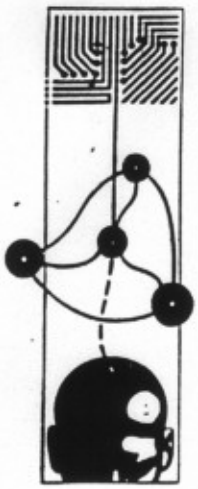

II CONGRESSO BRASILEIRO DE REDES NEURAIS
III ESCOLA DE REDES NEURAIS
CURITIBA, 29/OUT A 01/NOV/1995

Motivação, Fundamentos e Aplicações de Algoritmos Genéticos

Julio Tanomaru[†]

Universidade de Tokushima
Tokushima 770 Japão
e-mail: tanomaru@is.tokushima-u.ac.jp

Darwin frequently understood things far more clearly than both his supporters and his opponents, including those of the present day.

Ernst Mayr [102]

1 Introdução

As pesquisas sobre *modelos computacionais inteligentes* têm, nos últimos anos, se caracterizado pela tendência em buscar inspiração na natureza, onde existe um sem-número de exemplos vivos de processos que podem ser ditos "inteligentes". Para cientistas de computação, matemáticos e engenheiros, muitas das soluções que a mãe-natureza encontrou para complexos problemas de adaptação fornecem modelos interessantíssimos. Embora não se possa afirmar que tais soluções sejam todas *ótimas*, não há a menor dúvida de que os processos naturais, em particular os relacionados diretamente com os seres vivos, sejams soberbamente bem concebidos e adequados ao nosso mundo.

Físicos, biólogos e outros cientistas tentam desvendar os princípios que regem os fenômenos da natureza, enquanto que os matemáticos, cientistas de computação e engenheiros buscam idéias que possam ser copiadas ou pelo menos imitadas, e então aplicadas a problemas que a ciência atual ainda não consegue resolver satisfatoriamente. Por exemplo, o fenómeno de anelamento em metais, a enorme capacidade de processamento do sistema nervoso central de animais superiores obtida através da ação paralela de uma complexa rede de neurônios lentos e ruidosos, e o processo estocástico de evolução natural, forneceram inspiração para os paradigmas de computação denominados *Anelamento Simulado* [1, 88], *Redes Neurais* (ou *Neuronais*) *Artificiais* [6, 70], e *Computação Evolucionária* (EC), respectivamente.

Em termos computacionais, CE incorpora uma das metáforas mais apelativas: CE encara a teoria de evolução Darwiniana como um processo adaptativo de otimização, sugerindo um modelo em que populações de estruturas computacionais evoluem de modo a melhorar, em média, o desempenho geral da população com respeito a um dado problema, ou seja, a "adequabilidade" da população com respeito ao ambiente.

Atualmente, CE engloba um número crescente de paradigmas e métodos, dos quais os mais importantes são os Algoritmos Genéticos (AGs) [33, 36, 37, 56, 75, 78, 141], Programação Evolucionária (PE) [45], Estratégias Evolucionárias (EEs) [123, 124], Programação Genética (PG) [90, 91], e Sistemas Classificadores (SCs) [14, 23, 76, 156], entre outros. De fato, SCs and PG podem ser vistas como aplicações especiais de AGs, enquanto que PE e EEs foram desenvolvidas independentemente dos AGs, e somente agora começam a interagir. De modo geral, técnicas em CE dão origem aos chamados Algoritmos Evolucionários (AEs).

Algoritmos Genéticos são as mais difundidas e estudadas técnicas de CE, pela sua flexibilidade, relativa simplicidade de implementação, e eficácia em realizar busca global em ambientes adversos. Este artigo introduz as motivações e conceitos fundamentais da teoria dos Algoritmos Genéticos (AGs ou "GAs", a abreviatura em inglês), bem como um número de aspectos avançados que visam a aumentar sua eficiência. Exemplos de aplicações representativas, bem como aspectos de implementação prática são também abordados.

[†]Endereço: Department of Information Science and Intelligent Systems, Faculty of Engineering, The University of Tokushima, 2-1 Minami-josanjima, Tokushima, 770, Japan. Phone: +81-886-567488, FAX: +81-886-567489.

2 Algoritmos Genéticos: Motivação

2.1 Busca e Otimização

Uma grande parte dos problemas científicos pode ser formulada como problemas de busca e otimização: basicamente, existe uma série de fatores influenciando o desempenho de um dado sistema, e tais fatores podem assumir um número limitado ou ilimitado de valores, e podem ser sujeitos a certas restrições. O objetivo é encontrar a melhor combinação dos fatores, ou seja, a combinação de fatores que proporcione o melhor desempenho possível para o sistema em questão. Em termos técnicos, o conjunto de todas as combinações possíveis para os fatores constitui o chamado *espaço de busca*. Não é difícil perceber que existe uma dualidade entre os conceitos de busca e otimização, de tal modo que todo problema de busca pode ser considerado um problema de otimização e vice-versa.

Sem perda de generalidade, consideremos somente problemas de maximização[1]. Dada uma função $f : \Re^n \rightarrow \Re$ e um espaço de busca $S \subseteq \Re^n$, o problema de otimização pode ser formulado assim:

$$\boxed{\text{maximizar } f(x) \mid x \in S.} \quad (1)$$

Em termos de um problema de busca, o mesmo problema pode ser escrito:

$$\boxed{\text{encontrar } x^* \mid f(x^*) \geq f(x),\ \forall x \in S.} \quad (2)$$

A função $f(\cdot)$ pode ser derivável ou não, uni- ou multidimensional, e o espaço de busca S pode ser contínuo ou discreto, finito ou infinito, côncavo ou convexo. O problema é chamado *unimodal* quando há somente um ponto máximo x^* no espaço de busca, ou *multimodal* em caso contrário. Além disso, a função $f(\cdot)$ pode ser estacionária ou não-estacionária, isto é, variável com o tempo.

2.2 Métodos de Otimização

Numa classificação grosseira, há essencialmente três correntes de métodos gerais de otimização: métodos probabilísticos, numéricos e enumerativos. Há ainda um grande número de métodos híbridos.

Em primeiro lugar, deixemos os métodos aleatórios de lado e comecemos pelos métodos numéricos, que podem ser divididos em analíticos ou baseados em cálculo numérico. Os métodos analíticos de otimização são aplicáveis basicamente quando a função $f(\cdot)$ é explicitamente conhecida e derivável ou pode ser aproximada por alguma função derivável até o grau desejado de precisão. Neste caso, basta resolver as equações que resultam de igualar as derivadas da função a zero dentro do espaço de busca.

Já em relação aos métodos baseados em cálculo numérico, quando os espaço de busca é linear e, portanto, convexo, técnicas de Pesquisa Operacional como o método *simplex*, são suficientes [72, 86, 139]. Já em ambientes não-lineares, técnicas de gradiente ou de estatística de alta ordem são geralmente empregadas [69, 80, 97, 100]. Em métodos de gradiente, a idéia essencial é tomar um ponto partida e calcular o ponto seguinte com base no gradiente local, da função a ser otimizada [97]. Métodos de Newton requerem ainda as segundas derivadas parciais. Métodos de gradiente conjugado [69], quase-Newtonianos, e estratégias de métrica variável [119] aproximam a informação das derivadas iterativamente.

Embora sejam comumente empregados, tais métodos somente são capazes de encontrar ótimos locais, e são ineficazes para otimização de funções multimodais, como a ilustrada na Fig. 1. Além disso, em problemas de otimização combinatória em espaços discretos, geralmente a função a ser otimizada é não somente multimodal, mas também não-diferenciável e/ou não-contínua.

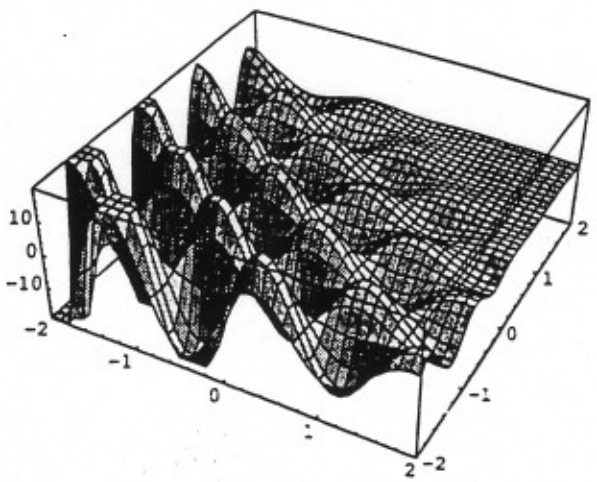

Figura 1: Exemplo de uma função multimodal.

Métodos enumerativos de otimização examinam cada ponto do espaço de busca, um por um, em busca dos pontos ótimos. A idéia pode ser intuitiva, mas é obviamente imprática quando há um número infinito ou extremamente grande de pontos a examinar.

Por outro lado, os métodos probabilísticos são métodos que empregam a idéia de busca probabilística, o que não quer dizer que sejam métodos totalmente baseados em sorte, como é o caso dos chamados métodos aleatórios. Por exemplo, a técnica conhecida como anelamento simulado que, como o nome indica, simula o fenômeno de

[1] Problemas de maximização podem ser convertidos em problemas de minimização e vice-versa, através de artifícios bem simples como, por exemplo, multiplicar a função por -1.

anelamento, ou seja, o resfriamento por etapas de ligas metálicas derretidas, usa o conceito de transições probabilísticas. A idéia é que, a altas temperaturas, as moléculas se comportam de modo aleatório, transitando tanto para estados de maior como de menor energia; à medida que a temperatura abaixa, porém, a probabilidade das moléculas saltarem temporariamente para estados de maior energia diminui. Essa técnica tem sido empregada com sucesso para a otimização global de funções multimodais, mas ainda apresenta muitos problemas, como o excessivamente longo tempo de processamento, difícil escolha de parâmetros e estratégias de resfriamento, etc.

2.3 Otimização na Natureza e AGs

Vista de forma global, a evolução natural implementa mecanismos adaptativos de otimização que, embora estejam longe de serem uma forma de busca aleatória, com certeza envolvem aleatoriedade. É esse tipo de busca *inteligente*, mas não determinística, que os AGs tentam imitar.

Embora Charles Darwin tenha formulado a Teoria da Evolução no final do século passado [29], foi só recentemente que se tentou idealizar um modelo matemático do processo evolutivo. Nos anos 60, John Holland, da Univ. de Michigan, começou a definir as bases de algoritmos de otimização de inspiração genética. Seu trabalho culminou na publicação do livro *Adaptação em Sistemas Naturais e Artificiais* em 1975 [75]. O livro, que é hoje muito citado mas pouquíssimo lido, foi pouco divulgado na época, em grande parte devido ao estilo pesado, com notação insossa e excessivamente complexa. Felizmente Holland e seus muitos discípulos, quase todos seus alunos de pós-graduação, continuaram sua linha de investigação, publicando resultados com alguma timidez mas com perseverança.

A grande popularidade que os AGs atingiram recentemente se deve a dois importantes fatores: a publicação de um livro tutorial sobre AGs [56] por um dos alunos de doutorado de Holland, David Goldberg, um pesquisador extremamente ativo e com excelente potencial didático, e às Conferências Internacionais sobre Algoritmos Genéticos (ICGAs), que têm se realizado a cada dois anos nos Estados Unidos desde 1985.

Os AGs pertencem à classe dos métodos probabilísticos de busca e otimização, embora não sejam aleatórios. Usa-se o conceito de probabilidade, mas AGs não são estúpidas buscas aleatórias. Pelo contrário, AGs tentam dirigir a busca para regiões do espaço onde é "provável" que os pontos ótimos estejam.

3 Fundamentos dos AGs

3.1 Uma Definição

Sem afirmar que esta é a melhor possível, aqui arriscamos a seguinte definição para AGs:

> *Algoritmos Genéticos (AGs) são métodos computacionais de busca baseados nos mecanismos de evolução natural e na genética. Em AGs, uma população de possíveis soluções para o problema em questão evolui de acordo com operadores probabilísticos concebidos a partir de metáforas biológicas, de modo que há uma tendência de que, na média, os indivíduos representem soluções cada vez melhores à medida que o processo evolutivo continua.*

3.2 Características Primárias

De modo geral, AGs têm as seguintes características [56, 75, 103, 141]:

- AGs operam numa população(conjunto) de pontos, e não a partir de um ponto isolado.
- AGs operam num espaço de soluções codificadas, e não no espaço de busca diretamente.
- AGs necessitam somente de informação sobre o valor de uma função objetivo para cada membro da população, e não requerem derivadas ou qualquer outro tipo de conhecimento.
- AGs usam transições probabilísticas, e não regras determinísticas.

3.3 Representação Cromossômica

O primeiro passo para aplicação de AGs a um problema qualquer é representar cada possível solução x no espaço de busca como uma seqüência de símbolos s gerados a partir de um dado alfabeto finito A. No caso mais simples, usa-se o alfabeto binário $A = \{0, 1\}$, mas no caso geral tanto o método de representação quanto o alfabeto genético dependem de cada problema.

Usando algumas das metáforas extremamente simplistas, mas empregadas pelos teóricos e praticantes de AGs com freqüência, cada seqüência s corresponde a um *cromossomo*, e cada elemento de s é equivalente a um *gene*. Como cada gene pode assumir qualquer valor do alfabeto A, cada elemento de A é equivalente a um *alelo*, ou seja, um valor possível para um dado gene. A posição de um gene num cromossomo, ou seja, o índice dentro da seqüência, corresponde a um *locus gênico*.

Além disso, na maior parte dos AGs assume-se que cada indivíduo seja constituído de um único cromossomo, razão pela qual é comum usarem-se os termos *indivíduo* e *cromossomo* indistintamente em trabalhos científicos e livros-textos. A grande maioria dos AGs propostos na literatura usam uma *população* de número fixo de indivíduos, com cromossomos também de tamanho constante.

3.4 Fluxo Básico

Tendo definido a representação cromossômica para o problema, gera-se um conjunto de possíveis soluções, chamadas de *soluções-candidatas*. Um conjunto de soluções codificadas de acordo com a representação selecionada corresponde a uma *população* de indivíduos, $P(0)$. AGs são algoritmos iterativos, e a cada iteração a população é modificada. Cada iteração de um AG é denominada uma *geração*, embora nem todos os indivíduos de uma população sejam necessariamente "filhos" de indivíduos da população na iteração anterior. Designando cada geração por um índice t, o fluxo geral de um AG simples é ilustrado na Fig. 2.

Procedure Algoritmo Genético

```
begin
    t ← 0
    inicializar P(t)
    avaliar P(t)
    while (not Condição_Terminal) do
    begin
        t ← t + 1
        selecionar P(t) a partir de P(t − 1)
        recombinar e mutar P(t)
        avaliar P(t)
    end
end
```

Figura 2: Fluxo básico de um algoritmo genético simples com três operadores: seleção, recombinação e mutação.

3.5 Inicialização

Na maior parte das aplicações, a população inicial de N indivíduos é gerada aleatoriamente ou através de algum processo heurístico.

Como no caso biológico, não há evolução sem variedade. Ou seja, a teoria da seleção natural ou "lei do mais forte" necessita de que os indivíduos tenham diferente graus de adaptação ao ambiente em que vivem. De acordo, é importante que a população inicial cubra a maior área possível do espaço de busca.

3.6 Avaliação e Adequabilidade

AGs necessitam da informação do valor de uma função objetivo para cada membro da população, que deve ser um valor não-negativo. Nos casos mais simples, usa-se justamente o valor da função que se quer maximizar. A função objetivo dá, para cada indivíduo, uma medida de quão bem adaptado ao ambiente ele está, ou seja, quanto maior o valor da função objetivo, maiores são as chances do indivíduo sobreviver no ambiente e reproduzir-se, passando parte de seu material genético a gerações posteriores.

A avaliação de cada indivíduo resulta num valor que, em inglês, é denominado "fitness". Na falta de tradução melhor, este autor emprega o termo *adequabilidade* neste trabalho.

3.7 Seleção

O mecanismo de seleção em AGs emula os processos de reprodução assexuada e seleção natural. Em geral, gera-se uma população temporária de N indivíduos extraídos com probabilidade proporcional à adequabilidade relativa de cada indivíduo na população, ou seja, a probabilidade de seleção de um cromossomo s é dada por

$$p_{\text{sel}} = \frac{a(s)}{\sum_{i=1}^{N} a(s_i)} \qquad (3)$$

onde $a(\cdot)$ é a função de adequabilidade.

Usando a probabilidade acima, selecionam-se N indivíduos. Neste processo, indivíduos com baixa adequabilidade terão alta probabilidade de desaparecerem da população, ou seja, serem extintos, ao passo que indivíduos adequados terão grandes chances de sobreviverem. Uma representação pictórica do processo de seleção é mostrada na Fig. 3.

3.8 Recombinação

O processo de recombinação é um processo sexuado — ou seja, envolve mais de um indivíduo — que emula o fenômeno de "crossover", a troca de fragmentos entre pares de cromossomos. Na forma mais simples, trata-se um processo aleatório que ocorre com probabilidade fixa p_{rec} que deve ser especificada pelo usuário. O processo de recombinação é ilustrado conceitualmente na Fig. 4.

3.9 Mutação

O processo de mutação em AGs é equivalente à busca aleatória. Basicamente, seleciona-se uma posição num cromossomo e muda-se o valor do gene correspondente aleatoriamente para um outro alelo possível. O processo é geralmente controlado

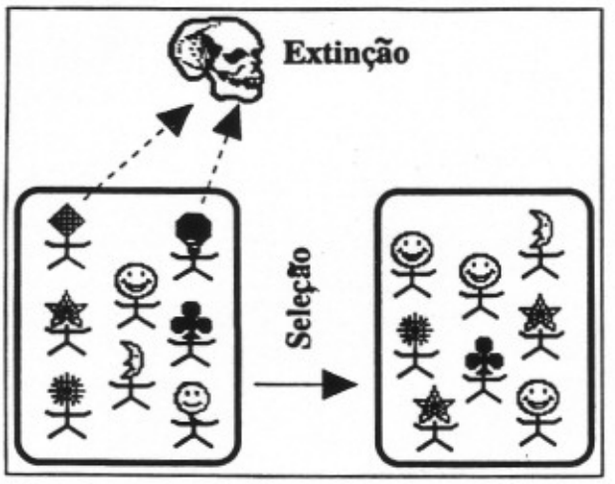

Figura 3: Processo de seleção para uma população de 8 indivíduos, resultando na extinção de 2 deles. Por outro lado, indivíduos de alta adequabilidade têm alta probabilidade de receber cópias.

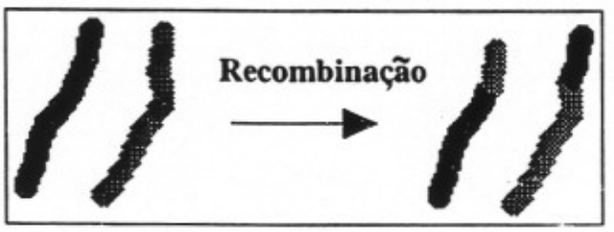

Figura 4: Processo de recombinação.

por um parâmetro fixo p_{mut} que indica a probabilidade de um gene sofrer mutação. O processo é ilustrado na Fig. 5.

Figura 5: Processo de mutação.

3.10 Condições de Término

Como estamos tratando de problemas de otimização, o ideal seria que o algoritmo terminasse assim que o ponto ótimo fosse descoberto. Jáno caso de funções multimodais, um ponto ótimo pode ser o suficiente, mas pode haver situações onde todos ou o maior número possível de pontos ótimos sejam desejados. Um problema prático é que, na maioria dos casos de interesse, não se pode afirmar com certeza se um dado ponto ótimo corresponde a um ótimo global. Como conseqüência, normalmente usa-se o critério do número máximo de gerações ou um tempo limite de processamento para parar um AG. Outro critério plausível é parar o algoritmo usando a idéia de estagnação, ou seja, quando não se observa melhoria da população depois de várias gerações consecutivas.

3.11 Outros Comentários

Como se vê nas Figs. 3–5, o processo de seleção ocorre ao nível do indivíduo, ao passo que recombinação e mutação ocorrem ao nível cromossômico. Embora populações na natureza tenham número variável de indivíduos, na grande maioria dos AGs o tamanho da população é fixo, por questões de simplicidade e facilidade de implementação em computador.

4 Exemplo de Execução

Esta seção ilustra a execução de um simples AG com um exemplo prático consistindo da otimização de uma função multimodal de uma variável real.

4.1 Definição do Problema

Consideremos um problema de maximização de uma função $f : \Re \rightarrow \Re$ dada por

$$f(x) = \cos(20x) - \frac{|x|}{2} + \frac{x^3}{4} \qquad (4)$$

no intervalo $-2 \leq x \leq 2$. A função é ilustrada na Fig. 6. Como se pode ver, trata-se de uma função não-linear com nada menos que 13 picos no intervalo de interesse, sendo um correspondente ao máximo global, ou ponto ótimo, $x^* \approx 1,88929$. Portanto,

$$f(x^*) \approx 1,73752 \geq f(x), \forall -2 \leq x \leq 2. \qquad (5)$$

Os pontos extremos de $f(\cdot)$ podem ser determinados igualando-se a primeira derivada de $f(\cdot)$ a zero e resolvendo a equação transcedental resultante numericamente. Se a expressão analítica de $f(\cdot)$ for desconhecida, porém, métodos numéricos são impotentes. Por se tratar de uma função multimodal, é evidente que puros métodos de gradiente não devem ser capazes de encontrar o ótimo global na maior parte das tentativas.

4.2 Representação Cromossômica

Para cada problema, encontrar uma representação cromossômica conveniente é sempre o primeiro passo para implementação de qualquer AG. Vamos usar um vetor binário para representar cada ponto do espaço de busca. Como há um número infinito de pontos no intervalo de interesse, a dimensão desse vetor ou seqüência binária depende da precisão requerida para o problema.

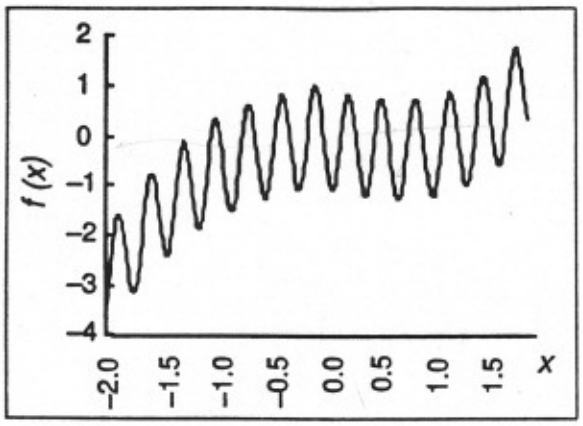

Figura 6: Função a ser maximizada.

Assumamos uma precisão de 4 casas decimais. Como o espaço de busca tem largura $2-(-2)=4$, tal precisão significa que o processo de busca deve distinguir pelo menos $4 \times 10^4 = 40.000$ pontos. Como resultando, seqüências binárias (cromossomos) de pelo menos 16 bits são necessárias, uma vez que

$$32.766 = 2^{15} < 40.000 < 2^{16} = 65.536. \quad (6)$$

Usando cromossomos binários de dimensão 16, dividimos o intervalo de busca em 2^{16} partes iguais e usamos uma mapeamento linear de cada seqüência binária para um número x no intervalo[2]. Usando a notação binária posicional (a mais comum), cada possível solução x.será representada por uma seqüência $s = [b_{15}b_{14}\ldots b_2b_1b_0]$, onde cada $b_i \in \{0,1\}$. A decodificação de um cromossomo é feita em duas etapas: primeiro converte-se o cromossomo da base 2 para base 10 de acordo com

$$s = [b_{15}b_{14}\ldots b_2b_1b_0] = \sum_{i=0}^{15} b_i 2^i = \bar{x}, \quad (7)$$

resultando num valor discreto $\bar{x} \in \aleph \mid 0 \leq \bar{x} \leq 2^{16}-1 = 65.535$. No passo seguinte, $\bar{x}$ é mapeado de volta ao espaço de busca de acordo com:

$$x = -x_{\min} + \frac{x_{\max} - x_{\min}}{2^{16}-1}\bar{x}, \quad (8)$$

onde $x_{\min}$ e $x_{\max}$ indicam os limites do intervalo de busca e são, neste exemplo, -2 e 2, respectivamente. Obviamente, os cromossomos [0000000000000000] e [1111111111111111] correspondem a esses limites.

4.3 População Inicial

O tamanho da população é, neste exemplo, fixada em $N = 50$ indivíduos. Conseqüentemente, 50 seqüências de 16 bits cada são geradas aleatoriamente para compor a população inicial. Para a escolha de cada bit, basta simular o lançamento de uma moeda honesta; o processo é repetido 800 vezes.

[2]Alternativamente pode-se dividir o espaço de busca em $2^{16}1$ partes iguais e duas partes menores para os extremos.

rótulo	s	x	$a(s)$
s_1	0101000000000000	0,4924	0,3443
s_2	1110101100001010	1,6725	2,9669
s_3	0001101010111010	-1,5824	3,5244
s_4	1000111000010101	0,2201	4.7445

Tabela 1: Alguns indivíduos da população inicial.

4.4 Função de Adequabilidade

A função de adequabilidade $a(\cdot)$ pode ser escolhida com alguma liberdade, mas deve obedecer a uma série de condições. Primeiro, deve resultar em um valor que, quanto maior, maiores as chances de sobrevivência e reprodução do indivíduo. Além disso, é conveniente que $a(\cdot)$ resulte sempre em valores positivos, para facilitar a implementação do processo de seleção. Para um cromossomo s, adotemos a seguinte função de adequabilidade:

$$a(s) = f(x) + 4, \quad (9)$$

que resulta sempre em valores positivos, conforme se pode deduzir da Fig. 6. A Tabela 1 mostra 4 indivíduos (cromossomos) da população inicial, bem como os valores de x e da adequabilidade correspondentes.

Para essa população inicial, em nossa simulação os seguintes valores de adequabilidade foram obtidos: mínimo $= 0,9808$, médio $= 3,5583$ e máximo $= 5,3750$. Além disso, a soma total dos valores de adequabilidade da população, ou seja, $\sum_{j=1}^{N} a(s_j)$, foi de $177,9153$. A adequabilidade relativa de cada indivíduo da população é então facilmente calculada como:

$$a_{\text{rel}}(s_i) = \frac{a(s_i)}{\sum_{j=1}^{N} a(s_j)}. \quad (10)$$

4.5 Exemplos de Resultados

A adequabilidade relativa de cada indivíduo dá uma medida da probabilidade do indivíduo sobreviver ao processo de seleção. O método de seleção mais conhecido é o chamado *método da roleta*, em que se imagina uma roleta fictícia num cassino fatídico onde, com base na adequabilidade de cada indivíduo relativamente à população, decide-se se ele deve *morrer* ou *sobreviver* e produzir descendentes. A roleta é girada e uma esfera é "jogada" cada vez que se necessita selecionar um indivíduo. Uma característica fundamental da roleta

rótulo	$a_{\text{rel}}(s)(\%)$	n_{esp}	n_{obt}
s_1	0,1935	0,0968	0
s_2	1,6676	0,8338	1
s_3	1,9809	1,9809	2
s_4	2,6667	1,3336	1

Tabela 2: Resultados da seleção de alguns indivíduos da população inicial.

é que cada cavidade ("casa") corresponde a um indivíduo, sendo a área da cavidade proporcional à adequabilidade, de modo que indivíduos de maior adequabilidade têm maior probabilidade de serem selecionados.

Exemplo de uma roleta aparece na Fig. 7, usando a população de indivíduos da Fig. 3. Como se trata de um processo probabilístico, indivíduos não são necessariamente selecionados de acordo com os valores de adequabilidade relativa. Contudo, pode-se estimar o número de cópias de um dado indivíduo, n_{esp}, multiplicando-se a sua adequabilidade relativa pelo tamanho da população.

Figura 7: Exemplo de roleta para seleção.

Para os cromossomos da Tabela 1, a Tabela 2 mostra a adequabilidade relativa, o número esperado de cópias devido ao processo de seleção e o número de cópias realmente obtidas através do método da roleta, n_{obt}. Neste exemplo, os resultados concordaram com os valores preditos. O processo de seleção resulta numa população temporária de N indivíduos.

Terminado o processo de seleção, certos cromossomos são extraídos aos pares da população resultante para recombinação. Na simulação, a probabilidade de um cromossomo sofrer recombinação, p_{rec}, foi fixa em 0,8 e empregou-se somente *recombinação aleatória de um ponto*. Neste processo, primeiro os cromossomos da população são pareados aleatoriamente. Em seguida, usando p_{rec} decide-se se cada par deve ou não sofrer recombinação. Em caso afirmativo, escolhe-se um ponto de corte aleatoriamente, e efetua-se a troca de segmentos correspondentes.

Por exemplo, suponhamos que a cópia do cromossomo s_2 seja pareada com a cópia de s_4, e que se tenha decidido que o par deve sofrer recombinação. Gera-se então um número aleatório entre 1 e $\ell(s) - 1$, onde $\ell(s)$ dá o comprimento da seqüência s, ou seja, 16 no caso deste exemplo. Esse número é usado para indicar em que posição os cromossomos devem ser cortados e então recombinados. Por exemplo, se for decidido que os últimos 5 bits devem ser trocados entre as cópias de s_2 e s_4 (cromossomos *pais*), os cromossomos *filhos* resultantes serão:

1110101100010101 e 1000111000001010

que poderão ou não ser melhores (ter melhor adequabilidade) que os pais.

O processo de mutação pode ocorrer depois de ou ao mesmo tempo que a recombinação. No caso deste exemplo, fixou-se a probabilidade de mutação p_{mut} em 0.01 para cada gene de modo que, em média, $p_{\text{mut}} \times \ell(s) \times N = 8$ bits devem sofrer mutação a cada geração. Obviamente, em se tratando de mutação puramente aleatória, o cromossomo resultante poderá ter ou não maior valor de adequabilidade.

4.6 Resultados Finais

Partindo-se de uma população de $N = 50$ indivíduos gerados aleatoriamente, usando o método da roleta para seleção e com probabilidades de recombinação e mutação fixadas em 0,8 e 0,01, respectivamente, o máximo global foi encontrado com a desejada precisãode 4 casas decimais na geração de número 48. A Fig. 8 mostra os valores da função $f(\cdot)$ para o melhor e o pior indivíduo da população, bem como o valor médio, para um total de 50 gerações. Como fica claro na figura, este simples AG teve sucesso em obter o ótimo global de $f(\cdot)$ em poucas gerações.

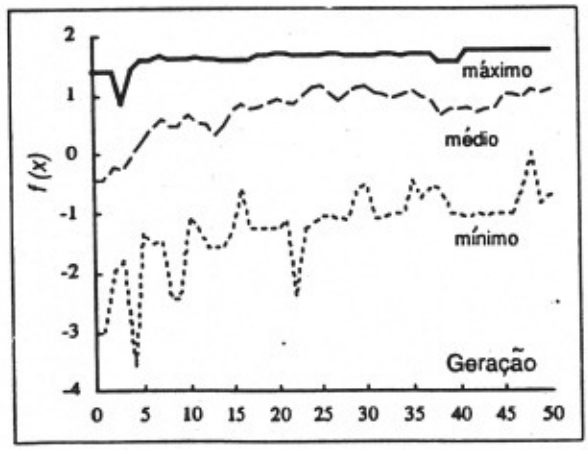

Figura 8: Exemplo de roleta para seleção.

Neste caso, terminou-se a execução do AG depois de 50 gerações, e adotou-se o procedi-

mento de sempre memorizar o melhor indivíduo obtido, independentemente do indivíduo estar ou não presente na população. Terminado o algoritmo, decodifica-se este melhor indivíduo de volta ao espaço de busca original. No caso desta simulação, o melhor indivíduo, obtido na geraçãode número 48, foi

$$s^{\text{melhor}} = 1010100010011111$$

com adequabilidade

$$a(s^{\text{melhor}}) = 5,7373$$

Decodificando-se o melhor cromossomo s^{melhor}, obtém-se

$$x^{\text{melhor}} = x^* = 1,8893$$

que corresponde ao máximo global com a desejada precisão.

No total, as 48 gerações equivaleram a $48 \times 50 = 2.400$ indivíduos averiguados, representando $3,66\%$ do espaço de busca de 2^{16} possíveis seqüências. Considerando-se que tudo o que foi feito consistiu em avaliar indivíduos, selecioná-los com uma roleta fictícia, trocar partes de seqüências e mudar um ou outro bit eventualmente, os resultados podem parecer surpreendentes. Na realidade, existem muitas variantes melhoradas do algoritmo usado nesta simulação, e resultados ainda mais expressivos podem ser obtidos, ilustrando o poder dos algoritmos genéticos. Em termos de tempo de processamento, o programa usado aqui foi escrito em C, e 50 gerações consumiram aproximadamente $0,2$s em uma estação de trabalho SUN-4.

4.7 Detalhes de Implementação

A implementação do AG usado nesta simulação é direta, e pode ser feita sem problemas em qualquer linguagem de programação como Pascal ou C. Seqüências binárias podem ser representadas facilmente como arranjos de inteiros, caracteres, ou mesmo usando uma representação mais econômica de 1 bit para cada gene. Em máquinas convencionais onde um inteiro é representado por 2 bytes e um carácter por 1, cada cromossomo ocuparia 32 bytes de memória no primeiro caso, 16 bytes no segundo, e somente 2 bytes com o mapeamento "1 bit = 1 gene". Este mapeamento faz um uso mais eficaz da memória disponível, mas torna os processos de codificação e decodificação um pouco mais complexos.

5 Problemas Práticos dos AGs

5.1 Papel dos Operadores

Em AGs simples como o descrito na Fig. 2, o papel dos operadores seleção, recombinação, e mutação pode ser visto da seguinte maneira simplista: o processo de seleção copia indivíduos da população de modo proporcional à adequabilidade relativa; recombinação combina pedaços de material genético, na esperança de que indivíduos melhores que os pais sejam obtidos; finalmente, mutação garante que todo o repertório genético esteja representado na sinfonia evolutiva, ou seja, garante que nenhum alelo desapareça para sempre da população.

Se não houvesse o processo de seleção, os AGs, além de perderem grande parte do carácter evolutivo, seriam processos ineficientes similares a buscas aleatórias. Sem recombinação, haveria somente busca aleatória a partir dos melhores elementos da população. Combinados, os processos de seleção e recombinação realizam uma espécie de busca local nas proximidades dos melhores indivíduos da população. Finalmente, sem mutação os AGs efetuariam busca usando somente a informação contida na população, e não teriam como repor material genético perdido durante o processo de seleção. Por exemplo, se o ponto ótimo de uma função exigisse que os três últimos elementos de um cromossomo binário sejam 101, uma condição necessária para que um AG sem mutação tivesse sucesso seria que pelo menos um indivíduo da população inicial tivesse a subseqüência 101 nas devidas posições. Mesmo assim, sem mutação essa subseqüência poderia desaparecer por completo da população devido à seleção, e o AG não poderia otimizar a função. Com mutação esse problema é, pelo menos em princípio, contornado.

5.2 Determinação de Parâmetros

Num AG básico, o usuário deve definir o tamanho da população, N, além das probabilidades de recombinação e mutação, respectivamente, p_{rec} e p_{mut}. Em AGs mais sofisticados, há ainda mais parâmetros, comprometendo parte da robustez dos algoritmos. Infelizmente, não há regras claras para a escolha desses parâmetros.

Quanto ao parâmetro N, a intuição indica que "quanto mais, melhor", uma vez que, em última análise, com uma população infinita cobrindo todo o espaço de busca, a solução ótima seria obtida na primeira geração. Na prática, é óbvio, temos que nos consignar com populações finitas. Partindo da análise de esquemas, Goldberg concluiu que, para cromossomos binários de comprimento ℓ, o tamanho ótimo deve ser uma função exponencial de ℓ [52]. Na prática, valores da ordem de 50 a 200 cromossomos resolvem a maior parte dos problemas, mas populações maiores podem ser necessárias para problemas mais complexos, como é o caso da geração automática de programas [90]. Em [126], consideram-se problemas onde pequenas populações são necessárias, geralmente em casos em que a avaliação de um cromossomo é excessivamente lenta. Nesses casos, propõe-se um

método intuitivo de escolha da população inicial de modo que os cromossomos representem a maior área possível no espaço de busca.

Em relação às probabilidades de recombinação e mutação, estudos empíricos [36] têm mostrado que bons resultados geralmente são obtidos com alto valor de p_{rec} ($\geq 0,7$) e baixo valor para p_{mut} (geralmente $\leq 0,01$). Métodos adaptativos são apresentados na próxima seção.

5.3 Erros de Amostragem

A seleção implementada pelo método da roleta é sujeita a erros de amostragem. Por exemplo, na idéia proposta é totalmente aceitável que, numa população de, digamos, 100 indivíduos, exatamente 100 cópias do pior indivíduo sejam obtidas. Erros de amostragem podem resultar na perda de informação genética importante e a resultados de baixa qualidade. Métodos de atenuação desses erros são revisados na próxima seção.

5.4 Convergência Prematura

Usando o modelo do AG simples para a otimização de funções multimodais, um fenômeno que se observa comumente é que o AG converge muito rapidamente (em umas poucas dezenas de gerações) para um ponto de alta qualidade, mas não o ótimo global, num fenômeno denominado *convergência prematura*.

Por exemplo, suponhamos que na população inicial um dado indivíduo próximo de um ótimo local, mas não global, tenha um valor de adequabilidade muito maior que o restante da população. Devido ao processo de seleção, tal *super-indivíduo* terá vários descendentes na próxima geração. Em casos extremos, indivíduos próximos ao ótimo global mas com baixa adequabilidade relativa, podem ser completamente extintos, o que geralmente resulta em convergência da população para um ótimo local não-global.

Assim sendo, *convergência prematura* está intimamente relacionada com a *perda de diversidade* da população. Em CE, *diversidade* é um conceito abstrato que indica o grau em que as mais diversas regiões do espaço de busca estão representadas na população, ou seja, o quão variada é a população.

5.5 Balanço Exploração-Exploração

Um dilema fundamental em AGs é o chamado *balanço exploração-exploração*. Infelizmente, a tradução para o português perde a sutileza do original em inglês, "balance exploration-exploitation", que seria o balanço entre "explorar" no sentido de investigar regiões desconhecidas do espaço de busca, e "explorar" no sentido de usufruir do conhecimento genético de indivíduos da população.

Basicamente, recombinação usa a bagagem genética de indivíduos, ao passo que mutação faz uma busca aleatória no espaço de busca. Deste modo, recombinação "exploits" indivíduos, e realiza uma busca local, enquanto que mutação "explores" novas regiões, realizando uma busca global. Muita "exploração= exploitation" e pouca "exploração= exploration" acelera a convergência, mas pode tornar a busca demasiadamente localizada, ineficaz para ambientes multimodais; no caso oposto, a busca pode ficar desordenada e assemelhar-se a uma busca aleatória.

5.6 Tempo de Processamento

Um obstáculo a grande parte das aplicações práticas dos AGs é o excessivo tempo de processamento. Em geral, a culpa não é dos AGs em si, mas do problema em questão. Geralmente AGs conseguem obter soluções no mínimo sub-ótimas investigando apenas uma fração do espaço de busca. Quando o espaço de busca é demasiadamente amplo, porém, a busca genética pode ser excessivamente lenta.

Em minha experiência, a maior parte do tempo de processamento em aplicações de AGs é gasta na computação da adequabilidade dos indivíduos. Por exemplo, se um cromossomo representa parâmetros da asa de um avião, o cálculo da adequabilidade pode requerer uma análise de elementos finitos que pode levar horas num computador. Possíveis soluções são os AGs paralelos, que são descritos na próxima seção.

5.7 Problemas Enganadores

Em [94], quatro razões que podem evitar que um AG encontre o ótimo global de uma função foram citadas: a) representação cromossômica não-apropriada, b) grandes erros de amostragem, c) efeito destrutivo da recombinação, e d) o problema é *enganador* ("deceptive", em inglês). Problemas enganadores, que surgem principalmente com cromossomos binários, enganam a busca genética e tendem a fazer o AG convergir para uma solução que não é globalmente ótima. Nesses problemas, há fragmentos cromossômicos que têm alta adequabilidade em separado mas que, juntos, formam uma solução não-ótima [35, 65, 94].

Uma estratégia para enfrentar problemas enganadores é aplicar uma transformação do espaço de busca, de modo que o problema se torne mais fácil para AGs. Em [94], um método usando uma transformação linear é descrito. Outra idéia são os AGs *bagunçados* ("messy GAs") [58], que são eficazes para muitos problemas enganadores.

6 Variações de AGs

Conforme explicado na seção anterior, o AG simples descrito acima sofre de vários problemas. Para aumentar a eficiência da busca genética, desde há vários anos a literatura técnica tem sido inundada por um número enorme de variações de AGs, e algumas das técnicas mais eficazes são apresentadas aqui.

6.1 Escalamento de Adequabilidade

A escolha de uma função de adequabilidade apropriada para cada problema é fundamental. Em problemas de maximização onde a função a ser otimizada dá sempre valores reais e positivos, há uma tentação natural em se usar essa função como uma medida de adequabilidade[3], o que nem sempre é uma estratégia inteligente.

O mapeamento *função objetivo → função adequabilidade* deve ser feito com cuidado. Em primeiro lugar, por uma questão semântica, é interessante que a adequabilidade de um indivíduo seja tal que, quanto mais alta, maiores sejam as chances de sobrevivência[4]. Em segundo lugar, vários métodos de seleção exigem que os valores de adequabilidade sejam sempre positivos, como é evidente no método da roleta.

Problemas de minimização podem ser convertidos facilmente em problemas de maximização através de uma simples mudança de sinal. Valores negativos de adequabilidade podem ser obtidos somando-se um certo valor mínimo a todos os valores crus de adequabilidade.

Nas primeiras gerações da execução de um AG, é interessante que a seleção não seja um processo muito rigoroso, de modo que mesmo indivíduos de baixa qualidade relativa tenham alguma chance de evoluir. Em jargão técnico, é interessante que a *pressão seletiva* seja baixa no início do processo evolutivo. Quando a população já está madura, depois de muitas gerações, os indivíduos tendem a ter valores de adequabilidade muito próximos, o que acaba dificultando que os melhores indivíduos se destaquem na população. Em outras palavras, nas últimas gerações é interessante aumentar a pressão seletiva.

A pressão seletiva está implicitamente relacionada com a diversidade da população. Alta pressão seletiva tende a fazer a diversidade cair rapidamente, levando a população a convergir em poucas gerações, o que pode resultar em convergência prematura. Por outro lado, uma baixa pressão seletiva pode tornar o processo de busca proibitivamente lento.

[3] Essa escolha é referida neste artigo como "adequabilidade crua".

[4] Alguns autores (por exemplo, Koza [90]), estranhamente preferiram corromper o significado do vocábulo, e buscam indivíduos de baixa adequabilidade.

6.1.1 Escalamento Linear

Uma maneira fácil de se controlar a pressão seletiva é o escalamento da função de adequabilidade. Goldberg propôs o seguinte mapeamento linear [56]:

$$a(\cdot) = \alpha f(\cdot) + \beta \tag{11}$$

onde $a(\cdot)$ e $f(\cdot)$ são as funções de adequabilidade e objetivo (ou adequabilidade crua), respectivamente, e α e β são coeficientes determinados para cada geração. Dados os valores mínimo, médio e máximo de $f(\cdot)$ e um parâmetro γ indicando a razão desejada entre a adequabilidade do melhor indivíduo e a adequabilidade média, α e β são calculados assim:

$$\text{se } f_{\min} \geq \frac{\gamma f_{\text{med}} - f_{\max}}{\gamma - 1} \tag{12}$$

$$\text{então} \begin{cases} \alpha = \dfrac{f_{\text{med}}{}^{(\gamma-1)}}{f_{\max} - f_{\text{med}}} \\ \beta = \dfrac{f_{\text{med}}(f_{\max} - \gamma f_{\text{med}})}{f_{\max} - f_{\text{med}}} \end{cases} \tag{13}$$

$$\text{senão} \begin{cases} \alpha = \dfrac{f_{\text{med}}}{f_{\max} - f_{\min}} \\ \beta = -\dfrac{f_{\text{med}} f_{\min}}{f_{\text{med}} - f_{\min}} \end{cases} \tag{14}$$

Traduzindo em palavras, a idéia expressada na Eq. (13) é efetuar o mapeamento linear de modo que indivíduos de adequabilidade crua média continuem na média após o escalamento e que, se possível, o máximo valor da adequabilidade escalada seja igual ao valor médio vezes γ. Se, porém, esta última condição implicar em valores negativos de adequabilidade, conforme verificado em (12), então os coeficientes α e β são determinados de modo a mapear o mínimo valor de adequabilidade crua para o valor 0 (Eq. (14)). Valores de γ entre 1,2 e 2 são recomendados. Métodos de escalamento evitam que alguns *super-indivíduos* dominem a população no início da busca, e geralmente produzem melhores resultados na otimização de funções multimodais.

6.1.2 "Janela" de Adequabilidade

Neste método, subtrai-se o menor valor da adequabilidade crua de todos os valores antes do processo de seleção, de modo que o pior indivíduo fica impossibilitado de ter descendentes na próxima geração. Este método é de implementação simplíssima, e diminui um pouco a pressão seletiva, embora não seja tão eficaz quanto o escalamento linear.

6.1.3 Graduação

De modo a evitar variações indesejáveis na pressão seletiva, Baker propôs uma medida de adequabilidade parcialmente desassociada da função objetivo [12]. Os cromossomos são ordenados de acordo com os valores da função objetivo, de modo que o primeiro cromossomo seja o melhor, o segundo seja o segundo melhor, e assim por diante; ou seja, os cromossomos são graduados ("ranked") e os valores de adequabilidade são atribuídos baseados somente na ordem.

Por exemplo, numa população de 100 indivíduos, um método simples seria atribuir adequabilidade 100 para o melhor indivíduo, 99 para o seguinte, e assim por diante.

A eficácia desse método é ainda polêmica dentro da comunidade dos AGs [56, 153], mas muitas implementações práticas de AGs possuem essa opção de método de seleção.

6.1.4 Corte σ

De modo a evitar que indivíduos muito ruins sejam reproduzidos por erros de amostragem, um método simples é "cortar" todos os pontos cuja adequabilidade seja menor que a média por mais do que um valor fixo, especificado em termos do desvio padrão(geralmente representado pela letra 'σ'). Por exemplo, pode-se atribuir adequabilidade $= 0$ para todos os indivíduos com adequabilidade menor que a média menos duas vezes o desvio padrão.

6.1.5 "Nicho" e Compartilhamento

O processo seletivo na maior parte dos AGs é extremamente impiedoso: todos os indivíduos competem entre si, sempre favorecendo os indivíduos mais "fortes" da população. Na natureza, contudo, espécies distintas e que tenham diferentes papéis (*nichos*) ecológicos não competem entre si e podem conviver num mesmo ambiente.

Essa idéia levou ao conceito de *compartilhamento* ("sharing") [55], em que a adequabilidade dos indivíduos é modificada de tal modo a diminuir o nível de competição entre indivíduos que estejam distantes no espaço de busca. Por exemplo, Goldberg e Richardson propuseram a seguinte função de adequabilidade

$$a(x_i) = \frac{f(x_i)}{\sum_{j=1}^{N} \lambda(\| x_i - x_j \|)} \qquad (15)$$

onde $\| x - y \|$ denota a distância Euclidiana entre os pontos, e $\lambda(\cdot)$ é a função de compartilhamento, que deve ser uma função decrescente da distância entre os pontos. É comum que $\lambda(\cdot)$ seja escalada de modo a resultar em 1 para a menor distância entre dois indivíduos da população e 0 para a maior distância.

Na otimização de uma função multimodal, o compartilhamento resulta na formação de subpopulações de indivíduos concentrados nos picos da função. Sem compartilhamento, a tendência do AG seria convergir para um único pico, não necessariamente o mais alto. Além disso, o número de indivíduos em cada subpopulação é proporcional à altura relativa dos pico correspondente, conforme indicado na Fig. 9.

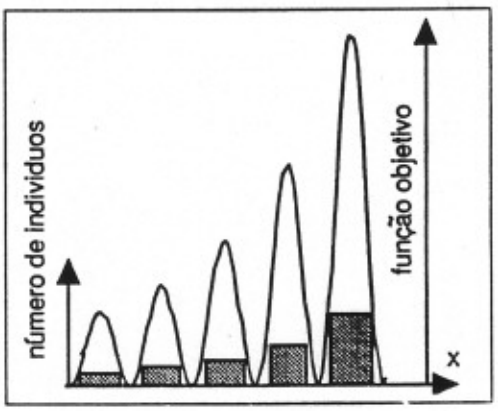

Figura 9: AG com compartilhamento. Subpopulações se concentram próximo aos picos, conforme indicam os histogramas.

Contra esse método pesam os fatos de que o cálculo das distâncias requer considerável tempo de processamento ($O(N^2/2)$), e a prática demonstra que as subpopulações, embora próximas aos ótimos locais, geralmente não convergem para eles.

6.2 Seleção com Baixo Erro de Amostragem

DeJong [36] realizou um árduo estudo empírico de vários métodos de seleção.

6.2.1 Modelo do Valor Esperado

Neste modelo, às vezes também denominado *amostragem estocástica sem reposição*, primeiramente para cada indivíduo calcula-se o número esperado de cópias a partir do valor da adequabilidade relativa computada pela Eq. 3. Em seguida, cada vez que um indivíduo é selecionado (pelo método da roleta, por exemplo), subtrai-se 1 do correspondente número de cópias. Se esse número for negativo ou nulo, o correspondente indivíduo fica proibido de ser selecionado novamente. Assim, garante-se que o número de cópias que cada indivíduo recebe não pode exceder 1+ a parte inteira do número esperado de cópias.

		Método de Seleção				
s	n_{esp}	R	D	E	RE	T
s_1	0,8	0 a 5	1	0 ou 1	0 a 2	0 a 5
s_2	1,6	0 a 5	2	1 ou 2	1 a 3	0 a 5
s_3	1,2	0 a 5	1	1 ou 2	1 a 3	0 a 5
s_4	1,0	0 a 5	1	1	1	0 a 5
s_5	0,4	0 a 5	0	0 ou 1	0 a 2	0

Tabela 3: Número de cópias para vários métodos de seleção. R = roleta, D = determinística, E = esperado, RE = resto estocástico, T = torneio.

6.2.2 Modelo de Amostragem Determinística

Como no método anterior, primeiro calcula-se o número esperado de cópias para cada indivíduo. De modo determinístico, alocam-se cópias correspondentes às partes inteiras dos valores esperados. As partes fracionárias são então colocadas em ordem decrescente, e as cópias restantes são então extraídas a partir do topo dessa lista, uma a uma. Ou seja, a seleção é totalmente determinística, sem depender da sorte.

6.2.3 Modelo do Resto Estocástico

Igual ao caso anterior, mas as partes fracionárias dos números de cópias esperadas são usados para compor uma roleta, a partir da qual as cópias restantes são selecionadas.

6.2.4 Modelo de Torneio Estocástico

Neste método, usa-se o método da roleta para extrair um par de indivíduos por vez, mas apenas o melhor de cada par (o de maior valor de adequabilidade) é realmente selecionado.

Vários outros métodos podem ser imaginados. Não existe um consenso sobre o melhor e mais justo método de seleção, mas eu tenho usado o modelo do resto estocástico com sucesso. A Tabela 3 indica o número de cópias obtidas para cada método de seleção para uma pequena população de $N = 5$ indivíduos com adequabilidades $a(s_1) = 4$, $a(s_1) = 8$, $a(s_1) = 6$, $a(s_1) = 5$, e $a(s_1) = 2$, o que dá uma média de 5. É evidente que o método da roleta, embora amplamente utilizado, dá margem a potencialmente nocivos erros de amostragem.

6.3 Outros Modelos Populacionais

Ao invés de somente produzir uma nova população e substituir a população anterior, existem várias alternativas. DeJong propôs substituir apenas uma fração da população por iteração[36], mas isso só vem a aumentar o número de parâmetros que o usuário tem de escolher.

6.3.1 Eliminação de Duplicatas

Modelos como o AG simples descrito na Fig. 2 permitem rápida perda de diversidade na prática. Para atenuar esse fenômeno, uma alternativa óbvia é eliminar a possibilidade de duplicatas na população[33]. Se, por exemplo, houver cromossomos duplicados depois de executados os processos de seleção, recombinação, e mutação, então as duplicatas são eliminadas e novos cromossomos são gerados para completar a população. Obviamente, o tempo de processamento de cada geração aumenta devido à necessidade se comparar cada indivíduo com o restante da população.

6.3.2 Estratégias Elitistas

Neste modelo, amplamente utilizado, garante-se que os melhores indivíduos de uma geração sempre aparecerão na geração seguinte. Ou seja, se a elite da população corrente não estiver presente na próxima população em decorrência de alguma operação genética, então os elementos ausentes são inseridos artificialmente no lugar dos piores indivíduos. O método mais comum de elitismo supervisiona apenas o melhor indivíduo da população, mas para grandes populações pode ser interessante garantir que, digamos, os dez melhores indivíduos sempre estejam presentes na próxima geração.

6.3.3 Reprodução de Estado Estável

Ao invés de substituir toda a população de uma vez, este modelo pressupõe que somente alguns indivíduos devam ser trocados a cada geração[153, 143, 33, 144]. No caso mais simples, insere-se apenas um indivíduo por vez no lugar do pior indivíduo da população atual.

6.3.4 Modelo de Geração Contínua

A idéia básica deste modelo é permitir a sobrevivência dos "pais" de boa qualidade. Conforme indicado na Fig. 10, primeiro a população atual é copiada[5], e uma outra população temporária é gerada através de recombinação e mutação. A partir de uma população atual de N indivíduos, um total de $2N$ indivíduos são assim gerados. Finalmente, N indivíduos são selecionados a partir dos $2N$ usando-se um método de seleção qualquer [103, 145]. Embora seja apelativo, este método requer o dobro de computações de adequabilidade por geração.

[5]Obviamente trata-se de uma cópia virtual, pois na prática não há necessidade de duplicar-se cada indivíduo.

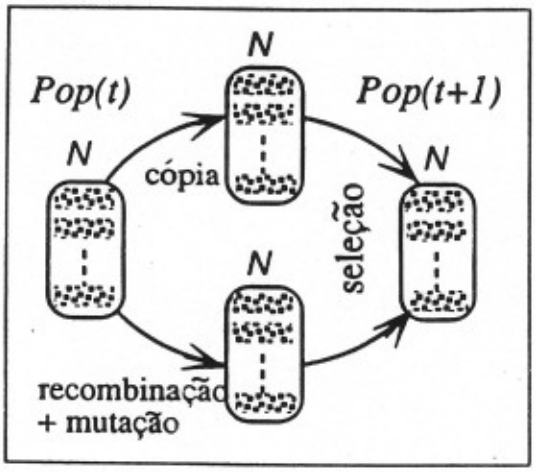

Figura 10: Modelo de geração contínua. A população seguinte é selecionada a partir da população atual e de indivíduos gerados por recombinação e mutação.

6.3.5 Genocídio Periódico

Em problemas práticos, muitas vezes o tempo de computação nos obriga a usar pequenas populações. Isso, porém, pode implicar amostragem imparcial na população inicial, além de rápida perda de diversidade. Nesses casos, um procedimento de fácil implementação é o genocídio (minha tradução do inglês "decimation"), em que, de tempos em tempos, a população de N indivíduos é aumentada para, digamos, $10N$ e, em seguida, novamente reduzida para N indivíduos através de uma seleção rigorosa que "mata" $9N$ indivíduos em excesso. Os indivíduos adicionais podem ser gerados por recombinação, mutação, ou simplesmente aleatoriamente. Embora seja apelativa, esta idéia introduz uma série de novos problemas, como a freqüência ideal de genocídio, em quanto se deve aumentar a população e como, etc.

6.3.6 Super-Seleção

Neste método, divide-se uma grande população em subpopulações de acordo com o nível de adequabilidade, e fixa-se o percentual de indivíduos a serem gerados a partir de cada subpopulação, de modo a amenizar erros de amostragem e acelerar a convergência. Por exemplo, pode-se dividir uma população de 2.000 indivíduos em 3 subpopulações, a primeira contendo os 100 melhores, a segunda outros 400 e a terceira, os 1.500 piores indivíduos, e então gerar-se 500, 700, e 1.200 indivíduos a partir das primeira, segunda, e terceira subpopulações, respectivamente [90].

6.4 Melhores Códigos

Os chamados "Códigos de Gray" fornecem uma atraente alternativa aos códigos binários posicionais discutidos até agora. Nos códigos de Gray, números consecutivos são representados por seqüências que diferem num único bit [50]. Por exemplo, os números de 0 a 7 são representados em três bits por $\{000, 001, 010, 011, 100, 101, 110, 111\}$ usando o código binário posicional, e por $\{000, 001, 011, 010, 110, 111, 101, 100\}$ nos códigos de Gray mais simples[6].

Hollstien propôs o uso de códigos de Gray em AGs aplicados à otimização de funções[79], e outras aplicações demonstram que esses códigos de fato minimizam o fenômenos das *colinas de Hamming* [145]. A conversão entre códigos binários posicionais e os de Gray é um processo simples, que pode ser facilmente incluído em qualquer AG [120].

Em várias aplicações cromossomos binários não são apropriados. Há muitos AGs propostos na literatura onde os genótipos envolvem números inteiros [103, 147, 148, 149] ou números reais [103, 145].

6.5 AGs Paralelos e Distribuídos

Conforme explicado anteriormente, um dos pontos fracos dos AGs é a baixa velocidade de execução, problema que é particularmente enfatizado com grandes populações. Desde o fim da década passada, vários modelos de implementação de AGs em computadores paralelos ou em redes de computadores têm sido propostos [28, 108, 109, 110, 116, 146].

Há basicamente duas categorias de AGs paralelos: os AGs insulares (que usam o modelo da ilha) e os AGs celulares, muitas vezes referidos como AGs massivamente paralelos ou AGs de granularidade fina [60], mas que podem ser considerados como uma subclasse de automata celulares [25, 151, 155].

6.5.1 AGs Insulares

Estes são talvez os modelos mais simples e apelativos, em que a população é particionada em subpopulações e cada subpopulação de indivíduos é atribuída a um processador de um computador paralelo com memória distribuída. Cada processador executa um AG convencional em sua subpopulação, mas periodicamente envia cópias dos melhores indivíduos para um processador vizinho, recebendo cópias dos melhores indivíduos do vizinho em troca. As cópias recebidas são, em geral, usadas para substituir os piores elementos da subpopulação, e o processo continua até que algum critério de término seja satisfeito. Esse processo de troca de indivíduos entre subpopulações "ilhadas" é denominado *migração*, conforme ilustrado na Fig. 11.

É evidente que o tempo de processamento em AGs cresce com o tamanho da população. As-

[6] Há, na realidade, múltiplos códigos de Gray, mas os descritos no texto são os mais comuns.

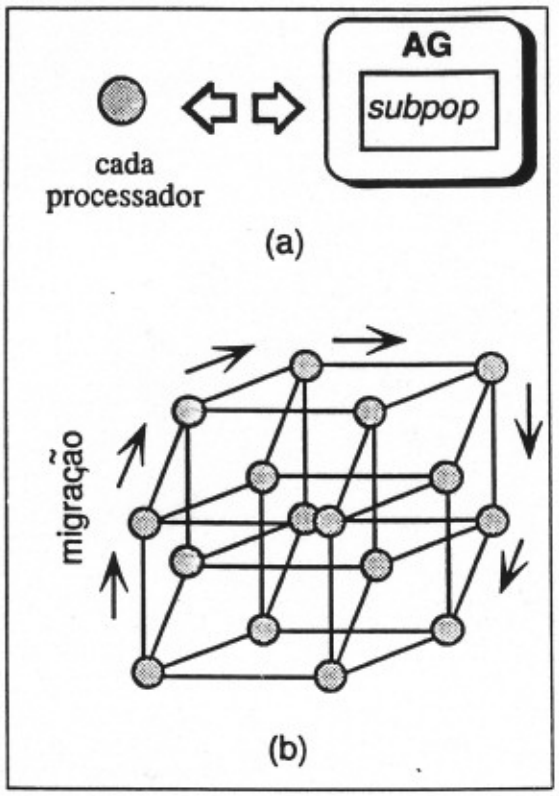

Figura 11: Esquema de um AG insular num computador hipercúbico de quarta dimensão. (a) Cada processador tem sua própria subpopulação e executa um AG. (b) Os melhores indivíduos de cada subpopulação migram periodicamente para processadores vizinhos.

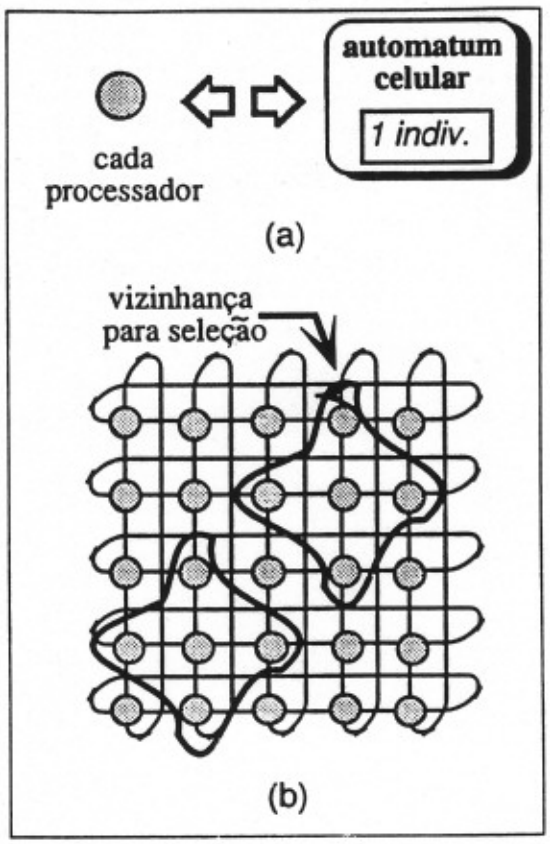

Figura 12: Esquema de um AG celular num computador paralelo de malha toroidal. (a) Cada processador tem apenas um indivíduo e equivale a uma máquina estocástica de estados finitos. (b) Cada processador executa seleção e recombinação em sua vizinhança.

sim sendo, pequenas subpopulações tendem a convergir rapidamente quando isoladas, espelhando a hipótese evolucionista conhecida como *Equilíbrios Pontuados*, em que se propõe que o processo evolutivo não seja gradual, mas sim ocorra aos saltos [39]. O uso da migração visa a dar um carácter global à busca, perturbando o equilíbrio de cada subpopulação[28].

Aceleração quase linear em relaçãoao número de processadores empregados foi relatada por Tanese usando um computador paralelo com rede hipercúbica [146]. Eu também tenho investigado AGs insulares, e obtive resultados similares usando um computador com 512 processadores conectados numa malha toroidal.

O estudo de estratégias eficientes de paralelização, como a estimação da freqüência ideal de migração, etc., ainda são assuntos abertos a investigação.

6.5.2 AGs Celulares

Outros paradigmas de AGs paralelos têm sido propostos, notadamente por Mühlenbein, cuja idéia básica é ter uma única população em que cada indivíduo realize busca local por "hill-climbing" [108, 109, 110] em sua vizinhança.

Neste modelo, ilustrado na Fig. 12, idealmente cada processador de um computador massivamente paralelo controla um só indivíduo. A população é distribuída numa superfície geográfica virtual, na maioria das vezes correspondendo a uma malha retangular, que pode ou não refletir o padrão de conectividade dos processadores. Cada processador seleciona um elemento em sua vizinhança, executa as operações de recombinação e mutação com algum outro vizinho, e escolhe o melhor entre os filhos resultantes. Como não há necessidade de globalização do processamento, a busca se torna assíncrona e a população total tende a manter um bom nível de diversidade [108, 109, 110].

6.6 Operadores Melhorados

Um grande número de operadores genéticos tem sido proposto na literatura. Na grande maioria, tratam-se de operadores específicos para um dado problema e uma dada representação cromossômica [103, 137], mas há também operadores de uso geral.

6.6.1 Variações de Recombinação

Uma variante de recombinação é fazer o pareamento dos cromossomos de acordo com o grau de adequabilidade, ao invés do simples pareamento aleatório. Ou seja, indivíduos de alta qualidade pareiam com outros indivíduos de alta qualidade, e vice-versa. Esta idéia geralmente aumenta a velocidade de convergência às custas de perda de poder de busca global.

Outra alternativa de recombinação é o uso de dois ou mais pontos de corte [36, 56, 140]. Infelizmente, a eficiência desses operadores é ainda assunto de debate.

Por outro lado, a *recombinação uniforme*, proposta originalmente por Syswerda [143] tem ganho muitos adeptos ultimamente, por ser superior à recombinação de um ou dois pontos. A idéia foi originalmente desenvolvida para cromossomos binários, mas existem extensões para outros domínios. Com recombinação uniforme, primeiro gera-se um padrão binário aleatório do mesmo comprimento dos cromossomos. Este padrão é então interpretado de modo que cada '1' indica que os genes correspondentes nos cromossomos pais devem ser trocados, ao passo que '0's indicam que os genes correspondentes não são trocados.

Por exemplo, tomemos dois cromossomos $s_1 =$ 10101010 e $s_2 =$ 11110000, e suponhamos que o seguinte padrão de recombinação $s_{\text{rec}} =$ 01100110 tenha sido gerado aleatoriamente. Por conseguinte, da esquerda para a direita, os segundo, terceiro, sexto e sétimo genes devem ser trocados, resultando nos cromossomos $\hat{s}_1 =$ 11101000 e $\hat{s}_2 =$ 10110010.

6.6.2 Operadores Adaptativos

Até o momento, a busca de operadores adaptativos de uso geral não produziu muitos resultados. Um método pelos menos conceitualmente interessante é o de otimizar os parâmetros de um AG usando um outro AG, chamado de *meta-AG* [62]. O método, originalmente proposto por Grefenstette, tem tido uns poucos seguidores, mas peca pela ineficiência. Uns poucos outros métodos tem sido desenvolvidos, mas em geral têm baixo desempenho [133].

Em [11] sugere-se o uso dos códigos de Gray com probabilidade de mutação constante e igual a $1/\ell$, onde ℓ é o comprimento de um cromossomo em bits, para problemas de otimização de funções unimodais com cromossomos binários. Para funções multimodais, contudo, argumenta-se que é necessário variar a probabilidade de mutação de modo a evitar ótimos locais, mas não se chega a nenhuma conclusão mais concreta.

6.6.3 Mais Operadores

Outros operadores incluem, entre outros, *permutação* [56, 114], *inversão* [75, 56], e *diploidismo* [138]. Infelizmente, não existe uma base teórica de quais são os melhores operadores e quando aplicá-los a problemas práticos gerais.

6.7 Computadores Genéticos

O advento de dispositivos lógicos programáveis, particularmente os FPGAs [8], aumentou consideravelmente as possibilidades de implementação em "hardware" de muitos algoritmos, com ganhos de até 3 ordens de magnitude em velocidade de execução em relação aos mesmos algoritmos executados em "software" [24]. Uma tentativa pioneira de implementação de um AG em "hardware" usando técnicas de "pipelining" e paralelismo é descrita em [132]. Atualmente, em meu laboratório o protótipo de um computador paralelo do tipo SIMD ("single-instruction-multiple-data"), específico para aplicações de AGs, está sendo desenvolvido.

7 Áreas de Aplicação

7.1 Problemas Clássicos

Muitos problemas clássicos de otimização combinatória são NP, ou seja, não-polinomiais, indicando que o espaço de busca cresce exponencialmente com as dimensões do problema [84, 85, 95]. Exemplos clássicos incluem o problema do vendedor ambulante (PVA ou "traveling salesman problem", TSP) [93, 95], o problema da mochila ("knapsack problem"), o problema do depósito ("bin packing") [95], roteamento de veículos [22], etc.

Embora se acredite que os problemas considerados NP são e serão sempre intratáveis, não importa quanto a ciência de computação evolua, há um resultado interessante que diz que, se um algoritmo eficiente para um dado problema NP existir, então o algoritmo poderá ser modificado para resolver todos os outros problemas NP também [84]. Por conseguinte, é sempre interessante aplicar técnicas de otimização a problemas NP clássicos, de modo que se possa ter uma idéia do potencial de cada método para resolver problemas combinatoriais que aparecem na prática.

No caso do PVA, há um vendedor ambulante que deve visitar um conjunto de cidades, e voltar à origem cobrindo a menor distância possível. Cada cidade deve ser visitada uma e só uma vez. Um PVA com n cidades tem espaço de busca com $(n-1)!/2$ possíveis rotas, o que dá um número formidável mesmo para poucas cidades. Para $n = 30$,

há um total de $4,42 \times 10^{30}$ rotas distintas. Com um computador capaz de analisar um milhão de rotas por segundo, busca exaustiva iria levar nada menos que o equivalente a 10^7 vezes a idade do universo!

A literatura é repleta de soluções para casos com até alguns milhares de cidades [93, 96]. AGs com representação apropriada e operadores específicos têm sido aplicados a problemas de dimensões mais modestas (tipicamente problemas com umas poucas dezenas de cidades), com relativo sucesso [61, 83, 114].

7.2 Planejamento de Tarefas

Este é um campo interessante e importante na prática. O problema mais representativo é o chamado "job-shop scheduling", grosseiramente traduzido aqui como planejamento de tarefas numa fábrica. Neste problema, há uma fábrica com máquinas capazes de executar uma série de operações específicas. A fábrica recebe encomendas de produtos que consomem um determinado tempo de cada máquina, e há operações que devem ser executadas em seqüência. O problema é como planejar o uso das máquinas de modo a dar conta das encomendas no menor tempo possível, evitando também que máquinas fiquem ociosas durante o processo. Este problema tem sido resolvido por AGs com relativo sucesso [27, 32, 42, 111], embora exija uma representação cromossômica e operadores relativamente sofisticados.

7.2.1 Outros Problemas de Planejamento

Outros exemplos de uso de AGs para solução de problemas de planejamento incluem: planejamento de horários de provas numa universidade [41], planejamento de mão-de-obra [148, 149], e roteamento de veículos [21, 150]. Devido à complexidade e restrições desses problemas, geralmente é necessário desenvolver representações e operadores sofisticados que embutam conhecimento sobre o problema a ser resolvido [63].

7.3 Síntese de Redes Neurais

Entre as muitas configurações existentes para redes neurais artificiais, a arquitetura mais popular é o *perceptron multi-camadas* (PMC) ("multilayer perceptron"), devido a sua simplicidade, flexibilidade, e amplo campo de aplicação [131]. Contudo, aplicações práticas de MLP envolvem etapas altamente experimentais, com cansativos e ineficientes processos de tentativa e erro.

Primeiro, deve-se especificar a arquitetura do PMC, o número de camadas, nós (neurônios), e o padrão de conectividade entre os nós. Depois, é necessário especificar os parâmetros de aprendizagem, os pesos iniciais, e então efetuar o treino, geralmente usando um método de gradiente. Este processo é lento, depende de sorte e procedimentos heurísticos, e deve ser repetido até que o desejado grau de desempenho seja atingido. Há vários problemas óbvios com tal procedimento. Para melhorá-lo, AGs têm sido propostos com três objetivos distintos: determinação dos pesos de um PMC de configuração conhecida, determinação da configuração de um PMC para resolver um dado problema, e determinação da configuração e pesos.

7.3.1 AGs para Aprendizagem

Geralmente o treino de PMCs é realizado por algum método de gradiente, quase sempre uma variante da *regra delta*, usando o algoritmo de *propagação retrógrada* ("backpropagation") para o cálculo de derivadas parciais [131].

Embora esse método seja eficaz em muitos casos, algoritmos de gradiente somente buscam um mínimo local da superfície de erros de um PMC, e geralmente tal superfície é extremamente complexa e multimodal.

Com o objetivo de proporcionar uma busca mais global, AGs podem ser empregados para determinação dos pesos de um PMC de configuração especificada [87, 107, 154]. Nesse caso, uma população de PMCs de mesma arquitetura é usada, ou seja, cada PMC com seus pesos constitui um indivíduo. Os pesos podem ser representados por longas seqüências binárias ou números reais. A avaliação de cada indivíduo é feita com base num conjunto de padrões para teste, de modo que a adequabilidade seja uma função decrescente do erro quadrático. Testes conduzidos com problemas simples de reconhecimento de padrões produziram resultados inconclusivos. Em [87], Kitano mostrou experimentalmente que AGs são mais lentos que algoritmos rápidos para treino de PMCs, como o "quickprop" [40], e propôs usar AGs somente para busca global no início do treino, e então um método de gradiente convencional para busca local.

7.3.2 AGs para Determinação da Arquitetura

Uma outra aplicação interessante de AGs é a determinação de boas arquiteturas de PMCs para uma determinada tarefa. Em [67, 68], cada indivíduo corresponde à planta de um PMC, ou seja, o número de camadas, número de nós por camada e padrão de conectividade são representados por seqüências binárias, e a busca opera então no espaço das arquiteturas de PMCs. Além disso, os parâmetros de aprendizagem usando a regra delta também foram codificados nos cromossomos.

Não somente a representação cromossômica e os operadores genéticos se tornam relativamente complexos, mas o processo de avaliação de cada indivíduo consome muito tempo e é até questionável. Para avaliação, PMCs com pesos gerados aleatoriamente são construídos para cada indivíduo, e treinados usando-se um conjunto de padrões de treino. Depois disso, cada PMC é avaliado usando-se um outro conjunto de padrões de teste. Somente depois, a adequabilidade de cada indivíduo é estimada usando-se informação referente à velocidade de convergência, erro de reconhecimento, etc. Este esquema é ilustrado na Fig. 13.

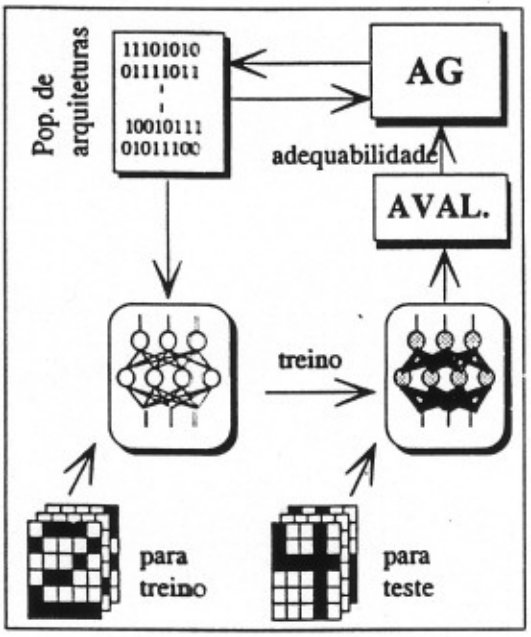

Figura 13: Determinação de arquiteturas de PMCs usando um AG. Cada indivíduo da população corresponde à planta de um PMC, e conjuntos de padrões de treino e teste são usados para avaliar cada indivíduo.

Em testes com problemas simples, boas arquiteturas foram determinadas para minimização do erro de reconhecimento, aceleração da convergência, e assim por diante. Mais interessante, o método permite a descoberta de padrões de conectividade que geralmente não são investigados em processos de tentativa e erro [67, 68]. Há uns outros poucos métodos propostos com a mesma filosofia [104].

7.3.3 AG para Síntese Total de um PMC

Numa proposta ainda mais ousada, Maniezzo propôs um AG para gerar não somente a arquitetura, mas também todos os pesos de um PMC em [99]. Neste caso, cada indivíduo da população corresponde a uma complexa seqüência binária, onde há bits para indicação da granularidade (precisão dos pesos), conectividade (se uma dada conexão existe ou não) e cada peso de um PMC que admite conexões entre camadas não adjacentes.

Combinando tradicionais operadores de recombinação e mutação, devidamente modificados de modo a sempre dar origem a PMCs viáveis, com um operador AG-simplex [17], que acentua o poder de busca global dos AGs, Maniezzo conseguiu resultados interessantes com simples problemas de aprendizado de funções lógicas e controle de um robô.

Uma de minhas pesquisas atuais visa o uso de um computador paralelo em dois níveis: 1) determinação de arquiteturas de redes neuronais recorrentes usando um AG; e 2) determinação dos pesos das melhores redes determinadas em 1) através de um procedimento híbrido combinando AGs e métodos de gradiente.

7.3.4 Mais AGs para Redes Neurais

Outras aplicações incluem, por exemplo: síntese de redes "fuzzy" [43], síntese de redes neuronais recorrentes [4], determinação de pesos de redes neuronais facilmente implementáveis em "hardware" [26], síntese de redes Booleanas (lógicas) [66], síntese de PMCs com crescimento automático [127], e determinação da topologia de uma rede neuronal de Kohonen (mapa auto-organizante [89]) [117].

7.4 Outras Aplicações de AGs

Outras aplicações incluem, por exemplo:

- Otimização de funções de um objetivo [34, 36] ou multi-objetivas [47].
- Análise de dados: "clustering" [19].
- Controle de processos: determinação de parâmetros para controle ótimo [92, 112], controle inteligente de um pêndulo invertido [113], identificação de sistemas não-lineares [81].
- Robótica: determinaçãoda trajetória de movimento de braços mecânicos [30], projeto e controle de robôs [31].
- Problemas de transporte linear e não-linear [103].
- Máquinas de estados finitos para jogos [10].
- Modelos econômicos: modelos do processo de inovação, estratégias de concorrência [3, 77].
- Modelos ecológicos: relações de simbiose, hospedeiro-parasita, etc. [73, 115, 121].
- Modelos do sistema imunológico [16].

- Modelos de sistemas sociais: evolução da cooperação [9, 10], evolução da comunicação [98, 152].
- Estudo das relações entre aprendizagem e evolução, ou o chamado "Efeito Baldwin" [2, 15, 74].

8 Dois Estudos Avançados

Abaixo apresentamos brevemente resultados de duas pesquisas deste autor envolvendo algoritmos evolucionários.

8.1 Problema de Roteamento Automático

O problema de roteamento automático (PRA) consiste em encontrar trilhas não-conflitantes unindo terminais distribuídos numa dada região. O projeto de algoritmos eficientes para PRA é extremamente importante na indústria de placas de circuito impresso e circuitos integrados [51]. Algoritmos derivados da teoria de grafos geralmente falham para complexos padrões de roteamento, ao passo que os algoritmos mais sofisticados exigem computadores paralelos poderosos ou "hardware" específico. A Fig. 14 mostra um PRA com três pares de terminais que devem ser conectados numa única camada. Problemas práticos chegam a ter centenas ou milhares de trilhas e 5 ou mais camadas.

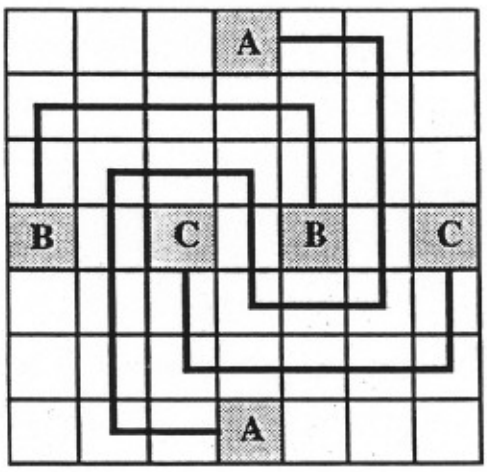

Figura 14: Exemplo de um PRA com 3 trilhas numa única camada.

PRA pertence à classe de problemas de otimização com espaços de busca altamente multimodais, onde é mais importante buscar uma solução viável em tempo prático do que a solução ótima absoluta.

A idéia proposta em [147] usa padrões de roteamento ("WP", do inglês "wiring pattern") como soluções candidatas. Cada padrão de roteamento possui uma trilha unindo cada par de terminais que devem ser conectados. No início do processo evolutivo, curtos-circuitos são permitidos, e o objetivo da evolução é melhorar a qualidade de roteamento, no sentido de eliminar os curtos-circuitos e ao mesmo tempo produzir trilhas simples, curtas, e com poucas curvas, de modo a aumentar a confiabilidade do circuito final. A Fig. 15 ilustra uma população com 5 soluções-candidatas para um pequeno PRA com 4 trilhas. Cada padrão de roteamento corresponde a uma coluna vertical da população, com todas as trilhas sobrepostas num mesmo espaço de roteamento, e curtos-circuitos são permitidos no início do processo evolutivo.

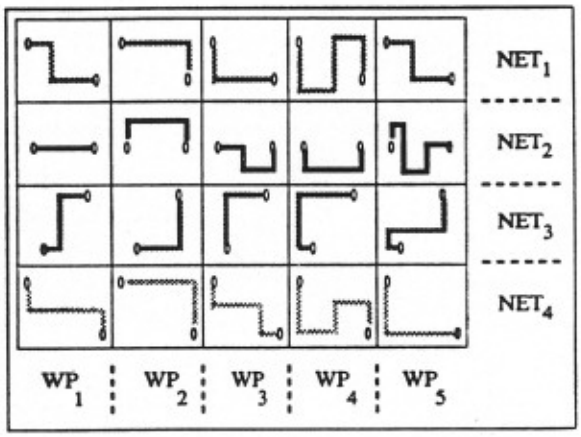

Figura 15: Exemplo de uma população de 5 soluções-candidatas para um PRA com 4 trilhas. Cada indivíduo da população corresponde a uma coluna do arranjo acima, com todas as trilhas superpostas num mesmo espaço de roteamento.

O problema foi formulado em termos de custos, dando custos altos a padrões com muitos curtos-circuitos, trilhas complexas, etc. O processo de seleção usa esses valores, dando mais cópias a padrões de roteamento de baixo custo. Recombinação ocorre probabilisticamente e somente trilhas conflitantes (com curtos-circuitos) são trocadas. Mais adiante, os custos de cada trilha são empregados por um operador sofisticado de mutação. Primeiro, baseado no custo de cada trilha, decide-se se uma determinada trilha deve ou não ser refeita. Em caso afirmativo, a trilha é apagada e retraçada usando um método de roteamento conhecido.

O método proposto teve sucesso em obter padrões de roteamento de alta qualidade, sem curtos-circuitos, para vários problemas com uma ou duas camadas para roteamento. Para um PRA com 100 trilhas em 2 camadas, exemplos de resultados obtidos a partir de 200 gerações usando uma população de 20 indivíduos são mostrados na Fig. 16.

O principal entrave à aplicação do método é o tempo de processamento, que às vezes leva horas

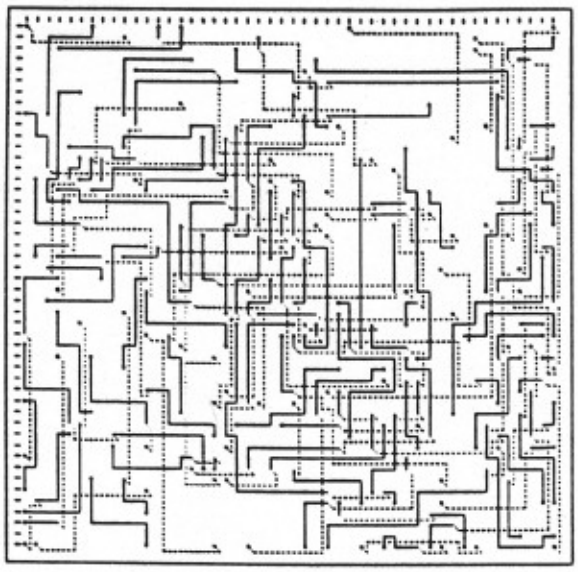

Figura 16: Exemplo de resultados sem curtos-circuitos obtidos por um AG para um PRA de 100 trilhas e duas camadas.

em estações de trabalho. Atualmente estamos estudando representações mais eficientes, aplicação de uma estratégia "dividir-para-conquistar", e implementação em um computador massivamente paralelo.

8.2 AGs para Funções Não-Estacionárias

Uma outra pesquisa considera a aplicação de AGs para a otimização de funções não-estacionárias, ou seja, ambientes que variam com o tempo [145]. Tais problemas ocorrem tanto na natureza (por exemplo, variações climáticas podem fazer com que indivíduos outrora adequados ao ambiente tenham menor chance de sobrevivência do que outros indivíduos até então não tão adequados) quanto na indústria (por exemplo, no controle de processos químicos ou em aeronaves, onde a dinâmica dos aparelhos varia violentamente com o consumo de combustível).

Com ambientes altamente dinâmicos, o fenômeno das colinas de Hamming é enfatizado, especialmente com códigos binários posicionais. Para demonstrar isso, considere a otimização de uma simples função de uma variável

$$f(x) = \frac{1}{\sqrt{x - 64 + 1}} \qquad (16)$$

no domínio discreto $x \in \aleph | 0 \leq x \leq 127$. É óbvio que o máximo desta função ocorre para $x^* = 64$

Para tal função, codificar cada ponto do espaço de busca em uma seqüência binária posicional de 7 bits parece ser uma escolha natural. No espaço genotípico, o máximo então corresponderia à seqüência $s^* = 1000000$.

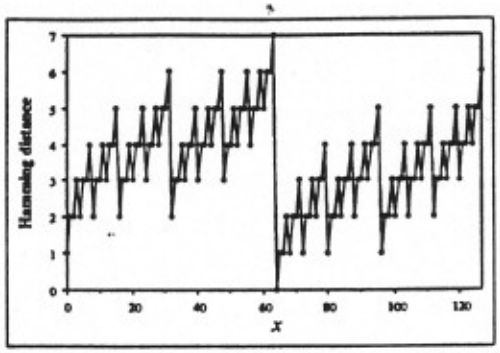

Figura 17: Distância de Hamming entre cada indivíduo e o ponto ótimo $x^* = 64$ usando código binário posicional.

A Fig. 17 mostra a distância de Hamming (número de bits diferentes em cada posição) entre a seqüência correspondente a cada indivíduo do espaço de busca e a seqüência correspondente ao ótimo. Observa-se que, no espaço de busca genotípico, há uma considerável distância (7 bits) entre o indivíduo $x = 63$ e x^*, muito embora os pontos estejam bem próximos no espaço de busca original. Ou seja, a codificação binária resultou no aparecimento de colinas de Hamming entre as seqüências correspondentes a indivíduos vizinhos. Por conseguinte, pontos $x \leq 63$ têm dificuldade em avançar em direção ao ótimo, o que não ocorre tão seriamente para os pontos $x \geq 65$.

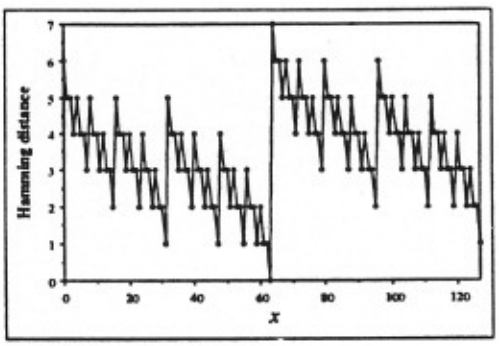

Figura 18: Distância de Hamming entre cada indivíduo e o ponto ótimo após variação ambiental ($x^* = 63$) usando código binário posicional.

Suponhamos agora que haja alguma alteração ambiental que mova o ótimo para $x^* = 63$. As novas distâncias de Hamming são ilustradas na Fig. 18. Como conseqüência de uma pequena mudança no ambiente, houve o aparecimento de uma colina de Hamming entre o cromossomo 1000000, até então o ótimo, e o novo pico $s^* = 0111111$.

Códigos de Gray minimizam o efeito das colinas de Hamming, conforme pode ser observado na Fig. 19, mas mesmo assim se observa uma assimetria no espaço genotípico.

De modo a evitar as colinas de Hamming e tornar o processo de busca mais suave, em [145]

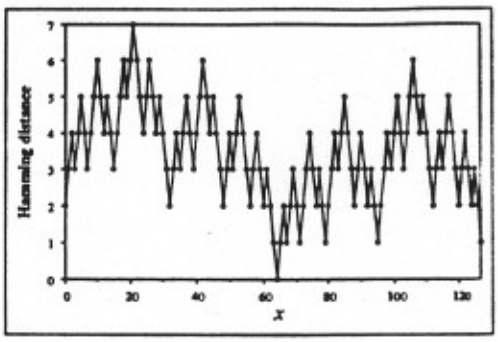

Figura 19: Distância de Hamming entre cada indivíduo e o ponto ótimo $x^* = 64$ usando um código de Gray.

propõe-se uma representação em que cada indivíduo corresponde a um vetor de componentes reais. Novos operadores de recombinação e mutação foram definidos, o primeiro de modo a efetuar interpolação-extrapolação na vizinhança dos cromossomos pais, o segundo de modo a globalizar o processo de busca.

Para exemplificar alguns dos resultados obtidos, consideremos a maximização da seguinte função de n variáveis reais para $-100 \leq x_i \leq 100$, $i = 1, 2, \ldots, n$:

$$f(\boldsymbol{x}) = \frac{100}{n} \sum_{i=1}^{n} \left(\frac{\cos(0,15\pi(x_i^* - x_i)) + 1}{2 + 0,0025(x_i^* - x_i)^2} \right). \tag{17}$$

É fácil perceber que $f(\cdot)$ é uma função altamente multimodal com um único pico de valor 100 em $\boldsymbol{x} = [x_1^* x_2^* \ldots x_n^*]$. Para uma variável, a função $f(\cdot)$ é ilustrada na Fig. 20, donde se vê que há 15 máximos locais por dimensão. Para $n = 10$ há, portanto, nada menos que $15^{10} \approx 5,7 \times 10^{11}$ máximos locais e um único máximo global, fazendo de $f(\cdot)$ uma função extremamente difícil de ser otimizada por métodos convencionais.

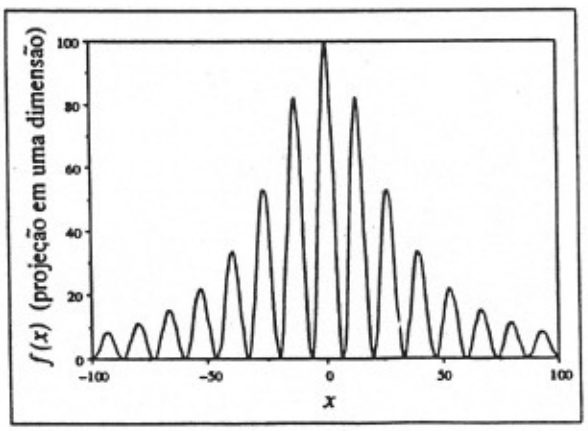

Figura 20: Função a ser otimizada em uma dimensão.

Para tornar as coisas ainda mais difíceis, adicionou-se a $f(\cdot)$ um aspecto não-estacionário, de modo que o sistema de coordenadas gira em círculosà medida que o tempo passa. Para $n = 10$ e com o sistema de coordenadas efetuando uma rotação a cada 200 gerações, alguns resultados são ilustrados na Fig. 21, onde "AG 1" corresponde ao AG proposto com indivíduos representados por vetores de números reais, enquanto que "AG 2" e "AG 3" correspondem aos AGs com códigos binário posicional e o código de Gray mais comum, respectivamente. As curvas indicam os melhores resultados de cada geração, e é evidente que o sistema proposto apresenta melhor desempenho que os demais.

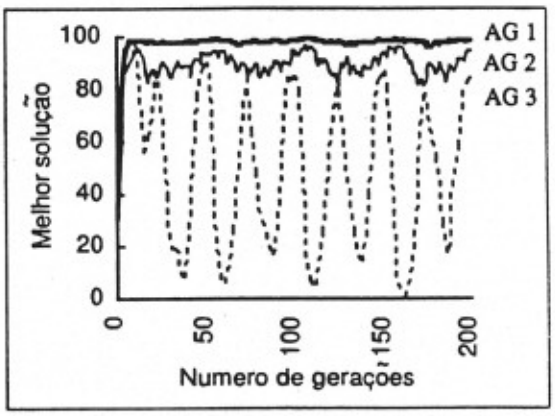

Figura 21: Resultados obtidos usando três tipos de AGs para a otimização de uma função multimodal altamente dinâmica.

9 Avanços Teóricos

Como grande parte dos processos não-determinísticos, a análise matemática dos AGs é extremamente difícil. Com AGs, o dito anônimo *"Teoria é quando se sabe tudo e nada funciona, enquanto que prática é quando tudo funciona, mas não se sabe por quê"* é corretíssimo.

9.1 Teoria dos Esquemas

Os primeiros esforços para entender o poder computacional dos AGs e esboçar uma análise matemática foram realizados por Holland [75], dando origem ao conceito de "esquema" ("schema", com plural "schemata", nos textos em inglês). Um esquema poderia ser pensado como um molde (em inglês, "template") representando um ou um conjunto de cromossomos. Considerando o alfabeto binário e cromossomos de comprimento ℓ, um esquema seria qualquer seqüência de ℓ caracteres gerados a partir do alfabeto extendido $\{0, 1, *\}$, onde '*' representa um "coringa", podendo ser ou 1 ou 0.

Um resultado importante é o chamado *Teorema dos Esquemas*, também chamado pomposa-

mente de *Teorema Fundamental dos Algoritmos Genéticos* [75, 56]. Nesse teorema, considerando o AG canônico, Holland demonstrou que esquemas com adequabilidade superior à média recebem um número exponencialmente crescente de representantes na próxima geração. Em sua eloqüência, Holland usou uma analogia com uma máquina de cassino com dois braços em as probabilidades de se ganhar são distintas para cada braço mas, obviamente, o jogador não tem conhecimento prévio de qual é o braço mais vantajoso ("two-armed bandit problem"). A Fig. 22 ilustra este problema para o caso em que o braço esquerdo é o mais vantajoso. Holland sugeriu que AGs automaticamente escolhem a melhor estratégia para se ganhar nesse jogo, alocando tentativas de modo exponencial para o braço com maior lucro observado.

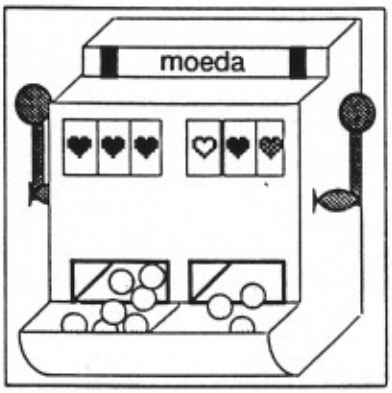

Figura 22: Problema da máquina de cassino de dois braços. Neste caso, o braço esquerdo é o mais vantajoso.

9.2 Paralelismo Implícito

Em sua tentativa de desvendar características dos AGs, Holland e seu discípulo favorito, Goldberg, tentaram estimar quantos esquemas seriam processados a cada geração e chegaram a um resultado surpreendente: Numa população de N indivíduos, a cada geração há um número de esquemas proporcional a N^3. Esse fenômeno, em que AGs processam um enorme número de esquemas mesmo com poucos cromossomos, foi denominado de *paralelismo implícito* [75].

Contudo, como no caso do teorema dos esquemas, as conclusões somente são válidas sob hipóteses muito fortes, incluindo população infinita e altamente diversificada. Tais condições não são possíveis na prática. De fato, Grefenstette e Baker mostraram que o mágico N^3 somente é válido, quando muito, nas primeiras gerações[64].

Além disso, enquanto que a análise de esquemas sugere que alfabetos de baixa cardinalidade devem ser escolhidos sempre que possível [56], resultados experimentais têm mostrado que alfabetos binários não são representações nem naturais nem eficazes em muitos problemas práticos [103]. Conseqüentemente, acredito que os esquemas não mereçam o destaque que têm recebido na literatura, e sejam mais um exercício fútil de matemática do que algo que venha realmente a iluminar o poder de processamento dos AGs.

9.3 Análises de Cadeias de Markov

Um grupo de pesquisadores [38, 54, 130] tem tentado compreender os mecanismos internos de AGs usando cadeias finitas de Markov , que descrevem trajetórias probabilísticas sobre um conjunto finito de estados [82, 86]. Em [130], Rudolph demonstra teoremas interessantes sobre os AGs mais básicos, com seleção baseada na adequabilidade, recombinação e mutação aleatórias, que ele denominou de AG canônico.

Resultados teóricos interessantes foram obtidos através da análise de cadeias finitas e homogêneas de Markov para $0 < p_{\text{mut}} < 1$ e $0 \leq p_{\text{rec}} \leq 1$. Primeiro, foi demonstrado que um AG canônico jamais converge para o ótimo global de uma função independentemente do método de inicialização, parâmetros, etc.! Para o alívio de muitos, contudo, Rudolph também demonstrou que, se elitismo for empregado antes ou depois da seleção, obtém-se convergência para o ótimo global, embora isso possa levar muitas gerações.

9.4 Complexidade Computacional

O conceito de razão de adequabilidade foi introduzido como o quociente entre a média da adequabilidade dos cromossomos com um dado alelo pela valor médio de adequabilidade de todos os cromossomos da população. Em [5], Ankenbrandt partiu de um AG simples e demonstrou que, se a razão de adequabilidade, r, for constante e conhecida, a complexidade computacional do AG obecerá a:

$$O(\text{AG}) \propto O(a(\cdot))\frac{N \ln N}{\ln r}, \qquad (18)$$

onde $O(\text{AG})$ indica a complexidade computacional do AG simples com seleção proporcional à adequabilidade, $O(a(\cdot))$ é a complexidade da função de avaliação, que depende do problema específico, e N é o número de indivíduos da população. Embora esse resultado possa servir para dar uma idéia da relação entre o tempo que um AG leva até a convergência e o tamanho da população, na prática é extremamente improvável que a razão de adequabilidade seja constante.

Em [57] vários métodos de seleção foram comparados em termos de complexidade computacional. Para uma população de tamanho N, os resultados obtidos aparecem na Tabela 4.

Método de seleção	Complexidade
roleta	$O(N^2)$
resto estocástico	$O(N)$
graduação	$O(N \ln N)+ O(\text{seleção})$
torneio	$O(n)$

Tabela 4: Complexidade computacional de métodos de seleção.

9.5 Problemas Difíceis para AGs

Métodos de geração de problemas enganadores têm sido pesquisados [35]. Além disso, Grefenstette demonstrou que o fato de um problema ser enganador não é condição nem suficiente nem necessária para o problema ser difícil para AGs (os chamados "GA-hard problems") [65].

10 Genética Natural e em AGs

Sendo um modelo tão simples de evolução, uma pergunta surge com freqüência: Até que ponto os AGs emulam o que se passa com os seres vivos deste planeta?

10.1 Cromossomos, Genes e ADN

Na natureza, seres vivos superiores guardam sua bagagem genética em cromossomos, que são filamentos alongados de proteína e ADN (ácido desoxi-ribonucleico). O número de cromossomos varia de espécie para espécie. Por sua vez, cada cromossomo possui um grande número de genes, cada qual consistindo de cadeias de bases nucleotídicas ou nucleicas chamadas Adenina (A), Citosina (C), Timina (T) e Guanina G). As moléculas de ADN têm formato de uma dupla hélice, algo similar a uma escada em espiral. As bases nucleotídicas, sempre aos pares (A-T, C-G), formam os degraus da escada.

Em última análise, são as bases nucleotídicas, e não os genes, que constituem os *quanta* genéticos, que podem ser imaginados como elementos de um alfabeto de cardinalidade 4. Em AGs simples, contudo, a análise pára nos genes, ou seja, os elementos de cada seqüência cromossômica.

10.2 Haploidismo e Diploidismo

Além disso, as células somáticas, que formam a maior parte das células dos seres vivos superiores, são diplóides, isto é, têm todos os cromossomos em pares, ao passo que as células sexuais ou gaméticas são haplóides, ou seja, não têm cromossomos aos pares. Nas células somáticas dos seres humanos normais há 23 pares de cromossomos, ao passo que nos gametas, a saber, espermatozóides e óvulos, há somente 23 cromossomos.

Na maior parte dos AGs, contudo, assume-se que os indivíduos (seres vivos da população) sejam haplóides e tenham bagagem genética composta por um único cromossomo[7] de tamanho fixo.

10.3 Espaço de Busca

Cada cromossomo humano é constituído por aproximadamente 50.000 genes relacionados através de uma rede de interacões inimaginavelmente complexa. Cada gene humano consiste de cerca de 300 tripletas de bases nucleotídicas, cada qual codificando um entre 20 tipos de aminoácidos, que formam as nossa proteínas. Assim sendo, desta análise grosseira pode haver um número de $20^{15.000.000}$ distintos genótipos, ou seja, aproximadamente $10^{19.500.000}$, um número tão grande que nem aparece em livros de Astronomia.

Na maior parte dos AGs, o espaço de busca genética é infinitamente mais modesto. Há relatos de grupos usando AGs com problemas de milhares de variáveis, cada qual codificada como uma seqüência de vários bits. Mesmo para um problema de, digamos, 1.000 variáveis, cada uma representada por uma seqüência de comprimento $\ell = 100$, haveria *somente* um total de $2^{100.000} \approx 10^{30.000}$ genótipos possíveis. Quando se considera que a idade do universo é estimada em 5×10^{17} segundos [71], fica claro que, embora o espaço de busca de AGs seja extremamente menor que o da evolução natural, ainda assim representa um desafio formidável.

10.4 Mapeamento Genótipo → Fenótipo

Num ser vivo, a estrutura dos cromossomos e genes de um indivíduo formam o seu *genótipo*, enquanto que as características do indivíduo formam o *fenótipo*. Para seres superiores na escala de evolução do planeta Terra, o mapeamento genótipo → fenótipo é um mistério que deverá permanecer sem solução por muito tempo, talvez para sempre. Mesmo os maiores entusiastas do ambicioso *Projeto Genoma* [20], levado a cabo por cientistas de todo o mundo e que dentro de alguns anos deverá ter catalogado todos os genes comuns aos seres humanos, sabem que desvendar as características fundamentais de um ser vivo a partir do seu mapa genético é assunto para escritores de ficção científica. Os humanos têm um total de 10^5 genes [20] e mais um número indeterminado de genes reguladores, que guardam segredos de mais de três bilhões de anos de evolução da vida no planeta, e determinam características de cada uma das 50 trilhões de células de um ser humano adulto,

[7] Há trabalhos considerando indivíduos multicromossômicos e diplóides, mas não são considerados neste trabalho.

incluindo as 10^{15} sinapses do sistema nervoso central [125] [8].

Enfim, na genética natural as relações de causa-efeito e associações são extremamente complexas, e os mecanismos de interação que *ligam* e *desligam* os genes e determinam desde a cor dos olhos até padrões de envelhecimento e predisposição a certos tipos de câncer, talvez nunca venham a ser totalmente esclarecidos.

Essas dificuldades são ainda exacerbadas pelos fenômenos de *pleiotropia* e *poligenia* [101]. No primeiro caso, um gene pode afetar várias características fenotípicas ao mesmo tempo, ao passo que no segundo caso, uma característica fenotípica pode ser o resultado da ação conjunta de multíplos genes, gerando complexos mapeamentos "1 → muitos", "muitos → 1".

Já no caso dos AGs, contudo, o usuário determina o mapeamento genótipo → fenótipo no momento em que especifica o método de representação cromossômica, condição *sine qua non* para o uso de um AG. Dado um fenótipo, a representação genotípica correspondente é, em geral, obtida facilmente. No sentido oposto o mesmo também é válido, muito embora haja casos de representações cromossômicas que não correspondem a nenhuma solução válida do espaço de busca original.

O mapeamento genótipo → fenótipo é ilustrado na Fig. 23(a), que também mostra casos de pleiotropia e poligenia com relações desconhecidas. Em contrapartida, a Fig. 23(b) mostra o equivalente em AGs, usando um caso típico de três parâmetros mapeados numa seqüência de 15 bits.

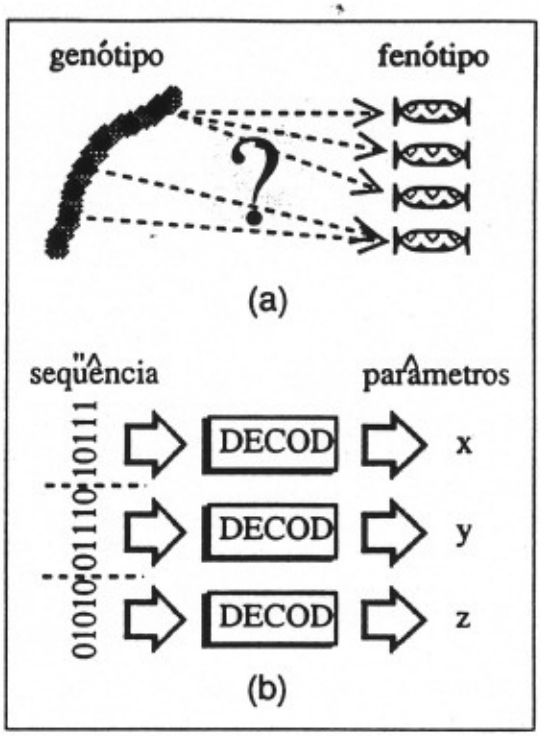

Figura 23: Mapeamento genótipo → fenótipo: (a) Na natureza, com complexas relações de pleiotropia e poligenia; (b) em AGs convencionais, há um mapeamento direto de segmentos da seqüência cromossômica para parâmetros que especificam um ponto do espaço de busca.

10.5 Seleção em AGs e na Natureza

Os processos seletivos que ocorrem em AGs e outros algoritmos evolucionários não passam de modelos altamente simplificados da seleção natural. No caso humano, por exemplo, a seleção de parceiros envolve uma miríade de fatores sociais, econômicos, culturais, etc., em populações interativas de tamanhos variáveis.

Há outros níveis de seleção também. Por exemplo, em 3 ou 4 mililitros de sêmen pode haver até meio bilhão de espermatozóides [7], cada qual com uma bagagem genética um pouco diferente, e em geral há somente um óvulo a ser fertilizado. É um "vestibular" extremamente concorrido! Além disso, indivíduos com exatamente a mesma bagagem genética, isto é, os *gêmeos*, são raros em seres superiores. Estatísticas indicam que, em nascimentos naturais, sem o uso de drogas de fertilização, gêmeos ocorrem a cada 85 nascimentos, trigêmeos a cada 7.500, quadrigêmeos a cada 650.000 e gêmeos quíntuplos a cada 57.000.000 de nascimentos [7].

A seleção em AGs, realizada com o uso de uma função objetivo e uma roleta, até parece ridícula. De modo distinto da natureza, duplicatas (gêmeos) são muito comuns na maior parte dos modelos usados, principalmente à medida que as gerações avançam. Todavia, mesmo simplificado ao extremo, a evolução em AGs funciona e tem inúmeras aplicações práticas em engenharia.

10.6 Operadores Genéticos

Enquanto que em seres vivos a probabilidade de mutações varia de gene para gene e ainda sofre grande influência de fatores externos, em grande parte dos AGs a probabilidade é especificada pelo usuário, e mutações são simuladas de modo extremamente prosaico. Os mecanismos biológicos que regem a troca de partes entre cromossomos homólogos também não são totalmente conhecidos, embora na maior parte dos AGs trocas aleatórias sejam geralmente empregadas.

Além disso, a ênfase que AGs costumam colocar na recombinação é ainda assunto de discussão nos meios acadêmicos. Holland vê a re-

[8] É óbvio que muitas ou todas as características são afetadas por fatores internos e externos. Isso é particularmente verdadeiro no caso do desenvolvimento do cérebro, que necessita de estímulos externos para se desenvolver plenamente [125, 135].

combinação como uma operação de fundamental relevância, e conseqüentemente os pesquisadores de AGs têm feito peripécias para implementar recombinação com cromossomos sofisticados, onde muitas vezes o uso de recombinação convencional resulta em indivíduos sem sentido [103]. Adeptos de outros paradigmas evolucionários, notadamente as estratégias evolucionárias [123, 124] consideram a mutação o único operador digno de ser implementado.

10.7 Escalas de Tempo

Acredita-se que o planeta em que vivemos tenha surgido há cerca de 3,8 bilhões de anos atrás. Os primeiros sinais de vida surgiram de 300 a 600 milhões de anos depois. Entre 1,8 e 1,3 bilhão de anos atrás, a atmosfera terrestre foi "contaminada" por oxigênio, e logo depois surgiram plantas e seres aeróbicos. Sexo teria aparecido mais tarde, cerca de 900 a 700 milhões de anos atrás, dando um ímpeto ao processo evolutivo. Os primeiros animais apareceram há 700 milhões de anos atrás. Seguiram-se os primeiros vertebrados (500 milhões de anos), os insetos (420 milhões), os mamíferos (200 milhões), os primeiros primatas (70 milhões), até os primeiros humanos, *Homo erectus*, que teriam sido chineses que apareceram entre 1,8 e 1,7 milhão de anos atrás [44]. Conclui-se, portanto, que cerca de 100.000 gerações separam o homem atual dos primeiros humanos.

Com AGs, uma geração pode levar de uns poucos milisegundos a várias horas, dependendo do tamanho da população, cromossomos, tempo necessário para medir a adequabilidade de um cromossomo, etc.

10.8 O que Darwin Não Sabia

Cientistas de computação envolvidos em pesquisas em CE costumam dizer que seus modelos são firmemente baseados na teoria Darwiniana de evolução. De fato, Darwin propôs uma série de teorias, e o conceito de *Darwinismo* tem variado de época para época, de pessoa para pessoa [102]. No caso geral, pessoas associam Darwinismo como uma doutrina em que os seres vivos evoluem gradualmente, produzindo proles geneticamente diversas, que competem pela sobrevivência num processo denominado *seleção natural*, que seria então o principal motor da evolução.

As teorias evolutivas deste século concordam com a importância da seleção natural, mas há evidências de que esse processo por si só não é suficiente para explicar os caminhos da evolução neste planeta. Uma das teses recentes é a de que a evolução não ocorre gradualmente, mas sim aos saltos. Outra teoria enfatiza a importância de processos puramente aleatórios para explicar a distribuição genética de muitas populações, numa teoria denominada de *Deriva Genética* (em inglês, "genetic drift") [142]. Juntamente com seleção natural, deriva genética parece ser outro importante fator para explicar os processos evolutivos dos seres vivos.

11 Árvore Genealógica dos AGs

11.1 Moderna Inteligência Artificial

Uma vez que não existe um consenso sobre o que é inteligência, não é possível encontrar uma definição precisa para *inteligência artificial*. No entanto, o termo "AI" geralmente se refere à inteligência artificial tradicional ou clássica, baseada na manipulação serial de símbolos [13]. Em contraste, há hoje em dia uma forte tendência dos cientistas de computação de propor e estudar métodos numéricos vagamente inspirados na natureza, como forma de realizar tarefas onde "inteligência" seja necessária.

Esses métodos e modelos inspirados na natureza formam uma nova área da ciência denominada por Bezdek de *Inteligência Computacional* (IC) [18][9] que inclui num grande leque as redes neurais artificiais [6, 70], os sistemas "fuzzy" ou de lógica nebulosa[10] [129, 157], e a computação evolucionária, dos quais os AGs são o mais conhecido representante.

Segue-se um resumo da minha visão da "família dos AGs", ou seja, quais são as novas áreas de computação relacionadas aos AGs, e quão afins são tais relações.

11.2 Parentes Afins

As três grandes áreas englobadas pela IC são inspiradas em distintos aspectos da inteligência. Redes neurais artificiais são, pelo menos a princípio, modelos extremamente simples de sistemas nervosos, e buscam a inteligência numa abordagem "bottom-up", do simples para o complexo. Sistemas de lógica nebulosa são modelos que visam a ganhar um pouco da nuança do raciocínio humano através de uma lógica flexível. Finalmente, sistemas de CE encaram o fenômeno de evolução como um processo inteligente de otimização.

Atualmente tem havido um considerável esforço para integrar as áreas de IC. Ver, por exemplo, [49, 134]. Cada área tem suas características,

[9] O termo "Inteligência Computacional" ainda não está muito difundido, e algumas conferências preferem o termo "Computação Flexível" ("Soft Computing").

[10] A tradução de "fuzzy systems" para "sistemas nebulosos" não me parece boa. Os termos "systemas de raciocínio aproximado" ou "sistemas de lógica flexível" são, na opinião deste autor, mais appropriados.

pontos fortes e fracos, e saber explorar o melhor de cada método é uma área ainda aberta para pesquisas futuras.

11.3 Descendentes Diretos

11.3.1 Sistemas Classificadores

Da mesma forma que os AGs, os Sistemas Classificadores (SCs) são fruto das idéias de John Holland. Em poucas palavras, SCs são basicamente sistemas de produções (regras) adaptativos, em que regras do tipo "if-then", evoluem de acordo com um AG [14, 23, 156]. Cada regra é denominada um classificador, e é representada por uma seqüência gerada a partir do alfabeto $A = \{0, 1, \#\}$, onde '#' age como um coringa. Populações de classificadores são manipuladas por AGs, resultando num interessante paradigma computacional.

Há uma variedade de SCs capazes de aprendizagem, ou seja, manipulação de, ou mesmo criação de novas regras através de "reinforcement learning" [76]. SCs constituem uma área com grandes avenidas para pesquisa.

11.3.2 Programação Genética

O campo da Programação Genética (PG), capitaneado por John Koza e James Rice, de Stanford, aplica as idéias de AGs a estruturas muito mais complexas que seqüências binárias [90]. Cada cromossomo representa uma árvore computacional de tamanho variável, em contraste com os cromossomos de tamanho fixo da grande maioria dos AGs.

Mais ainda, cada estrutura cromossômica em PG representa um programa, geralmente uma expressão-S da linguagem LISP. LISP é uma escolha natural dos adeptos de PG porque a sintaxe de LISP é extremamente simples e uniforme, e essencialmente cada programa em LISP pode ser representado por uma estrutura em árvore, sem distinções entre código e dados. Por exemplo, consideremos a seguinte expressão-S em LISP:

```
(IF (AND x (NOT (PLUSP (- (* (* x x) y) z))))
    (SETF x (+ x z))
    (SETF x (SIN (* (* (/ x (+ x x)) x) x))))
```

A expressão acima pode ser facilmente representada em árvore, conforme ilustrado na Fig. 24. Em PG, cada árvore representa um programa de computador e é equivalente a um indivíduo de uma população que evolui de acordo com um AG.

Em linguagem mais acessível, a árvore da Fig. 24 é equivalente a:

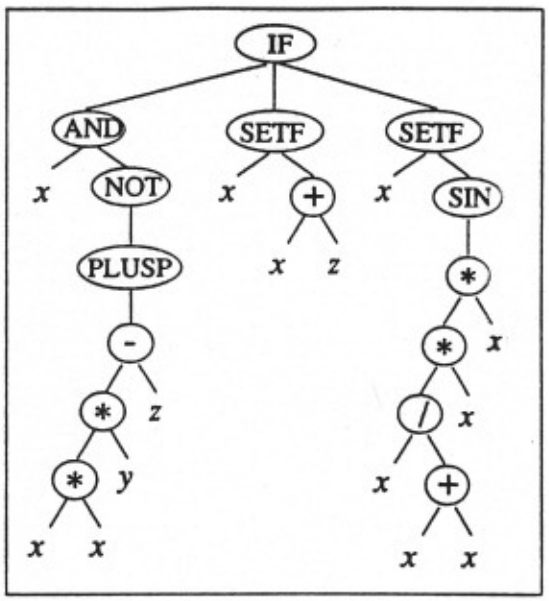

Figura 24: Exemplo de um programa de computador representado em árvore. Em PG, cada árvore equivale a um indivíduo, e uma população de programas evolue de modo análogo aos AGs.

```
IF (x ≠ 0) AND (x²y − z > 0)
    THEN x ← x + z
    ELSE x ← sin(x²/2)
```

Assim sendo, PG realiza a geração automática de programas de computador para resolver um dado problema. Embora os modelos atuais de PG ainda sejam vagarosos e ineficientes, PG tem sido aplicada a uma enorme gama de problemas incluindo regressão simbólica, reconhecimento de padrões, identificação e controle de processos, e geração automática de máquinas de estados finitos [90, 91].

11.4 Primos Próximos

Nos anos 60, outros paradigmas computacionais inspirados em modelos simplificados de evolução surgiram independentemente dos AGs, e esse métodos só recentemente começaram a interagir [46].

11.4.1 Estratégias Evolucionárias

Estratégias evolucionárias (EEs) tiveram origem na Alemanha com Rechenberg e Schwefel [123, 124, 136]. Enquanto Holland concebeu os AGs como modelos simplificados da evolução natural, os então estudantes alemães queriam basicamente desenvolver um método eficiente de otimização de funções reais multimodais e não-diferenciáveis.

No esquema mais simples de EE, o chamado EE-(1 + 1), um indivíduo-pai gera um só fi-

lho através de aplicação de mutações de distribuição Gaussiana, média zero e variança variável, de modo que pequenas mutações ocorram mais freqüentemente que mutações mais radicais. Sempre que um filho "melhor" que o pai é gerado, o pai é substituído e o processo é reiniciado. Para essa estrutura simples, chegou-se ao resultado teórico conhecido com *regra do 1/5 de sucesso*, que diz que aproximadamente 20% das mutações produzem indivíduos melhores que os pais. Outros modelos de EE usando recombinação e m indivíduos-pais são usados atualmente, e esses modelos são conhecidos como estratégias $(m+1)$.

11.4.2 Programação Evolucionária

Os métodos de Programação Evolucionária (PE) foram originalmente desenvolvidos por Fogel [45]. Tipicamente, em PE há uma população de N indivíduos que são copiados na totalidade numa população temporária e sofrem mutações variáveis. Um torneio estocástico é então realizado para extrair a seguinte população desse grupo de $2N$ indivíduos. Não há nenhuma restrição formal que implique que o tamanho da população deva ser constante, e não há recombinação.

11.5 Primos Distantes

11.5.1 Vida Artificial

Vida Artificial (VA) é um novo ramo de pesquisas que lida com "biologia sintética", ou seja, formas de vida artificiais, diferentes das estruturas de carbono observadas neste planeta. Embora não esteja diretamente ligado com AGs, o campo de VA usa também uma metáfora evolucionária [106, 121]. Porém, no caso de VA, seres abstratos (geralmente programas de computador) evoluem em ambientes cibernéticos. Por exemplo, no sistema mais conhecido, o "Tierra", programas de computador evoluem e competem por tempo de CPU e memória central [122]. Sistemas de VA com aplicações de AGs são sumarizados em [106].

11.5.2 Automata Celulares e Computação Emergente

Um *automaton celular* (AC) (plural = *automata celulares*) é uma máquina de estados finitos cercada por outros automata celulares idênticos. Cada AC é um sistema dinâmico discreto, onde tempo, espaço e o número de estados são finitos. Numa malha regular, geralmente de uma ou duas dimensões, cada ponto é denominado uma *célula* e pode assumir um dentre um conjunto finito de estados possíveis. Todas as células mudam de estado em sincronia e em tempos discretos, de acordo com regras de transição locais, o estado da própria célula e os estados das células vizinhas [25, 151].

As regras de transição podem ser determinísticas ou estocásticas. Como dito anteriormente, AGs Celulares podem ser vistos como casos particulares de AC estocásticos [155]. É particularmente interessante que ACs com simples regras podem dar origem a sistemas complexos que exibem comportamentos emergentes [25, 48, 151].

11.6 Um "Mar" de Siglas

A Fig. 25 ilustra a grande família de métodos computacionais à qual os AGs pertencem. Na figura, as seguinte siglas são empregadas: VA (vida artificial), GF (geometria de fractais), AC (automata celulares), CSC (ciências de sistemas complexos), IC (inteligência computacional), SLF (sistemas de lógica flexível), RNA (redes neuronais artificiais), CE (computação evolucionária), AG (algoritmos genéticos), PG (programação genética), SC (sistemas classificadores), EE (estratégias evolucionárias), PE (programação evolucionária), e AE (algoritmos evolucionários).

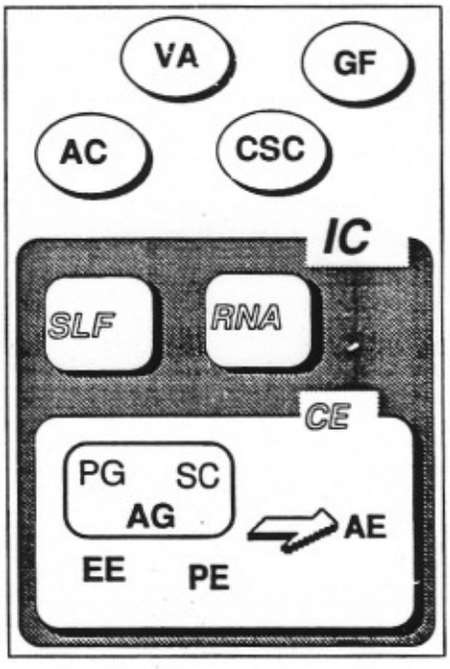

Figura 25: Novos ramos das ciências da computação e suas relações com os AGs.

12 Discussão

12.1 Quando Usar AGs?

Goldberg enunciou 4 razões pelas quais AGs podem ser atraentes para aplicações práticas [56].

- AGs podem resolver problemas difíceis rápida e confiavelmente.

- A construção de interfaces entre AGs e modelos existentes é geralmente simples.
- AGs são extensíveis.
- AGs são fáceis de se "hibridar" (combinar com outros métodos).

Os AGs são extremamente versáteis e requerem pouco conhecimento sobre a função a ser otimizada. Em geral, AGs rapidamente descobrem subregiões de alta qualidade em vastos espaços de busca, mas depois demoram a convergir.

Longe de ser uma panacéia, os AGs são mais uma ferramente à disposição do engenheiro moderno, que pode ou não usá-la dependendo do problema a ser enfrentado. Se houver bom conhecimento a respeito da função a ser otimizada, métodos numéricos derivados da análise matemática ou inteligência artificial clássica *quase sempre* serão mais eficazes que AGs. Porém, para funções desconhecidas, descontínuas e não diferenciáveis, AGs estão entre os melhores métodos conhecidos.

12.2 Um Procedimento Prático

Suponhamos que haja um certo problema de otimização a resolver, e que não haja condições ou informação suficientes para que se apliquem métodos convencionais de otimização. Em casos práticos, a limitação fundamental é tempo e, em geral, a computação da função objetivo toma a maior parte do tempo de execução. Suponhamos que o tempo de processamento aceitável seja suficiente somente para a avaliação de, digamos, 10.000 indivíduos (ou seja, amostra de 10.000 pontos do espaço de busca). Nesse caso, aconselho o seguinte procedimento prático:

1. Use toda a informação disponível para definir uma representação cromossômica amistosa para processamento por AGs, no sentido de que operadores possam ser definidos sem muita dificuldade.
2. Defina uma função de adequabilidade que realmente expresse todas as características importantes do problema.
3. Construa operadores probabilísticos que tenham, na média, tendência a melhorar as soluções-candidatas.
4. Defina uma população inicial e execute o AG, usando um pouco mais de metade do tempo de processamento disponível (neste caso, por exemplo, 6.000 cálculos de adequabilidade) e sempre memorizando os melhores resultados obtidos.
5. Use o restante do tempo de processamento (4.000 pontos, neste exemplo) para busca numérica por um método convencional a partir dos melhores pontos obtidos pelo AG.

Em suma, usa-se o AG para filtrar um vasto espaço de busca, deixando apenas regiões promissoras, e depois emprega-se um método convencional para efetuar busca local.

12.3 Sistemas Híbridos usando AGs

A partir da constatação de que nenhum método de otimização é perfeito, vários sistemas híbridos têm sido propostos. Uma técnica chamada de *interdigitação*, combinando AGs, sistemas especialistas, e métodos numéricos, foi proposta em [118]. Nessa técnica, o uso de cada método depende do conhecimento que o usuário tem sobre as relações causais e sobre o processo a ser otimizado. Os autores mostraram que, para o projeto de uma turbina de avião, interdigitação produziu melhores resultados que cada uma das técnicas de otimização isoladas.

12.4 Áreas Abertas para Pesquisa

Entre as muitas avenidas abertas para a pesquisa em AGs, poderíamos citar:

- Aspectos teóricos dos AGs: velocidade e condições de convergência, intervalos ideais para parâmetros.
- Estudo da importância dos operadores genéticos biológicos e os equivalentes em AGs [53].
- Modelos que permitam o uso de AGs a sistemas em tempo real.
- Modelos híbridos de uso geral usando redes neuronais e/ou conjuntos de lógica flexível.
- Desenvolvimento de aplicações industriais de alto desempenho.

13 Conclusão

Algoritmos Genéticos estão longe de ser uma panacéia para os problemas de otimização, mas são, sem dúvida, técnicas úteis e relativamente robustas de otimização de funções multivariáveis, multimodais, possivelmente descontínuas e não-diferenciáveis.

Vale a pena notar que os trabalhos fundamentais de Rosenblatt [128], Minsky e Papert [105] em redes neurais, Zadeh em teoria de conjuntos

nebulosos [157], Rechenberg em estratégias evolucionárias [123], Fogel em programação evolucionária [45] e Holland em algoritmos genéticos [75] foram desenvolvidos aproximadamente na mesma época, em meados dos anos 60 e início dos anos 70. Outrora confinados a seus laboratórios e discípulos, essas técnicas de Inteligência Computacional amadureceram ao longo dos últimos anos e hoje formam não só um dos mais fascinantes ramos da ciência atual, mas também ferramentas de engenharia de comprovado valor prático e de potencial ainda inexplorado.

Puros ou combinados com outros métodos, AGs têm sido aplicados com sucesso a um sem-número de problemas, e já há até patentes e produtos desenvolvidos com o uso de AGs [59]. Cursos regulares de AGs são oferecidos em aproximadamente 30 universidades no mundo a nível de pós-graduação, e o número de praticantes e pesquisadores já é grande e continua crescendo rapidamente. Há listas eletrônicas com milhares de assinantes, várias conferências de carácter internacional, e empresas investindo recursos generosos em AGs, indicando que eles têm um futuro promissor pela frente.

Referências

[1] E. Aarts and J. Korst, *Simulated Annealing and Boltzmann Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing*, John Wiley (1989).

[2] D. H. Ackley and M. L. Littman, "Interactions Between Learning and Evolution," in C. G. Langton, C. Taylor, J. D. Farmer, and S. Rasmussen (eds.), *Artificial Life II*, Addison-Wesley, pp. 487–507 (1992).

[3] J. Andreoni and J. H. Miller, "Auctions with Adaptive Artificial Agents," *Technical Report 91-01-004*, Santa Fe Institute (1991).

[4] P. J. Angeline, G. M. Saunders, and J. B. Pollack, "An Evolutionary Algorithm that Constructs Recurrent Neural Networks," *IEEE Trans. on Neural Networks*, Vol. 5, No. 1, pp. 54–65 (1994).

[5] C. A. Ankenbrandt, "An Extension to the Theory of Convergence and a Proof of the Time Complexity of Genetic Algorithms," in G. J. E. Rawlins (ed.), *Foundations of Genetic Algorithms*, Morgan Kaufmann, pp. 53–68 (1991).

[6] M. A. Arbib (ed.), *The Handbook of Brain Theory and Neural Networks*, The MIT Press, pp. 98–102 (1995).

[7] I. Asimov, *The Human Body: Its Structure and Operation*, Penguin Books (1992).

[8] P. M. Athanas and H. F. Silverman, "Processor Reconfiguration through Instruction-Set Metamorphosis," *IEEE Computer*, Vol. 26, No. 3, pp. 11–18 (1993).

[9] R. Axelrod, *The Evolution of Cooperation*, Basic Books (1984).

[10] R. Axelrod, "The Evolution of Strategies in the Iterated Prisoner's Dilemma," in L. Davis (ed.), *Genetic Algorithms and Simulated Annealing*, Morgan Kaufmann (1987).

[11] T. Bäck, "Optimal Mutation Rates in Genetic Search," *Proc. Fifth Int. Conf. Genetic Algorithms*, pp. 2–8 (1993).

[12] J. E. Baker, "Adaptive Selection Methods for Genetic Algorithms," *Proc. First Int. Conf. Genetic Algorithms*, pp. 101–111 (1985).

[13] J. A. Barnden, "Artificial Intelligence and Neural Networks," in M. A. Arbib (ed.), *The Handbook of Brain Theory and Neural Networks*, The MIT Press, pp. 98–102 (1995).

[14] R. Belew and S. Forrest, "Learning and Programming in Classifier Systems," *Machine Learning*, Vol. 3, pp. 193–223 (1988).

[15] R. K. Belew, "Evolution, Learning and Culture: Computational Metaphors for Adaptive Algorithms," *Complex Systems*, Vol. 4, pp. 11-49 (1990).

[16] H. Bersini and F. J. Varela, "The Immune Recruitment Mechanism," *Proc. Fourth Int. Conf. Genetic Algorithms*, pp. 520–526 (1991).

[17] H. Bersini and G. Seront, "In Search of a Good Crossover between Evolution and Optimization," in B. Manderick and R. M'anner (eds.), *Parallel Problem Solving from Nature 2*, Elsevier, pp. 479–488 (1992).

[18] J. C. Bezdek, "What is Computational Intelligence," in J. M. Zurada, R. J. Marks II, and C. J. Robinson (eds.), *Computational Intelligence - Imitating Life*, IEEE Press, pp. 1–12 (1994).

[19] J. N. Bhuyan, V. V. Raghavan, and V. K. Elayavalli, "Genetic Algorithm for Clustering with an Ordered Representation," *Proc. Fourth Int. Conf. Genetic Algorithms*, pp. 408–415 (1991).

[20] J. E. Bishop and M. Waldholz, *Genome: The Story of the Most Astonishing Scientific Adventure of Our Time — The Attempt to Map All the Genes in the Human Body*, Simon & Schuster (1990).

[21] J. L. Blanton Jr. and R. L. Wainwright, "Multiple Vehicle Routing with Time and Capacity Constraints using Genetic Algorithms," *Proc. Fifth Int. Conf. Genetic Algorithms*, pp. 452–459 (1993).

[22] L. Bodin, B. Golden, A. Assad, and M. Ball, "Routing and Scheduling of Vehicles and Crews: The State of the Art," *Computing Operations Research*, Vol. 10, pp. 62-212 (1983).

[23] L. B. Booker, D. E. Goldberg, and J. H. Holland, "Classifier Systems and Genetic Algorithms," *Artificial Intelligence*, Vol. 40, pp. 235–282 (1989).

[24] S. D. Brown, R. J. Francis, J. Rose, and Z. G. Vranesic, *Field-Programmable Gate Arrays*, Kluwer Academic Publishers (1992).

[25] J. L. Casti, *Complexification*, HarperPerennial, pp. 212–229 (1994).

[26] T. P. Caudell and C. P. Dolan, "Parametric Connectivity: Training of Constrained Networks using Genetic Algorithms," *Proc. Third Int. Conf. Genetic Algorithms*, pp. 370–374 (1989).

[27] G. A. Cleveland and S. F. Smith, "Using Genetic Algorithms to Schedule Flow Shop Releases," *Proc. Third Int. Conf. Genetic Algorithms*, pp. 160–169 (1989).

[28] J. P. Cohoon, S. U. Hegde, W. N. Martin, and D. Richards, "Punctuated Equilibria: A Parallel Genetic Algorithm," *Proc. Second Int. Conf. Genetic Algorithms*, pp. 148–154 (1987).

[29] C. R. Darwin, *On The Origins of Species by Means of Natural Selection*, Penguin Classics (1985).

[30] Y. Davidor, "A Genetic Algorithm Applied to Robot Trajectory Generation," in L. Davis (ed.), *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, pp. 144–165 (1991).

[31] Y. Davidor, *Genetic Algorithms and Robotics*, World Scientific (1991).

[32] L. Davis, "Job Shop Scheduling with Genetic Algorithms," *Proc. First Int. Conf. Genetic Algorithms*, pp. 136–140 (1985).

[33] L. Davis (ed.), *Handbook of Genetic Algorithms*, Van Nostrand Reinhold (1991).

[34] L. Davis, "Genetic Algorithms for Optimization: Three Case Studies," in J. M. Zurada, R. J. Marks II, and C. J. Robinson (eds.), *Computational Intelligence - Imitating Life*, IEEE Press, pp. 416–426 (1994).

[35] K. Deb and D. E. Goldberg, "Analyzing Deception in Trap Functions," in L. D. Whitley (ed.), *Foundations of Genetic Algorithms 2*, Morgan Kaufmann, pp. 93–108 (1993).

[36] K. A. DeJong, *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*, Doctoral Thesis, Univ. of Michigan (1975).

[37] K. A. DeJong, "Genetic Algorithms: A 25 Year Perspective," in J. M. Zurada, R. J. Marks II, and C. J. Robinson (eds.), *Computational Intelligence - Imitating Life*, IEEE Press, pp. 125–134 (1994).

[38] A. E. Eiben, E. H. L. Aarts, and K. M. Van Hee, "Global Convergence of Genetic Algorithms: A Markov Chain Analysis," in H. P. Schwefel and R. Männer (eds.), *Parallel Problem Solving from Nature*, Springer, pp. 4–12 (1991).

[39] N. Eldredge and S. J. Gould, "Punctuated Equilibria: An Alternative to Phyletic Gradualism," in T. J. M. Schopf (ed.), *Models of Paleobiology*, Freeman, Cooper and Co., pp. 82–115 (1972).

[40] S. Fahlman, "An Empirical Study of Learning Speed in Back-Propagation Networks," *Technical Report CMU-CS-88-162*, Carnegie Mellon University (1988).

[41] H. L. Fang *Investigating GAs for Scheduling*, Master Thesis, Univ. of Edinburgh (1992).

[42] H. L. Fang, P. Ross, and D. Corne, "A Promising Genetic Algorithm Approach to Job-Shop Scheduling, Rescheduling & Open-Shop Scheduling Problems," *Proc. Fifth Int. Conf. Genetic Algorithms*, pp. 375–382 (1993).

[43] D. S. Feldman, "Fuzzy Network Synthesis with Genetic Algorithms," *Proc. Fifth Int. Conf. Genetic Algorithms*, pp. 312–317 (1993)

[44] T. Ferris, *Coming of Age in the Milky Way*, Vintage (1988).

[45] L. J. Fogel, A. J. Owens, and M. J. Walsh, *Artificial Intelligent Through Simulated Evolution*, John Wiley (1966).

[46] D. B. Fogel, "An Introduction to Simulated Evolutionary Optimization," *IEEE Trans. on Neural Networks*, Vol. 5, No. 1, pp. 3–14 (1994).

[47] C. M. Fonseca and P. J. Fleming, "Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization," *Proc. Fifth Int. Conf. Genetic Algorithms*, pp. 416–423 (1993).

[48] S. Forrest (ed.), *Emergent Computation: Self-Organizing, Collective, and Cooperative Phenomena in Natural and Artificial Computing Networks*, MIT Press (1990).

[49] T. Fukuda and T. Shibata, "Fuzzy-Neuro-AG Based Intelligent Robotics," in J. M. Zurada, R. J. Marks II, and C. J. Robinson (eds.), *Computational Intelligence - Imitating Life*, IEEE Press, pp. 352–363 (1994).

[50] M. Gardner, "Mathematical Games," *Scientific American*, Vol. 227, No. 2, p. 106 (1972).

[51] J. M. Geyer, "Connection Routing Algorithm for Printed Circuit Boards," *IEEE Trans. on Circuit Theory*, Vol. CT-18, pp. 95-100 (1971).

[52] D. E. Goldberg, "Optimal Initial Population Size for Binary-Coded Genetic Algorithms," *TCGA Report 85001*, Univ. of Alabama, Tuscaloosa (1985).

[53] D. E. Goldberg, "The Genetic Algorithm Approach: Why, How, and What Next?," in K. S. Narendra (ed.), *Adaptive and Learning Systems - Theory and Applications*, Plenum Press, pp. 247–253 (1986).

[54] D. E. Goldberg and P. Segrest, "Finite Markov Chain Analysis of Genetic Algorithms," *Proc. Second Int. Conf. Genetic Algorithms*, pp. 1–8 (1987).

[55] D. E. Goldberg and J. Richardson, "Genetic Algorithms with Sharing for Multimodal Function Optimization," *Proc. Second Int. Conf. Genetic Algorithms*, pp. 41–49 (1987).

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

[57] D. E. Goldberg, "A Comparative Analysis of Selection Schemes Used in Genetic Algorithms," in G. J. E. Rawlins (ed.), *Foundations of Genetic Algorithms*, Morgan Kaufmann, pp. 69–93 (1991).

[58] D. E. Goldberg, K. Deb and B. Korb, "Don't Worry, Be Messy," *Proc. Fouth Int. Conf. Genetic Algorithms*, pp. 24-30 (1991).

[59] D. E. Goldberg, "Genetic and Evolutionary Algorithms: Come of Age," *Communications of the ACM*, Vol. 37, No. 3, pp. 113–119 (1994).

[60] V. S. Gordon and D. Whitley, "Serial and Parallel Genetic Algorithms as Function Optimizers," *Proc. Fifth Int. Conf. Genetic Algorithms*, pp. 177–183 (1993)

[61] J. J. Grefenstette, R. Gopal, B. Rosmaita, and D. V. Gucht, "Genetic Algorithms for the Traveling Salesman Problem," *Proc. First Int. Conf. Genetic Algorithms*, pp. 160–168 (1985).

[62] J. J. Grefenstette, "Optimization of Control Parameters for Genetic Algorithms," *IEEE Trans. Systems, Man, and Cybernetics*, Vol. SMC-16, No. 1, pp. 122-128 (1986).

[63] J. J. Grefenstette, "Incorporating Problem Specific Knowledge in Genetic Algorithms," in *Genetic Algorithms and Simulated Annealing* (Davis, L., ed.), Morgan Kaufmann, pp. 42–57 (1987).

[64] J. J. Grefenstette e J. E. Baker, "How Genetic Algorithms Work: A Critical Look at Implicit Parallelism," *Proc. Third Int. Conf. Genetic Algorithms*, pp. 20–27 (1989).

[65] J. J. Grefenstette, "Deception Considered Harmful," in L. D. Whitley (ed.), *Foundations of Genetic Algorithms 2*, Morgan Kaufmann, pp. 75–91 (1993).

[66] F. Gruau, "Genetic Synthesis of Modular Neural Networks," *Proc. Fifth Int. Conf. Genetic Algorithms*, pp. 318–325 (1993).

[67] S. A. Harp, "Towards the Genetic Synthesis of Neural Networks," *Proc. Third Int. Conf. Genetic Algorithms*, pp. 360–369 (1989).

[68] S. A. Harp and T. Samad, "Genetic Synthesis of Neural Network Architecture," in L. Davis (ed.), *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, pp. 202–221 (1991).

[69] L. Hasdorff, *Gradient Optimization and Nonlinear Control*, John Wiley (1976).

[70] S. Haykin, *Neural Networks: A Comprehensive Foundation*, IEEE Press (1994).

[71] S. W. Hawking, *A Brief History of Time: From the Big Bang to Black Holes*, Bantam Books (1988).

[72] F. S. Hillier and G. J. Lieberman, *Operations Research*, 2nd ed., Holden Day (1974).

[73] W., D. Hillis, "Co-Evolving Parasites Improve Simulated Evolution as an Optimization Procedure," *Physica D*, Vol. 42, pp. 228-234 (1990).

[74] G. E. Hinton and S. J. Nowlan, "How Learning Can Guide Evolution," *Complex Systems*, Vol. 1, pp. 495–502 (1990).

[75] J. H. Holland, *Adaptation in Natural and Artificial Systems*, Univ. of Michigan Press (1975).

[76] J. H. Holland, *Induction: Processes of Inference, Learning, and Discovery*, MIT Press (1986).

[77] J. Holland and J. H. Miller, "Artificial Adaptive Agents in Economic Theory," *Technical Report 91-05-025*, Santa Fe Institute (1991).

[78] J. H. Holland, "Genetic Algorithms," *Scientific American*, Vol. 267, No. 1, pp. 66–72 (1992).

[79] R. B. Hollstien, *Artificial Genetic Adaption in Computer Control Systems*, PhD Thesis, Univ. of Michigan (1971).

[80] R. Horst and H. Tuy, *Global Optimization—Deterministic Approaches*, Springer Verlag (1990).

[81] H. Iba, T. Kurita, H. Garis, and T. Sato, "System identification using Structured Genetic Algorithms," *Proc. Fifth Int. Conf. Genetic Algorithms*, pp. 279–286 (1993).

[82] M. Iosifescu, *Finite Markov Processes and Their Applications*, John Wiley (1980).

[83] P. Jog, J. Y. Suh, and D. V. Gucht, "The Effects of Population Size, Heuristic Crossover and Local Improvement on a Genetic Algorithm for the Traveling Salesman Problem," *Proc. of the Third Int. Conf. Genetic Algorithms*, pp. 110-115 (1989).

[84] D. S. Johnson and C. H. Papadimitriou, "Computational Complexity," in E. L. Lawer *et. al* (eds.), *The Traveling Salesman Problem: A Guided Tour to Combinatorial Optimization*, John Wiley, pp. 37–85 (1985).

[85] R. M. Karp, "Combinatorics, Complexity, and Randomness," *Communications of the ACM*, Vol. 29 No. 2, pp. 98–109 (1986).

[86] A. Kaufmann and R. Faure, *Introduction to Operations Research*. Academic Press (1968).

[87] H. Kitano, "Empirical Studies on the Speed of Convergence of Neural Network Training using Genetic Algorithms," *Proc. Eigth National Conf. Artificial Intelligence*, Vol. 2, pp. 789–795 (1990).

[88] S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vecchi, "Optimization by Simulated Annealing," *Science*, No. 220, pp. 671–680 (1983).

[89] T. Kohonen, *Self-Organization and Associative Memory*, Springer-Verlag (1989).

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

[91] J. R. Koza, *Genetic Programming II: Automatic Discovery of Reusable Programs*, MIT Press (1994).

[92] K. Krishnakumar, D. E. Goldberg, "Control System Optimization Using Genetic Algorithms," *Journal of Guidance, Control, and Dynamics*, Vol. 15, No. 3, pp. 735–740 (1992).

[93] E. L. Lawer, J. K. Kenstra, A. H. G. R. Kan, and D. B. Shmoys (eds.), *The Traveling Salesman Problem: A Guided Tour to Combinatorial Optimization*, John Wiley (1985).

[94] G. E. Liepins and M. D. Vose, "Deceptiveness and Genetic Algorithm Dynamics," in G. J. E. Rawlins (ed.), *Foundations of Genetic Algorithms*, Morgan Kaufmann, pp. 36–50 (1991).

[95] M. E. Lines, *Think of a Number*, The Institute of Physics Publishing, pp. 109–118 (1990).

[96] J. D. Litke, "An Improved Solution to the Traveling Salesman Problem with Thousands of Nodes," *Communications of the ACM*, Vol. 27, No. 12, pp. 1227–1236 (1984).

[97] D. G. Luenberger, *Optimization by Vector Space Methods*, John Wiley & Sons (1969).

[98] B. MacLennan, "Synthetic Ethology: An Approach to the Study of Communication," in C. G. Langton, C. Taylor, J. D. Farmer, and S. Rasmussen (eds.), *Artificial Life II*, Addison-Wesley, pp. 631–655 (1992).

[99] V. Maniezzo, "Genetic Evolution of the Topology and Weight Distribution of Neural Networks," *IEEE Trans. on Neural Networks*, Vol. 5, No. 1, pp. 39–53 (1994).

[100] B. Martos, *Nonlinear Programming: Theory and Methods*, North-Holland (1975).

[101] E. Mayr, *Animal Species and Evolution*, Belknap Press (1963).

[102] E. Mayr, *One Long Argument: Charles Darwing and the Genesis of Modern Evolutionary Thought*, Harvard Univ. Press (1991).

[103] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, Springer-Verlag (1992).

[104] G. F. Miller, P. M. Todd, and S. U. Hegde, "Designing Neural Networks using Genetic Algorithms," *Proc. Third Int. Conf. Genetic Algorithms*, pp. 379–384 (1989).

[105] M. L. Minsky and S. Papert, *Perceptrons: An Essay in Computational Geometry*, MIT Press (1969).

[106] M. Mitchell and S. Forrest, "Genetic Algorithms and Artificial Life," *Artificial Life*, Vol. 1, No. 3, pp. 267–289 (1994).

[107] D. Montana and L. Davis, "Training Feedforward Neural Networks using Genetic Algorithms," *Proc. Eleventh Joint Conf. Artificial Intelligence*, pp. 762-767 (1989).

[108] H. Mühlenbein, "Parallel Genetic Algorithms, Population Genetics and Combinatorial Combinatorial Optimization," *Proc. Third Int. Conf. Genetic Algorithms*, pp. 416–421 (1989).

[109] H. Mühlenbein, "Parallel Genetic Algorithms and Neural Networks as Learning Machines," in D. J. Evans, G. R. Joubert, and H. Liddell (eds.), *Parallel Computing'91*, pp. 91–103 (1991).

[110] H. Mühlenbein, "Evolution in Time and Space: The Parallel Genetic Algorithm," in G. J. E. Rawlins (ed.), *Foundations of Genetic Algorithms*, Morgan Kaufmann, pp. 316–337 (1991).

[111] R. Nakano, "Conventional Genetic Algorithms for Job-Shop Problems," *Proc. Fourth Int. Conf. Genetic Algorithms*, pp. 474–479 (1991).

[112] J. P. Nordvik, and J. M. Renders, "Genetic Algorithms and Their Potential for Use in Process Control: A Case Study," *Proc. Fourth Int. Conf. Genetic Algorithms*, pp. 480–486 (1991).

[113] M. O. Odetayo and D. R. McGregor, "Genetic Algorithm for Inducing Control Rules for a Dynamic System," *Proc. Third Int. Conf. Genetic Algorithms*, pp. 177-182 (1989).

[114] I. M. Olivier, D. J. Smith, and J. R. C. Holland, "A Study of Permutation Crossover Operators on the Traveling Salesman Problem," *Proc. Second Int. Conf. Genetic Algorithms*, pp. 224–230 (1987).

[115] N. H. Packard, "Intrinsic Adaptation in a Simple Model for Evolution," in C. G. Langton (ed.), *Artificial Life*, Addison-Wesley, pp. 141-155 (1989).

[116] C. B. Pettey, M. R. Leuze, and J. J. Grefenstette, "A Parallel Genetic Algorithm," *Proc. Second Int. Conf. Genetic Algorithms*, pp. 155–161 (1987).

[117] D. Polani and T. Uthmann, "Training Kohonen Feature Maps in Different Topologies: an Analysis using Genetic Algorithms," *Proc. Fifth Int. Conf. Genetic Algorithms*, pp. 326–333 (1993)

[118] D. J. Powell, M. M. Skolnick, and S. S. Tong, "Interdigitation: A Hybrid Technique for Engineering Design Optimization Employing Genetic Algorithms, Expert Systems, and Numerical Optimization," in L. Davis (ed.), *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, pp. 312-331 (1991).

[119] M. J. D. Powell, "Some Properties of the Variable Metric Algorithm," in F. A. Lootsma (ed.), *Numerical Methods for Non-Linear Optimization*, Academic Press, pp. 1–17 (1972).

[120] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C*, 2nd ed., Cambridge Univ. Press (1992).

[121] T. S. Ray, "Is it Alive, or is it GA?," *Proc. Fourth Int. Conf. Genetic Algorithms*, pp. 527–534 (1991).

[122] T. S. Ray, "An Approach to the Synthesis of Life," in C. G. Langton, C. Taylor, J. D. Farmer, and S. Rasmussen (eds.), *Artificial Life II*, Addison-Wesley, pp. 371–408 (1991).

[123] I. Rechenberg, *Evolutionsstrategie: Optimierung Technischer Systeme nach Prinzipien der Biolgischen Evolution*, Frommann-Holzboog Verlag (1973).

[124] I. Rechenberg, "Evolution Strategy," in J. M. Zurada, R. J. Marks II, and C. J. Robinson (eds.), *Computational Intelligence - Imitating Life*, IEEE Press, pp. 147–159 (1994).

[125] R. M Restak, *The Brain*, Bantam Books (1984).

[126] C. R. Reeves, "Using Genetic Algorithms with Small Populations," *Proc. Fifth Int. Conf. Genetic Algorithms*, pp. 92–99 (1993).

[127] S. G. Romaniuk, "Evolutionary Growth Perceptrons," *Proc. Fifth Int. Conf. Genetic Algorithms*, pp. 334–341 (1993).

[128] F. Rosenblatt, *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*, Spartan (1962).

[129] T. J. Ross, *Fuzzy Logic with Engineering Applications*, McGraw-Hill (1995).

[130] G. Rudolph, "Convergence Analysis of Canonical Genetic Algorithms," *IEEE Trans. on Neural Networks*, Vol. 5, No. 1, pp. 96–101 (1994).

[131] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Internal Representations by Error Propagation," in D. E. Rumelhart and J. L. McClelland (eds), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, MIT Press, pp. 318–362 (1986).

[132] S. D. Scott, *HGA: A Hardware-Based Genetic Algorithm*, Master Thesis, Univ. of Nebraska (1994).

[133] J. D. Shaffer, "An Adaptive Crossover Distribution Mechanism for Genetic Algorithms," *Proc. Second Int. Conf. Genetic Algorithms*, pp. 36–40 (1987).

[134] J. D. Schaffer, "Combinations of Genetic Algorithms with Neural Networks or Fuzzy Systems," in J. M. Zurada, R. J. Marks II, and C. J. Robinson (eds.), *Computational Intelligence - Imitating Life*, IEEE Press, pp. 371–382 (1994).

[135] C. J. Shatz, "The Developing Brain," *Scientific American*, Vol. 267, No. 3, pp. 61–67 (1992).

[136] H. P. Schwefel, "On the Evolution of Evolutionary Computation," in J. M. Zurada, R. J. Marks II, and C. J. Robinson (eds.), *Computational Intelligence - Imitating Life*, IEEE Press, pp. 116–124 (1994).

[137] D. Smith, "Bin Packing with Adaptive Search," *Proc. First Int. Conf. Genetic Algorithms*, pp. 202–207 (1985).

[138] R. E. Smith and D. E. Goldberg, "Diploidy and Dominance in Artificial Genetic Search," *Complex Systems*, Vol. 6, pp. 251–285 (1992).

[139] W. R. Smythe Jr. and L. A. Johnson, *Introduction to Linear Programming with Applications*, Prentice-Hall (1966).

[140] W. M. Spears and K. A. DeJong, "An Analysis of Multi-Point Crossover," in G. J. E. Rawlins (ed.), *Foundations of Genetic Algorithms*, Morgan Kaufmann, pp. 301–315 (1991).

[141] M. Srinivas and L. M. Patnaik, "Genetic Algorithms: A Survey," *Computer*, Vol. 27, No. 6, pp. 17–26 (1994).

[142] D. T. Suzuki, A. J. F. Griffiths, J. H. Miller, and R. C. Lewontin, *An Introduction to Genetic Analysis*, 4th ed., W. H. Freeman (1989)

[143] G. Syswerda, "Uniform Crossover in Genetic Algorithms," *Proc. Third Int. Conf. Genetic Algorithms*, pp. 2–9 (1989).

[144] G. Syswerda, "A Study of Reproduction in Generational and Steady-State Genetic Algorithms," in G. J. E. Rawlins (ed.), *Foundations of Genetic Algorithms*, Morgan Kaufmann, pp. 94–101 (1991).

[145] T. Tagami and J. Tanomaru, "Real-Coded Genetic Algorithms for Non-Stationary Function Optimization," *Trans. of the Inst. of Electronics, Information, and Communications Engineers of Japan* (in press, in Japanese).

[146] R. Tanese, "Parallel Genetic Algorithms for a Hypercube," *Proc. Second Int. Conf. Genetic Algorithms*, pp. 177–183 (1987).

[147] J. Tanomaru and K. Oka, "Evolutionary Algorithm for Automatic Wire Routing," *Proc. 4th Golden West Int. Conf. Intelligent Systems*, pp. 97–101 (1995).

[148] J. Tanomaru, "Planejamento de Mão-de-Obra por um Algoritmo Genético," *Proc. II Congresso Brasileiro de Redes Neurais* (1995).

[149] J. Tanomaru, "Solution of a Difficult Workforce Scheduling Problem by a Genetic Algorithm," *Trans. of the Information Processing Society of Japan* (in press).

[150] S. R. Thangiah, R. Vinayagamorthy, and A. V. Gubbi, "Vehicle Routing with Time Deadlines using Genetic and Local Algorithms," *Proc. Fifth Int. Conf. Genetic Algorithms*, pp. 506–513 (1993).

[151] T. Toffoli, "Cellular Automata," in M. A. Arbib (ed.), *The Handbook of Brain Theory and Neural Networks*, The MIT Press, pp. 166–169 (1995).

[152] G. M. Werner and M. G. Dyer, "Evolution of Communication in Artificial Organisms," in C. G. Langton, C. Taylor, J. D. Farmer, and S. Rasmussen (eds.), *Artificial Life II*, Addison-Wesley, pp. 659–687 (1992).

[153] D. Whitley, "The Genitor Algorithm and Selection Pressure: Why Rank-Based Allocation of Reproductive Trials is Best," *Proc. Third Int. Conf. Genetic Algorithms*, pp. 116–121 (1989).

[154] D. Whitley and T. Hanson, "Optimizing Neural Networks using Faster, More Accurate Genetic Search," *Proc. Third Int. Conf. Genetic Algorithms*, pp. 391–396 (1989).

[155] D. Whitley, "Cellular Genetic Algorithms," *Proc. Fifth Int. Conf. Genetic Algorithms*, p. 658 (1993)

[156] S. W. Wilson and D. E. Goldberg, "A Critical Review of Classifier Systems," *Proc. Third Int. Conf. Genetic Algorithms*, pp. 244–255 (1989).

[157] L. A. Zadeh, "Fuzzy Sets and Systems," *Proc. Symposium on Systems Theory*, pp. 29–37 (1965).