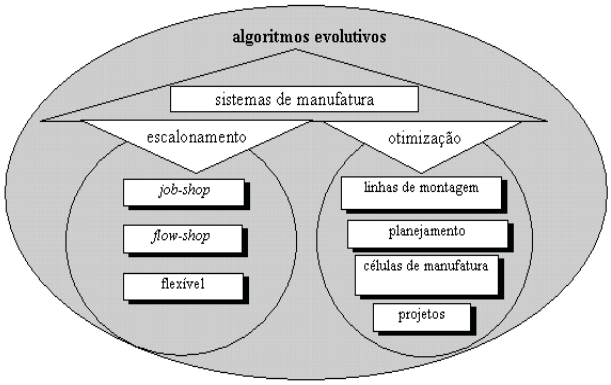

Figura 5.1: Aplicações de algoritmos evolutivos em sistemas produtivos

Entre as aplicações de AES destacam-se as seguintes:

- *escalonamento job-shop*: O problema de escalonamento *job-shop* consiste em ordenar n tarefas (*jobs*) para serem processados em m máquinas. Cada *job* envolve um número de operações de máquina diferentes.

- *escalonamento flow-shop*: O problema de escalonamento *flow-shop* (ou problema de sequenciamento de tarefas) é um outro problema de otimização em manufatura que tem atraído particular interesse de pesquisa. A aplicação de AEs para este problema é fácil, desde que ele pode ser formulado com um problema do tipo caixeiro viajante com representação de caminhos. O problema de sequenciamento de tarefas envolve a ordenação de n jobs para serem processados em m máquinas. A diferença entre problema de escalonamento *job-shop* e escalonamento *flow-shop* é que no último caso é submetido ao mesmo sequenciamento de máquinas, enquanto a seqüência de operações é a mesma em cada máquina. Isto significa que a solução do problema pode ser representada como uma permutação de todas as tarefas.

- *escalonamento flexível:* Os AEs, principalmente os AGs, têm sido aplicados para problemas de escalonamento. Entre estes algoritmos para *job-shops*, uma usual consideração é que as rotas das tarefas para as máquinas são fixas, mas isto não é verdade para sistemas flexíveis de manufatura, onde os jobs têm flexibilidade de rota para as máquinas. Existem outras aplicações relevantes de AEs ligadas a otimização de sistemas produtivos, principalmente em abordagens para configuração de:

(a) *linhas de montagem*: Diversos problemas de otimização são associados com linhas de montagem, tais como: problema de planejamento de seqüência de montagem, sequenciamento de linhas de montagem de modelo fixo e problemas de balanceamento de linhas de montagem.

(b) *planejamento e projetos*. O projeto é uma fase complicada e que consome tempo no desenvolvimento de um produto. Geralmente existe um considerável esforço devotado ao desenvolvimento de sistemas de CAD (*Computer Aided Design*) de forma a simplificar e aumentar a velocidade do procedimento de projeto. Os AEs têm sido aplicados com sucesso para problemas complexos de otimização de projeto, substituindo procedimentos heurísticos de tentativa e erro.

(c) *células de manufatura*. A manufatura de células consiste da aplicação da tecnologia de grupos em sistemas de manufatura. Os AEs têm sido aplicados nos três estágios do projeto de células de manufatura, a citar: (i) problema da formação de células (agrupamento de máquinas em células), (ii) layout de células na planta, e (iii) layout das máquinas com as células. A principal motivação da aplicação de AEs, neste caso, é que a implementação de cada um destes estágios, onde métodos de otimização tradicional são incapazes de encontrar soluções ótimas em tempo razoável.

5.2.5 Aplicações em otimização de funções matemáticas

A otimização pode ser definida como o procedimento de selecionar um projeto superior baseado em alguns critérios pré-definidos de um conjunto de projetos alternativos factíveis [129]. Existem diversas formas de classificar os problemas de otimização. Algumas das classificações mencionadas na literatura são descritas a seguir [129], [135].

- *classificação baseada no número de parâmetros*: Os problemas de otimização podem ser classificados em unidimensionais e multi-dimensionais baseados no número de parâmetros envolvidos no problema.

- *classificação baseada no número de restrições*: Um problema de otimização pode ser classificado como com restrições e sem restrições, dependendo se existem restrições para a resolução do problema.

- *classificação baseada no número de funções objetivo*: Dependendo do número de funções objetivo no problema de otimização, o problema pode ser classificado como de objetivo simples ou com múltiplos objetivos.

- *classificação baseada na natureza do espaço de busca*: A natureza do espaço de busca também define uma classificação dos problemas de otimização em (i) espaço de busca conhecido e (ii) espaço de busca desconhecido. Neste caso, pode-se também adotar classificações quanto ao número de soluções ótimas que o problema tem em problema unimodal e multimodal.

- *classificação baseada na natureza das funções objetivo*: A função objetivo envolve um problema de otimização quantitativa e/ou qualitativa.

- *miscelânea de classificações*: Existem diversas outras classificações de problemas de otimização, tais como as baseadas na: (i) natureza das equações envolvidas (linear, não-linear, geométrica e quadrática), (ii) separabilidade das

funções (separavel ou não-separável), (iii) natureza das variáveis de projeto (estáticas ou dinâmicas) e (iv) permissibilidade de valores para as variáveis de projeto (inteiras ou reais).

Os métodos de otimização determinísticos (*hill-climbing*, *branch and bound*, algoritmos *greedy*) são eficientes para otimização de funções convexas, contínuas, unimodais com boa precisão nos resultados obtidos com custo computacional pequeno.

Por outro lado, os métodos probabilísticos ou baseados em transições aleatórias (*simulated annealing*, AEs, Monte Carlo, busca tabu) são uma classe de métodos que tratam funções unimodais e multimodais, funções convexas e não convexas (ver figura 5.2), funções contínuas e descontínuas, mas o preço de serem geralmente mais genéricos é que os resultados são menos precisos (e necessitam de uma rigorosa análise estatística) e possuem elevada complexidade computacional.

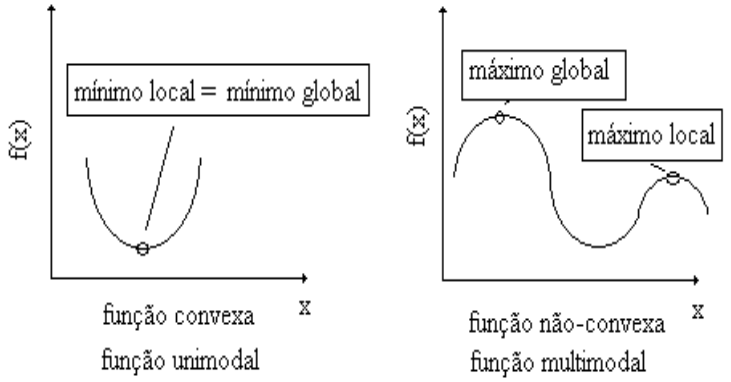

Figura 5.2: Alguns tipos de funções

Benchmark de funções matemáticas complexas

Apesar alguns estudos quanto a provas de convergência de AEs e outros métodos de otimização existirem, muitas das taxas de convergência e qualidade de cada abordagem de otimização pode somente ser medida ou comparada através de testes *benchmark*. Os *benchmarks* que podem ser utilizados para acessar o mérito e o desempenho destes algoritmos devem ser simples de utilizar, vastamente utilizados e confiáveis.

Neste caso, a área de otimização de funções matemáticas é relevante pois serve de base para análise de AEs quanto a: (i) otimalidade, (ii) precisão, (iii) sensibilidade, (iv) convergência e (v) complexidade computacional, através do número de operações de ponto flutuante e/ou tempo de CPU. Na literatura tem sido propostas várias funções *benchmarks* para análise de algoritmos de otimização, tais como [6], [110]:

- função de Schaffer;

- função de Goldstein-Price;
- função de Branin;
- função de Shekel;
- função de Shubert;
- função de Stuckman;
- função de Bohachevsky;
- função de Colville;
- função de Rastringin;
- função de Schwefel;
- função de Griewangk;
- função de Akley.

A seguir são mencionadas alguns exemplos de outras funções usuais à análise do desempenho de AEs.

- função de De Jong (ver figura 5.3): função contínua, convexa e unimodal.

$$f_1(x) = \sum_{i=1}^{n} x_i^2, \text{ onde } -5,12 \leq x_i \leq 5,12 \tag{5.1}$$

com mínimo global $f_1(x)$=0 para x_i=0, onde i=1,...,n.
Figura para $n = 2$

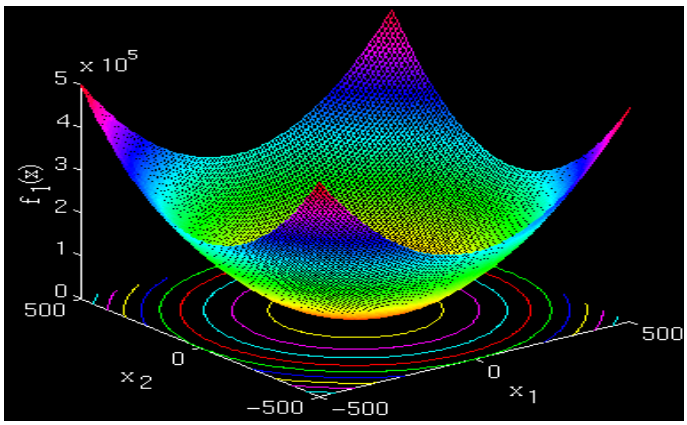

Figura 5.3: Função de De Jong

- função de Rosenbrock ou função Banana (ver figura 5.4):

$$f_2(x) = \sum_{i=1}^{n} 100\left(x_{i+1} - x_i^2\right)^2 + (1 - x_1)^2, \text{ onde } -2,048 \leq x_i \leq 2,048 \qquad (5.2)$$

com mínimo global $f_2(x)$=0 para x_i=1, onde i=1,...,n.

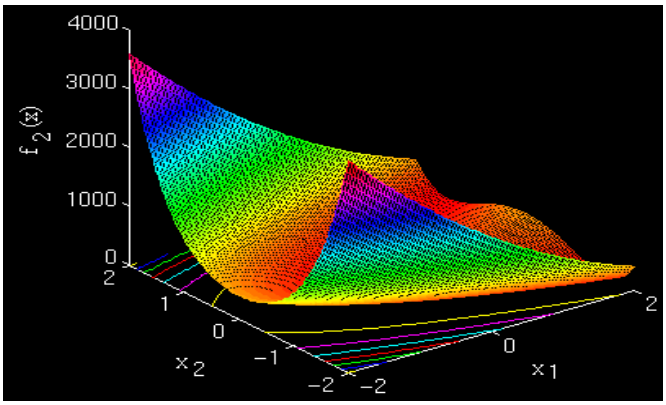

Figura 5.4: Função de Rosenbrock (função banana)

- função de Easom (ver figura 5.5): função unimodal com ótimo global em uma área restrita

$$f_3(x) = -cos(x_1)cos(x_2)e^{-(x_1-\pi)^2-(x_2-\pi)^2}, \ -100 \leq x_i \leq 100, \text{ para } i = 1, 2. \quad (5.3)$$

com mínimo global $f_3(x)$=0 para (x_1, x_2)=(π, π), onde i=1,...,n.

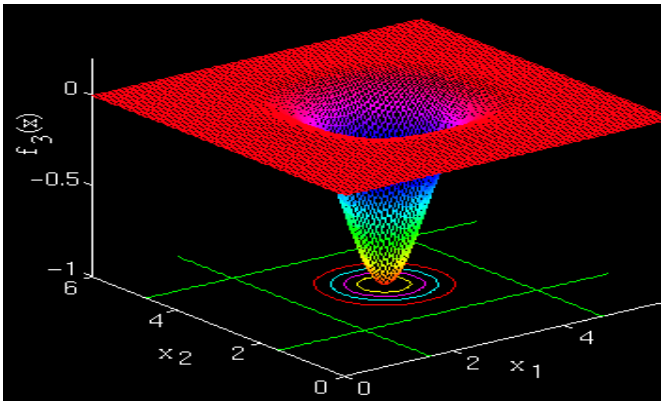

Figura 5.5: Função de Easom

Otimização com múltiplos objetivos

Os aspectos relativos a otimização multiobjetivo (ou otimização com múltiplos critérios) e esquemas baseados na definição de otimalidade de Pareto têm sido alvo de pesquisas emergentes por diversos grupos de pesquisa.

A otimização com múltiplos objetivos constitui-se por tratar-se de um tópico de pesquisa importante para ambos cientistas e engenheiros, não somente por causa da natureza de múltiplos objetivos de muitos problemas do mundo real, mas também porque existem muitas questões em aberto nesta área [38], [28], [44], [54], [55], [30].

De fato, não existe sempre uma definição universalmente aceita do "ótimo" de uma otimização com um único objetivo, o que dificulta a comparação de resultados de um método para outro, pois usualmente a decisão sobre o qual é a "melhor" resposta está vinculada ao responsável pela tomada da decisão (ver figura 5.6).

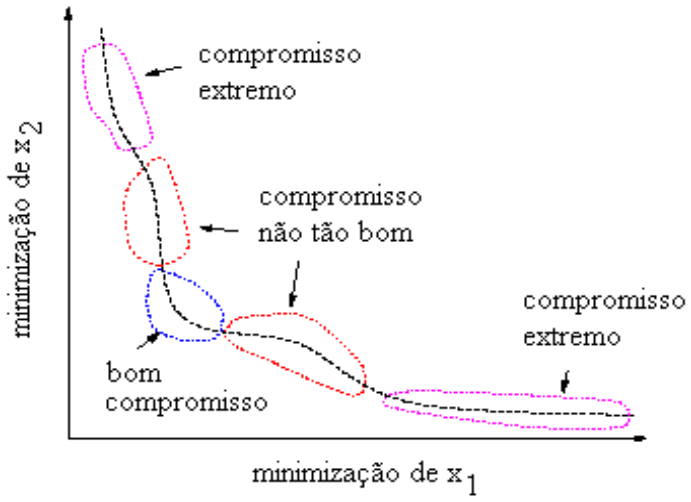

Figura 5.6: Exemplo de um problema de minimização de duas variáveis

Na literatura, nos últimos anos, tem sido relatadas várias aplicações de AEs em engenharia hidráulica, projeto naval, engenharia ambiental, engenharia elétrica e eletrônica, telecomunicações, robótica, controle de processos, engenharia civil, transportes, engenharia aeronática, finanças, geografia, química, física, medicina, bioinformática e manufatura (escalonamento, *layout*, localização de facilidade e gerenciamento) [30].

Sob outro ponto de vista, a nível acadêmico tem sido propostas várias funções *benchmark* para análise de desempenho de AEs para problemas com múltiplos objetivos. A seguir são mencionados alguns exemplos destas funções.

- função proposta por Binh e Korn [14]: função convexa,

$$F(x) = [f_1(x,y), f_2(x,y)], \quad \text{com} \tag{5.4}$$

$$f_1(x,y) = x^2 + y^2, \tag{5.5}$$

$$f_2(x,y) = (x-5)^2 + (y-5)^2, \tag{5.6}$$

onde as restrições são: -5 $\leq x$ e $y \leq 10$.

- função proposta por Fonseca e Fleming [53]: função côncava,

$$F(x) = [f_1(x,y), f_2(x,y)], \quad \text{com} \tag{5.7}$$

$$f_1(x,y) = 1 - \exp\left[-(x-1)^2 - (y+1)^2\right], \tag{5.8}$$

$$f_2(x,y) = 1 - \exp\left[-(x+1)^2 - (y-1)^2\right], \tag{5.9}$$

e não existem restrições.

- função proposta por Lis e Eiben [99]: função côncava

$$F(x) = [f_1(x,y), f_2(x,y)], \quad \text{com} \tag{5.10}$$

$$f_1(x,y) = \sqrt[8]{x^2 + y^2}, \tag{5.11}$$

$$f_2(x,y) = \sqrt[4]{(x-0,5)^2 + (y-0,5)^2}, \tag{5.12}$$

onde as restrições são: -5 $\leq x$ e $y \leq 10$.

- função proposta por Viennet et al. [165] :

$$F(x) = [f_1(x,y), f_2(x,y), f_3(x,y)], \quad \text{com} \tag{5.13}$$

$$f_1(x,y) = \frac{(x-2)^2}{2} + \frac{(y+1)^2}{13} + 3, \tag{5.14}$$

$$f_2(x,y) = \frac{(x+y-3)^2}{36} + \frac{(-x+y+2)^2}{8} - 17, \tag{5.15}$$

$$f_3(x,y) = \frac{(x+2y-1)^2}{175} + \frac{(2y-x)^2}{17} - 13, \tag{5.16}$$

onde as restrições são: -4 $\leq x$ e $y \leq 4$.

Otimização com restrições

Os problemas encontrados no mundo real possuem tipicamente propriedades de: (i) múltiplas metas, (ii) funções objetivo com ruídos e variantes no tempo, (iii) dados mal-estruturados, (iv) restrições complexas. O tratamento de problemas com restrições afeta o espaço de busca, pois as restrições dividem o espaço de busca em soluções factíveis e não-factíveis (ver figura 5.7) [29], [141], [157].

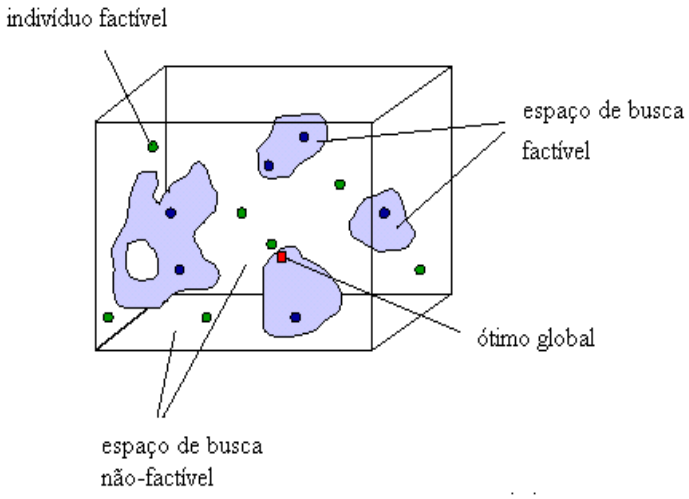

Figura 5.7: Problemas com restrições

Neste contexto, os AEs podem ser úteis no tratamento de problemas não-lineares com restrições, onde são utilizadas freqüentemente funções de penalização para lidar com restrições rígidas e pouco aceitáveis (*hard and soft constraints*). A idéia da aplicação de funções de penalização é ponderar de forma a "degradar" a qualidade de soluções não-factíveis. A literatura é rica em abordagens de penalização estáticas, dinâmicas, adaptativas e com uso de subpopulações de AEs.

5.2.6 Aplicações no suporte à configuração e otimização de projetos

Apesar de existirem evidências de que a utilização de tecnologias de AEs e metodologias de projeto adaptativas serem promissoras para otimização de sistemas, existe ainda atualmente um pequeno reconhecimento (ou investimento) na exploração de suas capacidades busca e otimização no projeto de equipamentos industriais.

Tais capacidades suportam um integração sinergística com procedimentos de projeto preliminar e conceitual para auxiliar a busca com espaços de projeto pré-definidos enquanto também permite a exploração em áreas não bem definidas que podem estar sujeitas a presença de restrições iniciais, objetivos severos e restrições

nas variáveis de projeto.

Uma interação entre um projetista (ou uma equipe de projetistas) com estratégias de projeto presentes nos AEs podem resultar em uma exploração significativa envolvendo processamento *off-line* de resultados iniciais e informação de projeto relatada levando a uma redefinição do ambiente de projeto. Tal redefinição e a utilização de conhecimentos de projetistas aliado a AEs podem resultar na descoberta de soluções de projeto inovadoras ou mesmo criativas. Não existe uma lista definitiva, mas os aspectos que devem ser considerados para que uma integração de técnicas de suporte usando AEs seja bem sucedida incluem:

- a habilidade de amostrar, de forma eficiente, espaços de projeto complexos descritos por diferentes modelos de representação e simulação, isto é, representar dados quantitativos, qualitativos, lingüísticos, incertezas e perturbações;
- a adição, remoção e/ou variação de restrições, objetivos e limites das variáveis de projeto;
- a identificação de múltiplas soluções/regiões de alto desempenho de espaços complexos;
- o desenvolvimento de sistemas de exploração/busca que possam capturar conhecimento de projetos específicos através da interação com o projetista;
- o processamento *on-line* da informação relacionada a múltiplos critérios de projeto relacionados a projeto, manufatura, economia e requerimentos de *marketing*;
- a habilidade de acessar regiões de projeto quer sejam factíveis e que auxiliem na identificação de soluções ótimas.

A importância de tais aspectos tem começado a ser evidente a partir de recentes pesquisas que relatam a integração de AEs com procedimentos de projeto [124], [125]. Entretanto, procedimentos de projeto preliminar/conceituais que sejam inteiramente baseados em AEs não são indicados pois atualmente com o estado da arte da teoria dos AEs não são considerados viáveis no meio industrial. Existem exemplos de aplicações industriais que usam AEs e fazem uso do conhecimento/interação de projetistas em projetos preliminares de aeronaves da Bristish Aerospace e na determinação de geometrias de turbinas da Rolls Royce.

O desenvolvimento de protótipos de produtos pode tambem fazer uso de integração adicional de outras técnicas da inteligência artificial, tais como sistemas nebulosos, redes neurais, agentes inteligentes, raciocínio baseado em casos e sistemas especialistas, que resultam em um aprimoramento significativo das capacidades de processamento e busca no suporte a engenheiros e projetistas.

5.2.7 Aplicações em mineração de dados (*data mining*)

A mineração de dados (*data mining)* consiste em um conjunto de conceitos e métodos com o objetivo de encontrar uma descrição, preferencialmente compreensível,

de padrões e regularidades em um determinado conjunto de dados. Os termos *data mining* e descoberta de conhecimento em base de dados (*Knowledge Discovery in Databases* - KDD) muitas vezes são confundidos como sinônimos para identificar o processo de descoberta de informação útil de bancos de dados.

O termo KDD foi estabelecido para enfatizar que conhecimento é o produto final de uma descoberta baseada em dados (*data-driven*). Desta forma KDD se refere a todo o processo de descoberta de conhecimento enquanto mineração de dados se refere a uma das etapas deste processo. As etapas do KDD envolvem preparação dos dados, seleção, limpeza, transformação, mineração de dados e interpretação dos resultados (ver figura 5.8) [48].

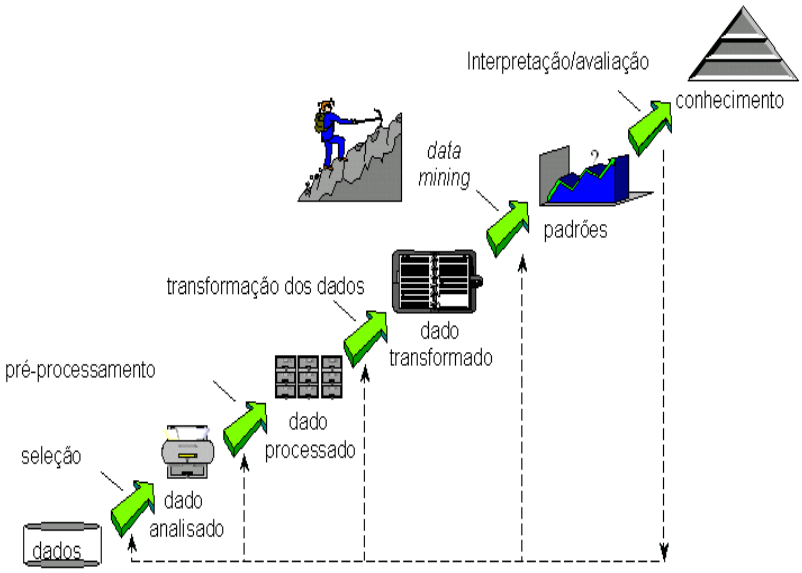

Figura 5.8: Etapas de KDD

Um padrão é definido como um tipo de declaração (ou modelo de uma declaração) sobre o conjunto de dados que está sendo analisado. Uma instância de um padrão é uma declaração em uma linguagem de alto nível que descreve uma informação interessante descoberta nos dados. A descoberta de relações nos dados compreende todas as instâncias de padrões selecionados no espaço das hipóteses que sejam suficientemente interessantes, de acordo com algum critério estabelecido.

As várias tarefas desenvolvidas em mineração de dados têm como objetivo primário a predição e/ou a descrição. A predição usa atributos para predizer os valores futuros de uma ou mais variáveis (atributos) de interesse. A descrição contempla o que foi descoberto nos dados sob o ponto de vista da interpretação humana [48].

Em síntese, a mineração de dados pode providenciar informações relevantes de

forma a auxiliar usuários na tomada de decisões. Uma adequada definição para *data mining* é "um método efetivo para descoberta de regularidade e anomalias em vários tipos de banco de dados".

As técnicas de data mining são utilizadas para extrair e resumir informações que estão "latentes" em banco de dados. Neste aspecto destacam-se as aplicações emergentes no suporte a tomada de decisão. Exemplos são na análise de risco, *marketing* direcionado, análise do comportamento de clientes, análise de estoques, previsão de fraude em cartões de crédito e gerenciamento de *portfolio* [56], [137] .

5.2.8 Aplicações em projeto de *hardware*

A área de *hardware* evolutivo tem demonstrado o sucesso da utilização de AEs para geração de circuitos digitais. A utilização e atual disponibilidade da tecnologia de *Field Programmable Array* (FPGA), ao contrário da utilização de simulação de circuitos, realiza operações de cálculo que propiciam a obtenção do valor do *fitness* que guia a otimização baseada em AEs.

Na eletrônica evolutiva (ou evolucionária), considera-se a síntese de circuitos como uma tarefa de busca em um espaço de possíveis circuitos (soluções), os quais devem satisfazer a diversos objetivos. As duas principais vantagens do emprego de computação evolucionária são a eliminação da necessidade de conhecimento prévio do projetista e a obtenção de uma nova classe de circuitos eletrônicos, diferentes daqueles projetados por técnicos e engenheiros. A primeira deriva do fato de que algoritmos genéticos iniciam a busca de soluções para problemas de forma aleatória, sem necessitar, em princípio, de inputs de especialistas; a segunda contribui para a aquisição de novas metodologias de projeto e permite, em muitos casos, a síntese de circuitos mais eficientes que os convencionais [178].

Uma nova classe de circuitos integrados analógicos, denominada *Field Programmable Analog Array* (FPAA), tem sido utilizada com sucesso em experimentos de eletrônica evolutiva. As plataformas reconfiguráveis pretendem estabelecer uma nova tendência na síntese evolucionária de circuitos eletrônicos, digitais ou analógicos, que possuam características de auto-reparo, auto-reconfiguração e auto-adaptação. Estas são características essenciais aos sistemas que precisam funcionar por muito tempo em ambientes hostis. Os sistemas com estas características são desejáveis na produção de equipamentos a partir de chips reconfiguráveis a fim de obter sistemas mais robustos e auto-reparáveis, visando a diminuição da taxa de equipamentos descartados por estarem fora das especificações e, finalmente, possibilitando adaptar tais sistemas a condições específicas. Tais circuitos são sintetizados, otimizados ou reparados através de métodos da computação evolutiva.

5.2.9 Algoritmos evolutivos inspirados na computação quântica

A computação quântica é uma área emergente de pesquisa que abrange todos os aspectos do desenvolvimento de computadores quânticos, desde a base clássica da informática (lógica, máquinas de Turing, estruturas de dados e algoritmos) até

problemas de otimização combinatória, supercomputação científica e teoria quântica comum a estudos da mecãnica quântica e física nuclear.

Neste contexto, os AEs inspirados em computação quântica podem ser caracterizados pela representação de um indivíduo (solução), uma função de avaliação e uma diâmica populacional. Entretanto a representação do indivíduo um Q-*bit*, definida como a menor unidade de informação, para a configuração de *strings* de Q-*bit* [69].

5.3 Aplicações "emergentes" na indústria

O crescimento da competição global está fazendo que vários seguimentos industriais visem a otimização de suas atividades para ganharem mercados. Entretanto, a complexidade e a falta de pesquisa sistemática na área de otimização de projetos de engenharia no meio industrial tem prevenido a exploração do potencial de diversas abordagens emergentes de otimização usadas na academia. Os dois maiores inibidores da utilização industrial de metodologias emergentes em larga escala são a falta de otimizadores robustos e a confiança do projetista na utilização de novas metodologias.

Neste caso a indústria tem na maior parte das vezes tratado os problemas de otimização de projeto com heurísticas de tentativa e erro ou através a adoção de simplificações, muitas vezes, exageradas de problemas complexos. Isto tem levado a uma perda de oportunidade de obtenção de projetos melhores, com custos reduzidos e ciclo de projeto reduzido. O crescimento da pesquisa na área de otimização de problemas "reais" da indústria usando AEs tem sido encorajado pelo desejo de seguir esta oportunidade.

Os problemas de projeto de engenharia "reais", ao contrário dos problemas teóricos (*benchmarks*), são aqueles encontrados na indústria. Alguns exemplos destes problemas são o projeto de estruturas de aeronaves com mínimo peso, projeto de superfície de automóveis para aprimoramento do *layout* e eficiência aerodinâmica, projeto de configurações de bombeamento, projeto de turbinas e equipamentos de transferência de calor para máxima eficiência.

Os problemas encontrados no meio industrial possuem várias características peculiares, tais como [134]:

- a principal característica dos problemas na indústria é a presença de mútiplas medidas de desempenho (ou objetivos) que devem ser otimizados simulateamente;
- a complexidade da maior parte dos problemas é incrementada pela presença de acoplamentos entre as variáveis de projeto (otimização);
- presença de variáveis reais e inteiras no mesmo problema;
- muitos problemas reais envolvem abordagens qualitaitvas como manufaturabilidade do produto e preferências especiais do projetista;

- adicionalmente os problemas requerem algumas restrições a serem satisfeitas;
- o custo computacional para resolução do problemas de otimização presentes na indústria é também aumentado pela presença de diversas soluções ótimas;
- falta de conhecimento *a priori* considerado na forma do espaço de busca é também geralmente observado nestes problemas. Não exise também nenhuma informação sobre o desempenho e a localização de pontos ótimos e sub-ótimos no espaço de busca;
- finalmente, o desenvolvimento de modelos mamteáticos para a solução de problemas de otimização na indústria é uma tarefa complexa.

Apesar de todas as características mencionadas, algumas aplicações industriais de AEs são uma realidade na Europa e Estados Unidos, principalmente na resolução de problemas de otimização, gerenciamento, economia, projeto, roteamento, escalonamento e reconhecimento de padrões. Os exemplos da aplicação de AEs existem em diversas áreas do conhecimento, contudo algumas das mais relevantes aplicações são resumidas nas figura 5.9 [46], [47].

aplicação	empresa
projeto de bulbos de lâmpadas e turbinas de avião	General Electric
ambiente de simulação de aviões	Beranek & Newmann
propulsão de navios	Naval Surface Weapons Center
roteamento de redes de comunicação	CNET, Cap Volmac, HP
escalonamento e controle	Rolls-Roice
gerenciamento	First Quadrant
aplicações comerciais	The Prediction Company
consumo de combustível de vários fornos	Courtaulds Films
suprimento de energia e gerenciamento de potência	Siemens AG
controle e tarefas de otimização em aplicações aeroespaciais	Daimler-Benz Aerospace
projeto de redes de telecomunicações	Nortel Smart Network
gerenciamento de bancos de dados distribuídos	BT Labs

produto	empresa	descrição
OMEGA	Cap Gemini e KiQ Ltd	modelos para previsão
Genetica, EvoSchool	Genetica Advanced Software	aplicações de escalonamento
Evolver	Palisade Corporation	problemas de otimização
AutoStat	AutoSimulations	simulação e otimização
SimRunner	PROMODEL Corp.	simulação e otimização

Figura 5.9: Algumas aplicações e ambientes comerciais com a utilização de AEs

5.4 Tendências das pesquisas atuais

Os AEs, nas suas configurações usuais, também apresentam dificuldades para a determinação do ótimo global, sem a utilização de uma metodologia de otimização local. O tratamento desta limitação é realizado através da configuração de abordagens de evolução Lamarckiana ou algoritmos meméticos [112]. Para obter-se os benefícios da configuração híbrida de busca global e local, uma forma eficiente é executar, inicialmente, um algoritmo evolutivo para localizar a região de ótimo global e após aplicar-se outra metodologia de otimização para a busca local (por exemplo: *simulated annealing*, quase-Newton, gradiente conjugado, Levenberg-Marquardt, método simplex, entre outras).

Os princípios evolutivos de Lamarck [118] e efeito Baldwin [73] têm sido combinados à AEs, visando aumentar a velocidade de convergência dos métodos frente a buscas locais. Outra abordagem relevante de AEs é no tratamento de problemas de divisão de recursos, algoritmos culturais, coordenação de agentes de *software* e cooperação em comunidades de robôs [68].

Diversos procedimentos de suporte a tomada de decisão e teoria de jogos baseados em conjuntos de estratégias e preferências têm sido abordados na literatura. Os AEs são abordagens com potencialidades para modelagem de racionalidade em teoria de jogos e tratamento de processos dinâmicos de equilíbrio e transição através de procedimentos de co-evolução [144].

A implementação em processamento paralelo e de agentes distribuídos é facilitada por aspectos de simplicidade de adaptação dos AEs à configuração de importantes mecanismos, tais como presa-predador, migração e modelos de difusão. A utilização de diversos processadores possibilita um aumento de ganho em tempo computacional, em relação a máquinas com processamento seqüêncial.

Existem algumas tendências relevantes quanto as aplicações e/ou combinações de AEs com outras abordagens. Entre as quais pode-se citar: dinâmicas complexas e co-evolução, *rough sets,* ambiente de projeto virtual baseado em agentes evolutivos, otimização por colônias de partículas, teoria do caos, interação com outras metodologias em problemas de multi-agentes, colônia de formigas associado a *data mining*, busca tabu, computação quântica combinada a AEs, *hardware* evolutivo, coloração de grafos, otimização inteira e mista inteira, controle inteligente, aprendizado de máquina, projeto e otimização estrutural.

Outras aplicações emergentes na indústria são a utilização de sistemas híbridos inteligentes usando procedimentos de projeto que usem métodos da *soft computing*. Neste caso, características de aprendizado, aquisição flexível de conhecimento, processamento de conhecimento, auto-organização, representação de incertezas podem ser exploradas em combinações de redes neurais, algoritmos evolutivos, sistemas nebulosos e teoria do caos.

As aplicações dos sistemas híbridos inteligentes estão presentes em diversas áreas. As aplicações comerciais usuais destes sistemas combinando redes neurais, sistemas nebulosos, e as vezes, otimização via AEs são em análise de mercado financeiro (Nikko Securities), ventilador elétrico (Sanyo), máquina fotocopiadora (Sanyo), máquina de lavar (Toshiba, Sanyo, Hitachi) e agentes autonômos para

agricultura [67].

Em um panorama mais geral estes sistemas ditos "inteligentes" possuem diversas potencialidades adaptativas, autonomia, suporte a decisão, otimização e funções emergentes para o desenvolvimento de inovações no meio industrial. Estas abordagens estão crescendo rapidamente com aplicações em áreas, tais como: controle de processos, projetos de engenharia, mercado financeiro, controle de qualidade, diagnóstico de falhas, aviação, sistemas de potência, eletrônica de potência, transportes, avaliação de crédito, diagnóstico médico e simulação cognitiva [148], [70], [16], [121].

5.5 Exercícios

1. Comente brevemente sobre as potencialidades e limitações dos algoritmos evolutivos para problemas presentes no meio industrial.

2. Mencione 20 aplicações potenciais para resolução com algoritmos evolutivos.

5.6 Material de apoio e pesquisa

- Efeito Baldwin:

  http://www.cs.bath.ac.uk/~jjb/web/baldwin.html

- Programação genética: *The Genetic Programming Tutorial Notebook*

  http://www.geneticprogramming.com/Tutorial/index.html

- *An overview of genetic algorithms: Part 1, fundamentals*:

  ftp://ftp.cs.wayne.edu/pub/EC/GA/papers/over93.ps.gz

- *Evolutionary Computation FAQ* - David Beasley:

  http://nwww.faqs.org/faqs/ai-faq/genetic/

- Otimização com múltiplos objetivos:

  http://www.lania.mx/~ccoello/EMOO/

- Sistema imunológico artificial:

  http://www.dca.fee.unicamp.br/~vonzuben/

- *Particle swarm optimization*:

  http://web.ics.purdue.edu/~hux/PSO.shtml
  http://www.particleswarm.net/

- Evolução diferencial:

  http://www.icsi.berkeley.edu/~storn/code.html#hist

- Problemas com restrições em AEs:

  http://www.coe.uncc.edu/~zbyszek/papers.html

- Aspectos de projeto evolutivo usando computadores

  http://lslwww.epfl.ch/pages/embryonics/thesis/home.html

5.7 *Software* (demonstrações e auxílio ao aprendizado)

- Demonstração de AEs para problemas com múltiplos objetivos

  http://userweb.elec.gla.ac.uk/y/yunli/ga_demo/
  http://www.tik.ee.ethz.ch/~zitzler/moea.html

- Evolução "artificial" (em Java)

  http://math.hws.edu/xJava/GA/

- Problema de escalonamento

  http://www.cs.bham.ac.uk/~wbl/four_node.demo.html

- Algoritmo genético simples

  http://www.ida.his.se/~bjorne/Java/SGA/sga.html

- Programação genética

  http://www.systemtechnik.tu-ilmenau.de/~pohlheim/EA_Java

- Vários:

  http://www.aridolan.com/ofiles/ga/gaa/gaa.html
  http://members.aol.com/anarchyxi/ants2.htm

5.8 *Software* (códigos fonte)

- Gallops 3.2.4 (C):

  http://garage.cps.msu.edu/software/galopps/index.html

- GAlib (C++):

  http://lancet.mit.edu/ga/

- jaga (Java):

  http://cs.felk.cvut.cz/~koutnij/studium/jaga/jaga.html

- *Simple generalized GA* (Perl):

  http://www.skamphausen.de/software/AI/ga.html

- Illigal (vários):

  http://www-illigal.ge.uiuc.edu/sourcecd.html

- GA MJMax (C++):

  http://www.michaelwax.com/geneticmain.html

- Vários em Matlab:

  http://www.systemtechnik.tu-ilmenau.de/~pohlheim/EA_Matlab
  http://www.mathtools.net/Java/Genetic_algorithms/MATLAB/

- Vários (*CMU Artificial Intelligence Repository*):

  http://www-2.cs.cmu.edu/afs/cs/project/ai-repository/ai/areas

- Outros:

  http://www.iitk.ac.in/kangal/soft.htm
  http://www.geneticprogramming.com/ga/GAsoftware.html

Capítulo 6

Conclusão e perspectivas

Este trabalho apresentou os alguns fundamentos, tendências e um espectro da aplicabilidade dos AEs em diversas áreas da matemátrica, computação, automação e sistemas produtivos. Quanto ao projeto e à configuração em sistemas eficientes e robustos de projeto, os AEs são empregados com sucesso devido as seguintes características: (i) tratarem adequadamente os sistemas sujeitos a restrições; (ii) não requererem as informações relativas a derivadas, estas usualmente necessárias em métodos convencionais de otimização; (iii) adequarem-se à implementação em paralelo e distribuídas; (iv) possibilitarem a utilização do conhecimento obtido a priori pelo projetista; e (v) tratarem com sistemas complexos e espaços de busca com múltiplas modas e/ou múltiplos objetivos.

Contudo, algumas limitações estão presentes nos AEs. Os algoritmos evolutivos tratam-se de métodos estocásticos e seu desempenho varia de execução para execução (a menos que o mesmo gerador de números aleatórios com a mesma semente é utilizado). Devido a isto, a média da convergência sobre diversas execuções do AE é um indicador de desempenho mais útil que uma simples execução.

Os AEs apresentam vantagens e desvantagens em relação aos métodos tradicionais e este aspecto serve para enfatizar a necessidade de não abandonar-se os métodos convencionais de otimização.

Entre as vantagens dos AEs têm-se: (i) não existe a necessidade de assumir-se características do espaço do problema, (ii) vastamente aplicável (algoritmos de propósito geral), (iii) baixo custo de desenvolvimento e aplicação, (iv) facilidade de incorporar outros métodos, (v) as soluções obtidas são interpretáveis (diferente de muitas configurações de redes neurais), e (vi) pode ser executado interativamente e possibilita a acomodação de soluções propostas pelo usuário no procedimento de otimização.

Entre as desvantagens dos AEs deve-se mencionar que: (i) não garante uma solução ótima, (ii) pode necessitar de sintonia de alguns parâmetros inerentes a metodologia evolutiva adotada, e (iii) frequentemente apresenta alto custo computacional.

De acordo com o teorema *no free lunch* (NFL) [171] não existe algum algoritmo

para resolução de todos problemas de otimização que é genericamente (em média) superior que algum outro algoritmo competidor. A questão se os AEs são inferiores ou superiores a algum método alternativo é insensata. O que pode ser afirmado somente é que AEs comportam-se melhor que outros métodos com respeito a resolução de uma classe específica de problemas, e como conseqüência que eles comportam-se pior para outras classes de problemas.

Bibliografia

[1] A. Agapie, Theoretical analysis of mutation-adaptive evolutionary algorithms, *Evolutionary Computation*, **9**(2), 127-146.

[2] H. E. Aguirre, K. Tanaka & T. Sugimura, Cooperative model for genetic operators to improve GAs. in "Proceedings of International Conference on Information Intelligence and Systems", Bethesda, MD, USA, pp. 98-106, 1999.

[3] J. T. Alander, An indexed bibliography of genetic algorithms in control, *Report 94-1*, Department of Information Technology and Production Economics, University of Vaasa, Vaasa, Finland, 1995.

[4] R. W. Anderson, Learning and evolution: A quantitative genetics approach, *Journal of Theoretical Biology*, **175**, 89-101, 1995.

[5] P. J. Angeline, Adaptive and self-adaptive evolutionary computations, *Computational Intelligence: A Dynamic Systems Perspective*, (Palniswami, M.; Attikiouzel, Y.; Marks, R.; D. Fogel & T. Fukuda (eds.)), Piscataway, NJ: IEEE Press, pp. 152-163, 1995.

[6] T. Bäck, D. B. Fogel & Z. Michalewicz, *Handbook of evolutionary computation.* Bristol, Philadelphia: Institute of Physics Publishing. New York, Oxford: Oxford University Press, 1997.

[7] T. Bäck, U. Hammel & H. -P. Schwefel, Evolutionary computation: comments on the history and current state, *IEEE Transactions on Evolutionary Computation*, **1**(1), 3-17, 1997.

[8] T. Bäck & F. Hoffmeister, Adaptive search by evolutionary algorithms. in "Models of Selforganization in Complex Systems", (W. Ebeling, M. Peschel & W. Weidlich (eds.)), v. 64, Mathematical Research. Berlin, Germany: Akademie-Verlag, pp. 156-163, 1991.

[9] T. Bäck, G. Rudolph & H. -P. Schwefel, Evolutionary programming and evolution strategies: similarities and differences. in "Proceedings of the 2nd Annual Conference on Evolutionary Programming", San Diego, CA, USA, pp. 11-22, 1993.

[10] T. Bäck & H. -P. Schwefel, An overview of evolutionary algorithms for parameter optimization. *Evolutionary Computation*, **1**(1), 1-23, 1993.

[11] J. M. Baldwin, A new factor in evolution, *American Naturalist*, **30**, 441-451, 1896.

[12] P. J. Bentley (ed.), Evolutionary design by computers, Morgan Kaufmann, San Francisco, CA, USA.

[13] H. G. Beyer, Toward a theory of evolution strategies: self-adaptation, *Evolutionary Computation*, **3**(3), 311-348, 1995.

[14] T. T. Binh & U. Korn, Multicriteria control system design using an intelligent evolution strategy. in "Proceedings of Conference for Control of Industrial Systems", Belfort, France, v. 2, pp. 242-247, 1997.

[15] T. Blickle & L. Thiele, A comparison of selection schemes used in genetic algoritmhs, *TIK-Report*, n. 11, version 2, Computer Engineering and Communication Network Lab, Swiss Federal Institute of Technology, Zurich, Switzerland,1996.

[16] P. P. Bonissone, Y.-T. Chen, K. Goebel & P. S. Khedkar, Hybrid soft computing systems: industrial and comercial applications, *Proceedings of the IEEE*, **87**(9), 1641-1667, 1999.

[17] E. K. Burke & A. J. Smith, Hybrid evolutionary techniques for the maintenance scheduling problem, *IEEE Transacitons on Power Systems*, **15**(1), 122-128.

[18] B. Carse, T. C. Fogarty & A. Munro, Evolving fuzzy rule based controllers using genetic algorithms, *Fuzzy Sets and Systems*, **80**, 273-293, 1996.

[19] D. R. Carvalho & A. A. Freitas, Um algoritmo imunológico para descobrir regras para pequenos disjuntos em *data mining*. in "Anais do II Congresso Brasileiro de Computação", Itajaí, SC, Brasil, 2002.

[20] L. N. Castro & F. J. Zuben, An evolutionary immune network for data clustering. in "Proceedings of Brazilian Symposium on Artificial Neural Networks", SBRN, Rio de Janeiro, Brazil, 2002.

[21] N. Chaiyaratana & A. M. Zalzala, Recent developments in evolutionay and genetic algorithms: theory and applications. in "Genetic Algorithms in Engineering Systems: Innovations and Applications", Glasgow, UK, pp. 270-277.

[22] K. Chellapilla, Combining mutation operators in evolutionary programming, *IEEE Transactions on Evolutionary Computation*, **2**(3), 91-96, 1998.

[23] S.-H. Chen & C.-C. Ni, Coevolutionary instability in games: an analysis based on genetic algorithms. in "Proceedings of the IEEE International Conference on Evolutionary Computation", Indianapolis, IN, USA, pp. 703-708, 1997.

[24] L. S. Coelho, Fundamentos e aplicação de algoritmos evolutivos no projeto de controladores avançados. in "Anais do III Seminário Nacional de Controle e Automação", Salvador, BA, 2003.

[25] L. S. Coelho & A. A. R. Coelho, Rede parcialmente recorrente de Elman e algoritmo genético híbrido em identificação experimental de um túnel de vento. in "Anais do III Congresso Brasileiro de Redes Neurais, Florianópolis, SC, pp. 296-301, 1997.

[26] L. S. Coelho & A. A. R. Coelho, Algoritmos evolutivos em identificação e controle de processos: uma visão integrada e perspectivas. *Revista SBA Controle & Automação*, **10**(1), 13-30, 1999.

[27] L. S. Coelho & V. C. Mariani, Evolução lamarckiana baseada em programação evolutiva e método de Hooke-Jeeves aplicada à otimização de um sistema nebuloso. in "Anais do VI Congresso Brasileiro de Redes Neurais", São Paulo, SP, 253-258, 2003.

[28] C. A. Coello Coello, A comprehensive survey of evolutionary-based multiobjective optimization techniques, *Knowledge and Information Systems: An International Journal*, **1**(3), 269-308, 1999.

[29] C. A. Coello Coello, Constraint-handling using an evolutionary multiobjective optimization technique, *Civil Engineering and Environmental Systems*, **17**, 319-346, 2000.

[30] C. A. Coello Coello, D. A. Van Veldhuizen & G. B. Lamont, *Evolutionary algorithms for solving multi-objective problems*, Kluwer Academic/Plenum Publishers, New York, NY, USA.

[31] A. Chipperfield & P. Fleming, Evolutionary algorithms for control engineering, in "Proceedings of the 13th IFAC World Congress", San Francisco, CA, USA, pp. 181-186, 1996.

[32] O. Cordón, F. Herrera & M. Lozano, A classified review on the combination fuzzy logic-genetic algorithms bibliography, *Technical Report DECSAI-95129*, Department of Computer Science and A.I., University of Granada, Granada, 1995.

[33] O. Cordón, F. Herrera & M. Lozano, On the combination of fuzzy logic and evolutionary computation: a short review and bibliography. in "Fuzzy evolutionary computation", (W. Pedrycz (ed.)), Kluwer Academic: Boston, USA, pp. 57-77, 1997.

[34] P. Cristea, A. Arsene & B. Nitulescu, Evolutionary intelligent agents. in "Proceedings of the Congress on Evolutionary Computation", v. 2, pp. 1320-1328.

[35] C. Darwin, *Origin of species by means of natural selection, or the preservation of favored races in the struggle for life*. 6th Edition, v. I and II. John Murray : London, Albemarle Street, 1859 (1a. ed), disponível em *http://honors.ccsu.ctstateu.edu/Honors/EText/Darwin/DarwinOriginContents.html*

[36] L. Davis, *Handbook of genetic algorithms*. Van Nostrand Reinhold: New York, NY, USA, 1991.

[37] R. Dawkins, *The selfish gene*. Oxford University Press: UK, 1989.

[38] K. Deb, Multi-objective genetic algorithms: problem difficulties and construction of test problems, *Evolutionary Computation*, **7**(3), 205-230.

[39] I. De Falco, R. Del Balio, A. Della Cioppa, E. Tarantino, A comparative analysis of evolutionary algorithms for function optimisation. in "2nd On-line Workshop on Evolutionary Computation", hosted in Internet, 1996.

[40] J. S. Dias, A. C. Zimmermann, P. S. Borges & J. M. Barreto, Aprendizado e evolução: de Lamarck a Baldwin, in "V Simpósio Brasileiro de Redes Neurais", Belo Horizonte, MG, 1998.

[41] C. Dimopoulos & A. M. S. Zalzala, Recent developments in evolutionary computation for manufacturing optimization: problems, solutions, and comparisons. *IEEE Transactions on Evolutionary Computation*, **4**(2), 93-113, 2000.

[42] M . Dorigo, V. Maniezzo & A. Colorni, Positive feedback as a search strategy, *Technical Report 91-016*, Dipartimento di Elettronica e Informazione, Politecnico di Milano, Italy, 1991.

[43] R. C. Eberhart & J. Kennedy, A new optimizer using particle swarm theory. in "Proceedings of the Sixth International Symposium on Micro Machine and Human Science", Nagoya, Japan, Piscataway, NJ: IEEE Service Center, pp. 39-43, 1995.

[44] M. Ehrgott & X. Gandibleux. A survey and annotated bibliography of multiobjective combinatorial optimization, *OR Spektrum*, 22, 425-460, 2000.

[45] Á. E. Eiben, K. Hinterding & Z. Michalewicz, Parameter control in evolutionary algorithms, *IEEE Transactions on Evolutionary Computation*, **3**(2), 124-141, 1999.

[46] EvoNews, *Newsletter of the EvoNet Network of Excellence in Evolutionary Computation*. European Commision's ESPRIT IV Programme, issue 1, 1996.

[47] EvoNews, *Newsletter of the EvoNet Network of Excellence in Evolutionary Computation*. European Commision's ESPRIT IV Programme, issue 9, Winter, 1999.

[48] U. Fayyad, G. Piatetsky-Shapiro, P. Smith & R. Uthurusamy, *Advances in knowledge discovery and data mining*, American Association for Artificial Intelligence. Menlo Park, CA: MIT Press, 1996.

[49] S. J. Flockton & M. S. White, Pole-zero system identification using genetic algorithms. in "Proceedings of the 5th International of the Conference on Genetic Algorithms", pp. 531-535, 1993.

[50] L. J. Fogel, A. J. Owens & M. J. Walsh, *Artificial intelligence through simulated evolution*, Wiley, 1966.

[51] D. B. Fogel, An introduction to simulated evolutionary optimization. *IEEE Transactions on Neural Networks*, **5**(1), 3-14, 1994.

[52] D. Fogel, *Evolutionary computation: toward a new philosophy of machine intelligence*. IEEE Press: Piscataway, NJ, USA, 1995.

[53] C. M. Fonseca & P. J. Fleming, An overview of evolutionary algorithms in multi-objective optimization, *Evolutionary Computation*, **3**(1), 1-16, 1995.

[54] C. M. Fonseca & P. J. Fleming, Multiobjective optimization and multiple constraint handling with evolutionary algorithms – part I: a unified formulation, *IEEE Transactions on Systems, Man, and Cybernetics — Part A: Systems and Humans*, **28**(1), 26-37, 1998.

[55] C. M. Fonseca & P. J. Fleming, Multiobjective optimization and multiple constraint handling with evolutionary algorithms – part II: a application example, *IEEE Transactions on Systems, Man, and Cybernetics — Part A: Systems and Humans*, **28**(1), 38-47, 1998.

[56] A. A. Freitas, A survey of evolutionary algorithms for data mining and knowledge discovery. Advances in Evolutionary Computation, A. Ghosh & S. Tsutsui (eds.), Springer-Verlag, 2001.

[57] L. M. Gabora, Meme and variations: a computational model of cultural evolution, in "Lectures in Complex Systems", Addison Wesley: Reading, MA, USA, 1995.

[58] F. Glover, Tabu search - part I, *ORSA Journal on Computing*, 1(3), 190-206, 1989.

[59] F. Glover, Tabu search - part II, *ORSA Journal on Computing*, 2(1), 4-32, 1989.

[60] F. Glover, Tabu search - a tutorial, *Interfaces*, **20**(4), 74-94, 1990.

[61] F. Glover & M. Laguna, Tabu search, Massachusetts, Kluwer Academic Publishers, 1997.

[62] M. C. Goldbarg & M. P. L. Luna, *Otimização combinatória e programação linear*, Editora Campus, Rio de Janeiro, RJ, 2000.

[63] D. E. Goldberg, *Genetic algorithms in search, optimization, and machine learning*. Addison Wesley: Reading, MA, USA, 1989.

[64] D. E. Goldberg & K. Deb, A comparison of selection schemes used in genetic algorithms. in "Foundations of Genetic Algorithms", Rawling, G. J. E. (ed.), Morgan Kaufmann: San Mateo, CA, USA. pp. 69-93, 1991.

[65] S. Goonatilake & S. Khebbal, *Intelligent hybrid systems*, Chichester: John Wiley & Sons, USA, 1995.

[66] G. J. Gray, Y. Li, D. J. Murray-Smith & K. C. Sharman, Structural system identification using genetic programming and a block diagram oriented simulation tool, *IEE Electronics Letters*, **32**(15), 1423-1424, 1996.

[67] H. Hagras, V. Callagham, M. Coolley & M. Carr-West, A fuzzy-genetic based embedded-agent approach to learning and control in agricultural autonomous vehicles. in "Proceedings of the IEEE International Conference on Robotics & Automation", Detroit, Michigan, USA, pp. 1005-1010, 1999.

[68] D. Hales, Selfish memes and selfless agents-altruism in the swap shop. in "Proceedings of the International Conference on Multi Agent Systems", Paris, France, pp. 431-432, 1998.

[69] K.-H. Han & J.-H. Kim, Quantum-inspired evolutionary algorithm for a class a combinatorial optimization, *IEEE Transactions on Evolutionary Computation*, **6**(6), 580-593.

[70] L. He, K.-J. Wang, H.-Z. Jin, G.-B. Li & X. Z. Gao, The combination and prospects of neural networks, fuzzy logic and genetic algorithms. in "Proceedings of the IEEE Midnight-Sun Workshop on Soft Computing Methods in Industrial Applications", SMCia'99, Kussano, Finland, pp. 52-57, 1999.

[71] J. Heitkoetter & D. Beasley, *The hitch-hitker's guide to evolutionary computation: a list of frequently asked questions* (FAQ), 1996. Available via anonymous FTP from *rtfm.mit.edu: /pub/usenet/news.answers/ai-faq/genetic.*

[72] M. A. Henson & D. E. Seborg (eds.), *Nonlinear process control*, Prentice Hall PTR: Upper Saddle River, NJ, USA, 1997.

[73] G. E. Hinton & S. J. Nowlan, How learning can guide evolution, *Complex Systems*, **1**, 495-502, 1987.

[74] C. W. Ho, K. H. Lee & K. S. Leung, A genetic algorithm based on mutation and crossover with adaptive probabilities. in "Proceedings of IEEE Intenraitonal Conference on Evolutionary Computation", Washington, DC, USA, pp. 768-775.

[75] C. Hocaoğlu & A. C. Sanderson, Planning multiple paths with evolutionary speciation, *IEEE Transactions on Evolutionary Computation*, **5**(3), 169-190.

[76] J. H. Holland, *Adaptation in natural and artificial systems*. University of Michigan Press (Reprinted in 1992 by MIT Press), 1975.

[77] L. M. Howard & D. J. D'Angelo, The GA-P: a genetic algorithm and genetic programming hybrid, *IEEE Expert*, **10**(3), 11-15, 1995.

[78] J. Hunt, C. King & D. Cooke, Immunizing against fraud. in "IEEE Colloquium on Knowledge Discovery and Data Mining", London, Digest n. 96/198,1996.

[79] L. Ingber, Very fast simulated re-annealing, *Mathematical Computer Modelling*, **12**, 967-973, 1989.

[80] M. Jamshidi, R. A. Krohling, L. S. Coelho & P. Fleming, *Robust control systems with genetic algorithms*, CRC Press, Boca Raton, FL, USA, 2002.

[81] T. Jiang & F. Yang, An evolutionary tabu search for cell image segmentation, *IEEE Transactions on Systems, Man and Cybernetics — Part B*, 32(5), 675 -678.

[82] J. Jones & C. Houck, On the use of non-stationary penalty functions to solve constrained optimization problems with genetic algorithms. in "Proceedings of IEEE International Symposium on Evolutionary Computation", Orlando, FL, USA, pp. 579-584, 1994.

[83] A. H. Jones, L. Y-. Chih, P. B. D. M. Oliveira & S. B. Kenway,. Auto-tuning of dual mode controllers using genetic algorithms. in "Proceedings of IEE/IEEE GALESIA", Glasgow, UK, pp. 516-521, 1997.

[84] J. Kennedy & R. C. Eberhart, Particle swarm optimization. in "Proceedings of the IEEE International Conference on Neural Networks", Piscataway, NJ: IEEE Service Center, pp. 1942-1948, 1995.

[85] S. A. Kennedy, Five ways to a smarter genetic algorithm. *AI Expert*, December, 35-38, 1993.

[86] R. Khosla & T. Dillon, *Engineering intelligent hybrid multi-agent systems*. Kluwer Academic Publishers: Boston, USA, 1997.

[87] H. Kim, Y. Hayashi & K. Nara, An algorithm for thermal unit maintenance scheduling through combined use of GA, SA and TS, *IEEE Transactions on Power Systems*, **12**(1), 329-335.

[88] B. M. Kim, Y. B. Kim & C. H. Oh, A study on the convergence of genetic algorithms, *Computers on Industrial Engineering*, **33**(3-4), 581-588, 1997.

[89] S. H. Kim, C. Park & F. Harashima, A self-organized fuzzy controller for wheeled mobile robot using an evolutionary algorithm, *IEEE Transactions on Industrial Electronics*, **48**(2), 467-474, 2001.

[90] S. Kirkpatrick, C. D. Gelatt & M. P. Vecchi, Optimization by simulated annealing, *Science*, **220**, 45-54, 1983.

[91] J. R. Koza, *Genetic programming: on the programming of computers by means of natural selection*. MIT Press: Cambridge, MA, 1992.

[92] J. R. Koza, *Genetic programming II: automatic discovery of reusable programs*. MIT Press: Cambridge, MA, 1994.

[93] T. Krink, R. K. Ursem & R. Thomsen, Ant systems and particle swarm optimization, *EVALife Course*, Faal 2002, Topics of Evolutionary Computation, ALife Group, Dept. of Computer Science, University of Aarhus EV, Denmark, 2002.

[94] K. Kristinsson & G. A. Dumont, System identification and control using genetic algorithms, *IEEE Transactions on Systems, Man and Cybernetics*, **22**(5),1033-1046, 1992.

[95] R. A. Krohling, L. S. Coelho & Y. Shi, Cooperative particle swarm optimization for robust control system design. in "7th On-line World Conference on Soft Computing in Industrial Applications", Granada, Spain, *http://site: decsai.ugr.es/wsc7*, 2002.

[96] M. Kubo & Y. Kakazu, Simulating a competition for foods between ant colonies as a coordinated model of autonomous agents. in "Proceedings of International Conference on Systems, Man, and Cybernetics", La Touquet, France, v. 5, pp. 142-148, 1993.

[97] J. Lieslehto, PID controller tuning using evolutionary programming. in "Proceedings of the American Control Conference", vol. 5, pp. 2828-2833, 2001.

[98] D. A. Linkens & H. O. Nyongesa, Learning systems in intelligent control: an appraisal of fuzzy, neural and genetic algorithm control applications, *IEE Proceedings Control Theory and Applications*, **143**(4), 367-386, 1996.

[99] J. Lis & A. E. Eiben, A multi-sexual genetic algorithm for multiobjective optimization. in "Proceedings of IEEE International Conference on Evolutionary Computation", Nagoya, Japan, pp. 59-64, 1996.

[100] M. Littman, Limulations combining evolution and learning. In "Adaptive Individuals in Evolving Populations: Models and Algorithms", (R. K. Belew & M. Mitchell (eds.)), Massachusetts: Addison-Wesley, 465-477, 1996.

[101] S. G. B. C. Lopes, *Bio — volume 3 — genética, evolução, ecologia*, Editora Saraiva, São Paulo, SP, 2002.

[102] M. Mahfouf, D. A. Linkens & M. F. Abbod, Multi-objective genetic optimisation of GPC and SOFLC tuning parameters using a fuzzy-based ranking method, *IEE Proceedings-Control Theory Appl.*, **147**(3), 344-354, 2000.

[103] K. F. Man, K. S. Tang & S. Kwong, Genetic algorithms: concepts and applications, *IEEE Transactions on Industrial Electronics*, **43**(5), 519-534, 1996.

[104] V. Maniezzo & A. Colorni, The ant system applied to the quadratic assignment problem, *IEEE Transactions on Knowledge and Data Engineering*, **11**(5), 769-778, 1999.

[105] A. H. Mantawy, Y. L. Abdel-Magid & S. Z. Selim, Integrating genetic algorithms, tabu search, and simulated annealing for the unit commitment problem, *IEEE Transactions on Power Systems*, **14**(3), 829-836, 1999.

[106] B. McKay, M. J. Willis, G. A. Montague & G. Barton, Using genetic programming to develop inferential estimation algorithms. in "Proceedings of 1st International Conference on Genetic Programming", San Francisco, CA, USA, pp. 157-165, 1996.

[107] G. McNutt, Using co-evolution to produce robust robot control. in "Proceedings of the 36th Conference on Decision & Control", San Diego, CA, USA, pp. 2515-2520, 1997.

[108] P. Merz & B. Freisleben, Fitness landscape analysis and memetic algorithms for the quadratic assignment problem, *IEEE Transactions on Evolutionary Computation*, **4**(4), 337-352, 2000.

[109] L. Medsker & D. Bailey, Models and guidelines for integrating expert systems and neural networks. in "Hybrid architectures for intelligent systems", (A. Kandel & G. Langholz (eds.)), CRC Press, Boca Raton, FL, USA, pp. 154-171, 1992.

[110] Z. Michalewicz, *Genetic algorithms + data structures = evolution programs*. Springer-Verlag, Berlin: Germany, 1992.

[111] C. L. Morgan, On modification and variation, *Science*, **4**, 733-740, 1896.

[112] P. Moscato & M. G. Norman,. A 'memetic' approach for the traveling salesman problem — implementation of computational ecology for combinatorial optimisation on message-passing systems. in "International Conference on Parallel Computing and Transputer Applications", IOS Press, 1992.

[113] F. Müller, Heurísticas e metaheurísticas. in "Anais da V Escola Regional de Informática", Santa Maria, RS, Brasil, (M. A. Cândia & R. C. Nunes (eds.), 19-40, 1997.

[114] J. L. Navarro & P. Albertos, Fuzzy logic implementation of industrial controllers. in "Proceedings of the 13th IFAC World Congress", San Francisco, CA, USA, pp. 409-414, 1996.

[115] J. A. Nelder & .R Mead, A simplex method for function minimisation, *Computer Journal*, **7**, 308-313, 1965.

[116] A. Neves, A. Silva & E. Costa, Evolutionary path planning for nonholonomic robots. in "Genetic and Evolutionary Computation Conference", Orlando, FL, USA, pp. 466-472, 1999.

[117] S. Nolfi, O. Miglino & D. Parisi, Phenotypic plasticity in evolving neural networks, in "Proceedings of the International Conference from Perception to Evolution of Artificial Neural Networks (D. P. Gaussier & J-D. Nicoud (eds.)), Los Alamitos, CA: IEEE Press, pp. 146-157, 1994.

[118] M. Oliveira, J. Barreiros, E. Costa & F. Pereira, LamBaDa: an artificial environment to study the interaction between evolution and learning. in "Proceedings of IEEE International Conference on Evolutionary Computation", Washington, DC, v. 1, pp. 145-152, 1999.

[119] H. F. Osborn, Ontogenic and phylogenic variation. *Science* , **4**, 786-789, 1896.

[120] A. Ostermeier, A. Gawelczyk & N. Hansen, Step-size adaptation based on non-local use of selection information. in "Proceedings of Parallel Problem Solving from Nature", PPSN III, Berlin: Springer, Jerusalem, Israel, pp. 189-198, 1994.

[121] S. J. Ovaska, H. F. VanLandigham & A. Kamiya, Fusion of soft computing and hard computing in indutrial applications: an overview, *IEEE Transactions on Systems, Man, and Cybenetics — Part C: Applications and Reviews*, **32**(2), 72-79, 2002.

[122] D. Parisi, S. Nolfi & F. Cecconi, Learning, behavior and evolution. in "Proceedings of the First European Conference on Artificial Life", MIT Press/Bradford Books, Cambridge, MA, pp. 207-216, 1992.

[123] G. B. Parker, The incremental evolution of gaits for hexapod robots. in "Genetic and Evolutionary Computation Conference", San Francisco, CA, USA, pp. 1114-1121, 2001.

[124] I. C. Parmee, Exploring the design potential of evolutionary/adaptive search and other computational intelligence technologies. in "Adaptive Computing in Design and Manufacture", (I. C. Parmee (ed.)), pp. 27-44, Springer-Verlag, London, UK.

[125] I. C. Parmee, Exploring the design potential of evolutionary search, exploration and optimisation. in "Evolutionary design by computers", (P. Bentley (ed.)), Academic Press, London, UK.

[126] D. T. Pham & G. Jin, Genetic algorithm using gradient-like reproduction operator, *IEE Electronics Letters*, 31(18), 1558-1559, 1995.

[127] R. Poli & W. B. Langdon, Genetic programming with one-point crossover. in "Soft Computing in Engineering Design and Manufacturing", (P. K. Chawdhry, R. Roy & R. Pant (eds.)), Springer-Verlag: London, UK. pp. 180-189, 1998.

[128] W. H. Press, S. A. Teukolsky, W. T. Vetterling & B. P. Flannery, *Numerical recipes in c: the art of scientific computing*, 2nd ed., Cambridge Press, 1994.

[129] S. S. Rao, *Engineering optimization — theory and practice*, Wiley-Interscience, USA, 1996.

[130] W. Rauch & P. Harremoës, Genetic algorithms in real time control applied to minimize transient pollution from urban wastewater systems, *Water Res.*, **33**(5), 1265-1277, 1999.

[131] I. Rechenberg, *Evolutionsstragie: optimierung technicher systeme nach prinzipien der biologischen evolution*. Frommann-Holzboog, Stuttgart, Germany, 1973.

[132] J -M. Renders & S. P. Flasse, Hybrid methods using genetic algorithms for global optimization, *IEEE Transactions on Systems, Man, and Cybernetics — Part B: Cybernetics*, **26**(2), 243-258, 1996.

[133] F. Robin, A. Orzati, E. Moreno, O. J. Homan & W. Bachtold, Simulation and evolutionary optimization of electron-beam lithography with genetic and simplex-downhill algorithms, *IEEE Transactions on Evolutionary Computation*, **7**(1), 69-82, 2003.

[134] R. Roy, A. Tiwari and A. Braneby. Making evolutionary design optimisation popular in industry: issues and techniques. in "Proceedings of 6th Online World Conference on Soft Computing in Industrial Applications", (R. Roy, M. Köppen, S. Ovaska, T. Furuhashi (eds.)), *Soft computing and industry: recent applications*, London, UK, 2002.

[135] R. Roy, A. Tiwari, O. Munaux & G. Jared, Real-life engineering design optimisation: features and techniques. in "Proceedings of the 5th Online World Conference on Soft Computing in Industrial Applications", WSC5, IEEE, Finland.

[136] G. Rudolph, On correlated mutations in evolution strategies. in "Proceedings of the 2nd International Conference on Parallel Solving from Nature", (R. Männer & B. Manderick (eds.)), Brussels, Belgium, pp. 105-114, 1992.

[137] M. Salim & X. Yao, Evolving SQL queries for data mining. in "Proceedings of the 3rd International Conference on Intelligent Data Engineering and Automated Learning", IDEAL'02, Lecture Notes in Computer Science, v. 2412, Springer, pp.62-67, 2002.

[138] N. Saravanan & D. B. Fogel, An empirical comparison of methods for correlated mutations under self-adaptation. in "Proceedings of 5th Annual Conference on Evolutionary Programming", MIT Press, Cambridge, Massachusetts, London, England, pp. 479-485, 1996.

[139] T. Sato & M. Hagiwara, Bee system: finding solution by a concentrated search. in "Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics", Orlando, FL, USA, v. 4, pp. 3954-3959, 1997.

[140] J. D. Schaffer, Combinations of genetic algorithms with neural networks or fuzzy systems. in "Computational intelligence: imitating life", (J. M. Zurada, R. J. Marks II & C. J. Robinson (eds.), IEEE Press: Piscataway, NJ, pp. 371-382, 1994.

[141] V. Schnecke & O. Vornberger, Hybrid genetic algorithms for constrained placement problems, *IEEE Transactions on Evolutionary Computation*, **1**(4), 266-277, 1997.

[142] W. Schiffmann, M. Joost & R. Werner, Synthesis and performance analysis of multilayer neural network architectures, *Technical Report 16/1992*, University of Koblenz, Institute für Physics, 1992.

[143] H. -P. Schwefel, *Numerical optimization of computer models*. John Wiley & Sons: NY, USA, 1981.

[144] M. Sefrioui & J. Perlaux, Nash genetic algorithms: examples and applications. in "Proceedings of the Congress on Evolutionary Computation", La Jolla, CA, pp. 509-516, 2000.

[145] Y.-G. Seo, S.-B. Cho & X. Yao, The impact of payoff function and local interaction on the N-player iterated prisoner's dilemma, *Knowledge and Information Systems: An International Journal,* **2**(4), 461-478, 2000.

[146] Y. Shi & R. C. Eberhart, A modified particle swarm optimizer. in "Proceedings of the IEEE International Conference on Evolutionary Computation", Piscataway, NJ: IEEE Press, pp. 69-73, 1998.

[147] Y. Shi & R. C. Eberhart, Empirical study of particle swarm optimization. in "Proceedings of the Congress on Evolutionary Computation", Piscataway, NJ: IEEE Service Center, pp. 1945-1950, 1999.

[148] M. Shim, S. Seoung, B. Ko & M. So, Application of evolutionary computations at LG Eletronica. in "Proceedings of the IEEE International Fuzzy Systems Conference", v. 3, Seoul, Korea, pp. 1802-1806, 1999.

[149] S. Smith, The simplex method and evolutionary algorithms. in "Proceedings of the IEEE World Congress on Computational Intelligence", Anchrage, AL, USA, 709-804, 1998.

[150] L. N. C. Silva, Engenharia imunológica: desenvolvimento e aplicação de ferramentas computacionais inspiradas em sistemas imunológicos artificiais, *Tese de doutorado*, Faculdade de Engenharia Elétrica e de Computação, UNICAMP, Campinas, SP, 2001.

[151] A. Somayaji, S. Hofmeyr & S. Forrest, Principles of a computer immune system, Submitted to New Security Paradigms, 1997, *http://www.cs.unm.edu/~forrest/papers.html* (27/7/1999).

[152] B. Soucek & The Iris Group (eds.), *Dynamic, genetic, and chaotic programming: the sixth generation*. John Wiley & Sons: New York, NY, USA, 1992.

[153] M. Srinivas & L. M. Patnaik, Genetic algorithms: a survey, *IEEE Computer*, **27**(6), 17-26, 1994.

[154] M. Srinivas & L. M. Patnaik, Adaptive probabilities of crossover and mutation in genetic algorithms, *IEEE Transactions on Systems, Man, and Cybernetics*, **24**(4), 656-667, 1994.

[155] T. Stützle & M. Dorigo, ACO algorithm for the quadratic assignment problem. in "New ideas in optimization", (D. Corne, M. Dorigo & F. Glover (eds.)), McGraw-Hill, 1999.

[156] R. -L. Sun, Evolving population-based search algorithms through thermodynamic operation: dynamic system design and integration, *PhD. thesis*, Institute for Systems Research, University of Maryland, USA, 1995.

[157] P. D. Surry & N. J. Radcliffe, The COMOGA method: constrained optimisation by multiobjective genetic algorithms, *Control and Cybernetics*, **26**(3), 391–412, 1997.

[158] R. Storn & K. Price, Differential evolution: a simple and efficient adaptive scheme for global optimization over continuos spaces, *Technical Report TR-95-012*, International Computer Science Institute, Berkeley, 1995.

[159] R. Subbu, C. Hocaoglu & A. C. Sanderson, A virtual design environment using evolutionary agents. in "Proceedings of International Conference on Robotics and Automation", v. 1, pp. 247-253.

[160] H. Takagi, Interactive evolutionary computation: fusion of the capabilities of EC optimization and human evaluation, *Proceedings of the IEEE*, **89**(9), 1275-1296, 2001.

[161] D. Thierens, Adaptive mutation rate control schemes in genetic algorithms. in "Proceedings of IEEE International Conference on Evolutionary Computation", Honolulu, HI, USA, pp. 980-985.

[162] M. Tomassini, A survey of genetic algorithms. in "Annual Reviews of Computational Physics", v. III, World Scientific, pp. 87-117, 1995.

[163] A. Tuson & P. Ross, Adapting operator settings in genetic algorithms, *Evolutionary Computation*, **6**(2), 161-184.

[164] J. A. Vasconcelos, R. R. Saldanha, L. Krähenbühl & A. Nicolas, Genetic algorithm coupled with a deterministic method for optimization in eletromagnetics, *IEEE Transactions on Magnetics*, **33**(3), 1860-1863, 1999.

[165] R. Viennet, C. Fontiex & I. Marc, Multicriteria optimization using a genetic algorithm for determining a Pareto set, *Journal of Systems Science*, **27**(2), 255-260, 1996.

[166] H. -M. Voigt & J. M. Lange, Local evolutionary search enhacement by random memorizing. in "Proceedings of IEEE World Congress on Computational Intelligence", Anchorage, AL, USA, 547-552, 1998.

[167] R. A. Watson & S. G. Ficici, Embodied evolution: embodying an evolutionary algorithm in a population of robots. in "Genetic and Evolutionary Computation Conference", Orlando, FL, USA, pp. 335-342, 1999.

[168] A. Weismann, *The germ-plasm: a theory of heredity*, New York, NY, USA, Scribners, 1893.

[169] D. Whitley, V. S. Gordon & K. Mathias, Lamarck evolution, The Baldwin effect and function optimization. in "Parallel problem solving from nature", Lecture Notes in Computer Science, v. 866, Springer-Verlag: Berlin, pp. 6-15, 1994.

[170] R. P. Wiegand, W. C. Liles & K. A. De Jong, An empirical analysis of collaboration methods in cooperative coevolutionary algorithms. in "Genetic and Evolutionary Computation Conference", San Francisco, CA, USA, 1235-1242, 2001.

[171] D. H. Wolpert & W. G. Macready, No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, **1**(1), 67-82, 1997.

[172] S. Yang, Adaptive non-uniform crossover based on statistics for genetic algorithms. in "Proceedings of the Genetic and Evolutionary Computation Conference", New York, NY, USA. pp. 650-657.

[173] X. Yao, Evolving artificial neural network, *Proceedings of the IEEE*, **87**(9), 1423-1447, 1999.

[174] X. Yao & Y. Liu, Fast evolutionary programming. in "Proceedings of the 5th Annual Conference on Evolutionary Programming", San Diego, CA, USA, The MIT Press, pp. 451-460, 1996.

[175] J. Yen, J. C. Liao, B. Lee & D. Randolph, A hybrid approach to modeling metabolic systems using a genetic algorithm and simplex method, *IEEE Transactions on Systems, Man and Cybernetics — Part B*, **28**(2), 173 -191, 1998.

[176] L. A. Zadeh, Fuzzy sets, *Information and Control*, **8**, 338-353, 1965.

[177] L. A. Zadeh, Outline of a new approach to the analysis of complex systems and decision processes, *IEEE Transactions on Systems, Man, and Cybernetics*, **3**, 28-44, 1973.

[178] R. S. Zebulum, M. A. C. Pacheco & M. M. B. R. Vellasco, *Evolutionary electronics: automatic design of electronic circuits and systems by genetic algorithms*, CRC Press, Boca Raton, FL, USA.

Índice remissivo