Computação Evolutiva: Uma Abordagem Pragmática

Fernando J. Von Zuben

DCA/FEEC/Unicamp - Caixa Postal 6101 - 13083-970

e-mail: vonzuben@dca.fee.unicamp.br

Resumo: Este tutorial cobre aspectos básicos da história e do estado da arte da computação evolutiva, uma área de pesquisa muito jovem, sendo que a própria denominação foi proposta em 1991, representando um esforço em congregar diversas iniciativas de pesquisa que visavam simular aspectos específicos do processo evolutivo, originalmente propostas nos anos 50. Os propósitos, a estrutura geral e os princípios de operação dos algoritmos evolutivos são apresentados, sendo que todos eles têm uma estrutura básica comum: realizam reprodução, impõem variações aleatórias, promovem competição e executam seleção de indivíduos de uma dada população. Sempre que estes quatro processos estiverem presentes, seja na natureza ou em uma simulação computacional, a evolução é o produto resultante. O enfoque pragmático deste texto está vinculado ao emprego da computação evolutiva no desenvolvimento de técnicas para solução de problemas de otimização. Neste caso, o problema a ser resolvido faz o papel do ambiente, e cada indivíduo da população é associado a uma solução-candidata. Sendo assim, um indivíduo vai estar mais adaptado ao ambiente sempre que ele corresponder a uma solução mais eficaz para o problema. Com a evolução, espera-se a cada geração ir obtendo soluções-candidatas mais e mais eficazes, embora não exista a garantia de se chegar à solução ótima ao final do processo evolutivo. Neste contexto, um algoritmo evolutivo exerce o papel de um processo poderoso de busca iterativa e em paralelo, adequada para o tratamento de problemas de otimização caracterizados por uma explosão combinatória de possibilidades, ausência de diferenciabilidade do critério de otimalidade ou multimodalidade. Na verdade, todo problema suficientemente complexo, a ponto de dificultar a produção de uma formulação matemática abrangente e o atendimento de requisitos básicos de tratabilidade por ferramentas convencionais, se transforma em um candidato para ser abordado a partir da computação evolutiva, já que a aplicação de técnicas de solução conhecidas, dedicadas e capazes de garantir a obtenção de uma solução ótima, não é possível nestes casos.

Palavras-chave: computação evolutiva, fenótipo, genótipo, codificação genética, operadores genéticos, superfície de adaptação, algoritmos evolutivos, algoritmos genéticos, programação evolutiva, estratégias evolutivas, sistemas classificadores, programação genética, complexidade computacional, otimização combinatória, inteligência computacional, aprendizado de máquina.

Índice

1 Introdução

Conceitos de computação evolutiva têm sido empregados em uma variedade de disciplinas, desde ciências naturais e engenharia até biologia e ciência da computação. A idéia básica, surgida nos anos 50, é aplicar o processo de evolução natural como um paradigma de solução de problemas, a partir de sua implementação em computador. Os problemas de otimização são aqueles que vêm recebendo mais contribuições a partir de técnicas de computação evolutiva, razão pela qual serão adotados como exemplos e pontos de partida para a descrição e formalização dos conceitos e técnicas a serem apresentados. Além disso, muitos problemas de engenharia podem ser adequadamente apresentados como problemas de otimização (MICHALEWICZ & FOGEL, 2000), garantindo que o escopo de abordagem se mantenha muito amplo.

A vantagem mais significativa da computação evolutiva está na possibilidade de resolver problemas pela simples descrição matemática do que se quer ver presente na solução, não havendo necessidade de se indicar explicitamente os passos até o resultado, que certamente seriam específicos para cada caso. É lógico que os algoritmos evolutivos correspondem a uma seqüência de passos até a solução, mas estes passos são os mesmos para uma ampla gama de problemas, fornecendo robustez e flexibilidade. Sendo assim, a computação evolutiva deve ser entendida como um conjunto de técnicas e procedimentos genéricos e adaptáveis, a serem aplicados na solução de problemas complexos, para os quais outras técnicas conhecidas são ineficazes ou nem sequer são aplicáveis.

Trata-se de um novo paradigma de solução de problemas, pois se abre mão da garantia de obtenção da solução ótima para se conquistar a tratabilidade via uma ferramenta de propósito geral. Novos paradigmas afloram sempre que condições propícias para tal passam a vigorar, e no caso da computação evolutiva a condição primordial é a disponibilidade de uma grande quantidade de recursos computacionais.

Em termos históricos, três algoritmos para computação evolutiva foram desenvolvidos independentemente:

- algoritmos genéticos: HOLLAND (1962), BREMERMANN (1962) e FRASER (1957);
- programação evolutiva: FOGEL (1962);
- estratégias evolutivas: RECHENBERG (1965) e SCHWEFEL (1965).

Hoje temos também os sistemas classificadores (BOOKER *et al.*, 1989) e a programação genética (KOZA, 1992). Uma coletânea dos artigos seminais, a partir de 1956, pode ser encontrada em FOGEL (1998).

A computação evolutiva engloba, portanto, uma família de algoritmos inspirados na teoria evolutiva de Darwin. Os primeiros livros e teses sobre computação evolutiva, escritos por alguns dos próprios pioneiros da área, já apresentavam demonstrações impressionantes acerca da capacidade dos algoritmos evolutivos (FOGEL *et al.*, 1966; RECHENBERG, 1973; HOLLAND, 1975; DE JONG, 1975; SCHWEFEL, 1975), apesar das limitações de hardware existentes na época. No entanto, de modo similar a outras iniciativas de propor métodos de solução de problemas inspirados na natureza, tal como redes neurais artificiais e sistemas nebulosos, os algoritmos evolutivos também tiveram que atravessar um longo período de rejeição e incompreensão antes de receber o reconhecimento da comunidade científica. Os progressos verificados nos anos 90 confirmaram o poder impressionante dos algoritmos evolutivos na solução de problemas de elevada complexidade, assim como evidenciaram suas limitações. Uma referência elaborada com o intuito de ser completa, didática e muito adequada para descrever o estado da arte da pesquisa em computação evolutiva é o *Handbook of Evolutionary Computation* (BÄCK *et al.*, 1997), tendo sido atualizada recentemente (BÄCK *et al.*, 2000a,b). Outros trabalhos importantes para um aprofundamento no estudo desta

técnica, sob vários pontos de vista, são: BÄCK (1996), DAVIS (1991), FOGEL (1999), GOLDBERG (1989), HOLLAND (1992), KINNEAR (1994), KOZA (1992), MICHALEWICZ (1996), MITCHELL (1996) e SCHWEFEL (1995).

Os algoritmos evolutivos não devem ser considerados "prontos para uso", mas sim um elenco de procedimentos gerais que podem ser prontamente adaptados a cada contexto de aplicação. Basicamente, eles são modelos computacionais que recebem como entrada:

- uma população de indivíduos em representação genotípica (geração inicial), que correspondem a soluções-candidatas junto a problemas específicos; e
- uma função que mede a adequação relativa de cada indivíduo frente aos demais (função de adequação, adaptabilidade ou *fitness*).

A representação genotípica corresponde a uma descrição de cada indivíduo da população através de uma lista ordenada (cromossomo) ou árvore de atributos, descritos a partir de um alfabeto finito. Cada atributo da lista ou árvore é equivalente a um gene, e o valor do atributo corresponde a um alelo. O tamanho da lista ou árvore está diretamente associado ao número mínimo de atributos necessários para descrever cada indivíduo (solução-candidata) da população. No caso de representação em lista, os indivíduos da população geralmente têm tamanho único, embora já existam abordagem que permitem o tratamento de cromossomos de tamanho variável, ou seja, indivíduos mais e menos complexos co-existindo em uma dada geração. A representação em árvore é necessária sempre que uma lista de atributos não for capaz de descrever um indivíduo da população, e aí o tamanho da árvore é sempre variável.

2 Motivações para simulação computacional de processos evolutivos

As motivações para se desenvolver algoritmos capazes de simular processos evolutivos em computador podem ser classificadas como segue:

- necessidade de validar teorias e conceitos associados à biologia da evolução: computação evolutiva auxiliando na compreensão de processos evolutivos naturais;
- capacidade de lidar com problemas para os quais não é possível ou é muito custoso obter uma descrição detalhada, ou ainda junto ao qual não é possível impor restrições muito fortes, ambas condições necessárias para a aplicação de ferramentas de solução dedicadas e, portanto, mais eficientes. Por exemplo, algoritmos de programação linear requerem que a função-objetivo seja linear; caso ela não seja linear, algoritmos de busca baseados no gradiente requerem que a função-objetivo seja diferenciável e que se possa calcular esta derivada a um baixo custo computacional. Na ausência de linearidade e na impossibilidade de se obter a derivada (seja porque ela não existe ou por representar uma etapa muito custosa) da função-objetivo, algoritmos evolutivos passam a representar uma das poucas alternativas de se chegar à solução, como será mostrado mais adiante;
- possibilidade de recorrer a técnicas de solução adaptativas, ou seja, capazes de manter o desempenho mesmo quando o ambiente é não-estacionário, ou seja, quando o problema está sujeito a pequenas variações em suas especificações: não é necessário reiniciar todo o processo de busca de uma solução frente a pequenas mudanças nas especificações do problema, já que refinamentos podem ser obtidos a partir das soluções atuais;
- capacidade de gerar soluções suficientemente boas em um tempo suficientemente rápido junto a problemas de elevada complexidade: enquanto técnicas convencionais de obtenção da solução ótima são intratáveis, por requererem uma quantidade inatingível de recursos computacionais, algoritmos evolutivos são capazes de fornecer boas solução, não

necessariamente ótimas, requerendo uma quantidade aceitável de recursos computacionais;

- possibilidade de incorporar conhecimento em um computador (aprendizado de máquina) sem a necessidade de programá-lo para tal, ou seja, sem a necessidade de recorrer ao conhecimento humano expresso, por exemplo, através de uma base de regras: a computação evolutiva possibilita que o computador ganhe proficiência na execução de tarefas antes restritas a especialistas humanos, simplesmente realizando ações e recebendo a realimentação acerca das conseqüências das ações tomadas, única fonte de informação para a evolução do processo de aprendizagem.

3 Paradigmas de solução de problemas

O enfoque adotado em seções anteriores deixa evidente a existência de técnicas alternativas de abordagem de problemas, sendo necessário estabelecer uma classificação que ajude a distingui-las:

- métodos fortes: são concebidos para resolverem problemas genéricos, mas foram desenvolvidos para operarem em um mundo específico, onde impera linearidade, continuidade, diferenciabilidade e/ou estacionariedade. Exemplo: método do gradiente e técnicas de programação linear (busca iterativa).
- métodos específicos: são concebidos para resolverem problemas específicos em mundos específicos. Exemplo: toda técnica que conduz a uma solução na forma fechada.
- métodos fracos: são concebidos para resolverem problemas genéricos em mundos genéricos. Operam em mundos não-lineares e não-estacionários, embora não garantam eficiência total na obtenção da solução. No entanto, geralmente garantem a obtenção de uma "boa aproximação" para a solução, sendo que a complexidade algorítmica cresce a uma taxa menor que exponencial com o aumento do "tamanho" do problema. Exemplo: técnicas baseadas em computação evolutiva.

Independente da aplicação, métodos fracos devem ser considerados se e somente se métodos fortes (soluções clássicas) e métodos específicos (soluções dedicadas) não existem, não se aplicam, ou falham quando aplicados.

Conclui-se então que soluções baseadas em computação evolutiva devem ser consideradas como o último recurso. No entanto, subtraindo-se os problemas tratáveis pelos métodos fortes e métodos específicos, o campo de aplicação para técnicas de computação evolutiva é extremamente vasto.

4 A base biológica

4.1 A idéia da hereditariedade

A história da genética é fascinante, principalmente pela velocidade com que se partiu de observações para demonstrações experimentais dos mecanismos fundamentais envolvidos, isto a partir da segunda metade do século 19 (BURNS & BOTTINO, 1988). No entanto, as primeiras idéias vinculadas à hereditariedade datam de 6000 anos atrás. Por volta de 500 a.C. (tempo de Aristóteles), filósofos gregos propuseram que "vapores" derivados de vários órgãos se uniam sob a ação vitalizadora do sêmen, interpretado como sangue altamente purificado. Esta idéia influiu sobre a cultura da humanidade por cerca de 2000 anos. Neste período, defendia-se a idéia de que o sexo era determinado pela procedência do líquido seminal no homem: se do testículo direito, sexo masculino; se do testículo esquerdo, sexo feminino. A mulher era vista apenas como uma incubadeira neste processo.

Para enriquecer o debate, ocorreu em 1672 a descoberta do óvulo, pelo holandês Graaf. Esta descoberta foi de grande importância por indicar que as fêmeas de mamíferos também apresentavam ovulação, o que poderia significar que o papel da mulher na reprodução era muito mais importante do que se supunha. Em 1675 foi descoberto o espermatozóide pelo holandês Von Leeuwenhoeck. A partir de então, passou a vigorar a curiosa (e hoje absurda) idéia do homúnculo, ou seja, defendia-se que o embrião já estaria na forma final, apenas ganhando proporções maiores durante a gestação, embora não se sabia se ele era originado unicamente do espermatozóide ou unicamente do óvulo.

As primeiras idéias fundamentadas acerca da hereditariedade surgiram efetivamente em 1866, com o monge agostiniano Gregor Mendel. Ele atacou o problema de modo simples e lógico, escolheu material adequado, concentrou-se em poucas características contrastantes, desenvolveu um programa de cruzamentos controlados, tratou os resultados de forma eficiente e sugeriu fatores causais (hoje chamados de genes) como os responsáveis pelos fenômenos observados. Ninguém antes havia chegado tão perto da compreensão real da hereditariedade (ficou faltando apenas elucidar os mecanismos celulares envolvidos), mas foram necessários mais de 30 anos para que a comunidade científica se desse conta da importância e eficácia destes resultados. É curioso constatar que um dos fatores que dificultou a assimilação dos resultados de Mendel foi justamente o intenso debate reinante nos meios científicos após a divulgação das leis de seleção natural de DARWIN (1859).

4.2 Darwinismo × Lamarckismo

Desde DARWIN (1859), a teoria da evolução vem sendo a principal idéia unificadora nas mais diversas áreas da biologia, pois a seleção natural é a força propulsora que distingue os sistemas biológicos dos demais sistemas físicos e químicos.

A teoria da seleção natural não prevê apenas a ocorrência de variações sucessivas (com resultante não-nula) junto aos indivíduos de uma dada espécie, fato já conhecido anteriormente, mas também indica o tipo de variação, as quais devem necessariamente conduzir o organismo a uma melhor adaptação ao meio. O ponto forte da teoria darwinista (Charles Darwin, naturalista inglês, 1809-1882) é justamente a explicação de como se dá a adaptação, sendo que a teoria lamarckista (Jean Baptiste de Lamarck, naturalista francês, 1744-1829), a única alternativa a esta idéia na época, não podia explicar o processo adaptativo. No entanto, a rejeição da teoria lamarckista não podia ser demonstrada de forma simples. Até o início do século 20, não estava ainda claro qual das duas teorias explicava melhor o processo evolutivo.

Lamarck acreditava na herança direta de características adquiridas pelos indivíduos durante sua vida. Darwin, por sua vez, propôs que a seleção natural, associada à diversidade, poderia explicar melhor a evolução. O próprio Darwin, embora certo de que sua teoria descrevia os fatores predominantes do processo evolutivo, não descartava totalmente a teoria lamarckista, acreditando que ela podia representar um pequeno papel neste processo.

A teoria lamarckista foi considerada viável até que um trabalho publicado em 1893 passou a ser amplamente aceito. Weismann, autor do trabalho, constatou que organismos superiores apresentam dois tipos de células: as células germinativas (que passam informação genética aos descendentes) e as células somáticas (que compõem o organismo em suas partes não diretamente associadas à reprodução).

A contribuição de Weismann foi indicar a impossibilidade de que informações adquiridas pelas células somáticas sejam transmitidas aos descendentes pelas células germinativas. É possível expressar o argumento de Weismann em termos moleculares, através do dogma central da biologia molecular: a informação pode passar de DNA para DNA, e de DNA para proteína, mas não de proteína para DNA. Neste caso, a informação é representada

pela seqüência básica do DNA, que é transmitida para novas moléculas de DNA no processo de replicação e que especifica a seqüência de aminoácidos das proteínas no processo de tradução. O fato de que a informação passa do DNA para a proteína através do RNA (mensageiro intermediário) complica o argumento, mas não altera a essência, como apresentado na figura 1.

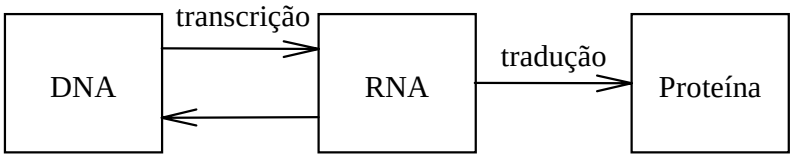


Figura 1 - Fluxo de informação em um sistema genético

Com base neste dogma, e assumindo que os ácidos nucléicos representam a única forma de se transmitir informação entre gerações, resultam implicações fundamentais para a evolução. Como toda mudança evolutiva requer modificação nos ácidos nucléicos (por exemplo, via mutação), então estas mudanças são essencialmente acidentais e inerentemente não-adaptativas. Esta é a base da teoria neodarwinista.

4.3 A teoria da seleção natural: do Darwinismo ao Neodarwinismo

DARWIN (1859) apresentou as seguintes hipóteses para explicar o processo de seleção natural:

1. os filhos tendem a ser em maior número que os pais;
2. o número de indivíduos de uma espécie permanece aproximadamente constante;
3. de (1) e (2), conclui-se que vai haver uma luta pela sobrevivência;
4. dentro de uma mesma espécie, os indivíduos apresentam pequenas diferenças, sendo que a maioria delas também está presente nos respectivos pais;

Conclui-se então que o princípio da seleção natural indica que os indivíduos cujas variações se adaptam melhor ao ambiente terão maior probabilidade de sobreviver e se reproduzir. A evolução darwiniana é nada mais que a conseqüência inevitável da competição entre sistemas de reprodução de informação, operando no interior de uma arena finita em um universo com diferencial de entropia positivo (ATMAR, 1992).

Embora Darwin tenha considerado estas hipóteses como suficientes para explicar a origem das espécies, hoje elas são aceitas apenas como suficientes para explicar os processos adaptativos em nichos ecológicos. Para transformar esta “teoria de ecologia” em uma “teoria da origem das espécies” é necessário considerar três hipóteses adicionais vinculadas à genética, conduzindo ao neodarwinismo:

5. algum processo de variação continuada deve ser responsável pela introdução de novas informações junto à carga genética dos organismos;
6. não há limite para a sucessão de variações que podem ocorrer;
7. a seleção natural é o mecanismo para preservação das novas informações que correspondam a uma maior adaptação.

Logo, a seleção natural é probabilística, e seu alvo primário é o indivíduo, embora seu efeito resultante vai se manifestar na espécie como um todo. A espécie é o beneficiário final do processo evolutivo (MAYR, 1988).

4.4 Terminologia básica para representação computacional

A terminologia básica a ser empregada representa uma analogia às entidades biológicas reais, sendo que as entidades computacionais corresponderão invariavelmente a estruturas bem mais simples que seus equivalentes biológicos:

- célula: unidade estrutural básica dos seres vivos, que se compõe de numerosas partes, sendo as fundamentais a parede ou membrana, o protoplasma e o núcleo. A célula é a menor unidade de matéria viva que pode existir de maneira independente, e ser capaz de se reproduzir. Toda célula de um mesmo organismo contém o mesmo conjunto de um ou mais cromossomos. Nos seres humanos, cada célula somática (não-germinativa) contém 23 pares de cromossomos.
- cromossomo: estrutura nucleoprotéica formada por uma cadeia de DNA, sendo a base física dos genes nucleares, os quais estão dispostos linearmente. Cada espécie apresenta um número característico de cromossomos. Quando os cromossomos são arranjados em pares (cada cromossomo proveniente de um dos pais), os respectivos organismos são chamados diplóides. Organismos cujos cromossomos não se apresentam aos pares são chamados haplóides.
- *crossover* (recombinação): consiste na troca (evento aleatório) de material genético entre dois cromossomos.
- genes: blocos funcionais de DNA, os quais codificam uma proteína específica. É a denominação que damos hoje ao fator mendeliano. Cada gene está localizado em uma posição (*locus*) particular do cromossomo. Quando dois genes se comportam segundo a 1ª lei de Mendel, são ditos alelos, e se encontram no mesmo *locus* de dois cromossomos homólogos.
- genoma: como muitos organismos apresentam células com mais de um cromossomo, o genoma é o conjunto de todos os cromossomos que compõem o material genético do organismo.

4.5 Fenótipo × Genótipo[1]

Indivíduos e espécies podem ser vistos como uma dualidade entre seu código genético (*genótipo*) e suas características comportamentais, fisiológicas e morfológicas (*fenótipo*) (FOGEL, 1994). Em sistemas evoluídos naturalmente, não existe uma relação biunívoca entre um gene (elemento do genótipo) e uma característica (elemento do fenótipo): um único gene pode afetar diversos traços fenotípicos simultaneamente (*pleiotropia*) e uma única característica fenotípica pode ser determinada pela interação de vários genes (*poligenia*). Os efeitos de pleiotropia e poligenia geralmente tornam os resultados de variações genéticas imprevisíveis. Sistemas naturais em evolução são fortemente pleiotrópicos e altamente poligênicos (HARTL & CLARK, 1989). O mesmo não ocorre em sistemas artificiais, onde uma das principais preocupações é com o custo computacional do sistema. Assim, em sistemas artificiais, existe uma relação de um-pra-um entre genótipo e fenótipo.

O processo de evolução pode ser formalizado como segue (ATMAR, 1994; FOGEL, 1999): considere dois espaços distintos – um espaço de estados genotípico (de codificação) **G** e um espaço fenotípico (comportamental) **F**. Considere também um alfabeto de entrada composto de símbolos provenientes do ambiente **I**.

O processo de evolução de uma população em uma geração encontra-se esquematizado na figura 2. Existem 4 mapeamentos atuando neste processo:

[1] Esta seção é uma adaptação autorizada de textos publicados em IYODA (2000), capítulo 3.

$$f_1 : \mathbf{I} \times \mathbf{G} \rightarrow \mathbf{F},$$
$$f_2 : \mathbf{F} \rightarrow \mathbf{F},$$
$$f_3 : \mathbf{F} \rightarrow \mathbf{G},$$
$$f_4 : \mathbf{G} \rightarrow \mathbf{G}.$$

O mapeamento f_1, denominada *epigênese*, mapeia elementos $g_1 \in \mathbf{G}$ em uma coleção particular de fenótipos p_1 do espaço fenotípico **F**, cujo desenvolvimento é modificado por seu ambiente, um conjunto de símbolos $\{i_1, \ldots, i_k\} \in \mathbf{I}$. Este mapeamento é inerentemente de muitos-pra-um, pois existe uma infinidade de genótipos que podem resultar num mesmo fenótipo; elementos de um conjunto infinito de códigos não-expressos (não-participantes na produção do fenótipo) podem existir em g_1 (ATMAR, 1994).

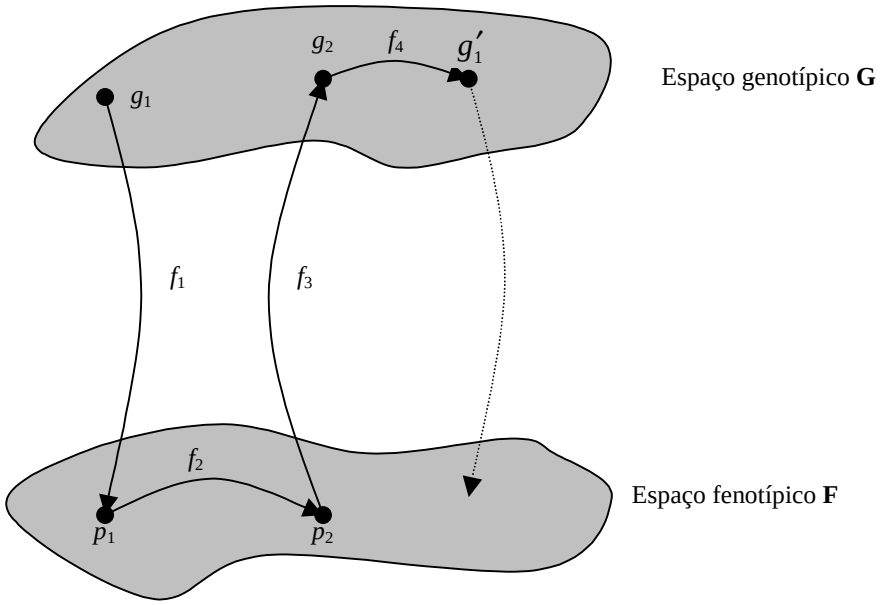


Figura 2: Evolução de uma população durante uma geração.

O mapeamento f_2, *seleção*, mapeia fenótipos p_1 em p_2. Este mapeamento descreve os processos de seleção e migração de indivíduos dentro da população local. Como a seleção natural opera apenas nas expressões fenotípicas do genótipo, o código g_1 não está envolvido no mapeamento f_2. ATMAR (1994) enfatiza que a seleção atua apenas no sentido de eliminar as variantes comportamentais menos apropriadas do inevitável excesso da população, já que assume-se aqui que os recursos provenientes do ambiente são limitados, exigindo a competição pela sobrevivência. Neste processo de competição, a seleção nunca opera sobre uma característica simples, isoladamente do conjunto comportamental.

O mapeamento f_3, *representação* (ATMAR, 1994) ou *sobrevivência genotípica* (FOGEL, 1999), descreve os efeitos dos processos de seleção e migração em **G**.

O mapeamento f_4, *mutação e recombinação*, mapeia códigos $g_2 \in \mathbf{G}$ em $g_1' \in \mathbf{G}$. Este mapeamento descreve as "regras" de mutação e recombinação, e abrange todas as alterações genéticas. A mutação é um erro de cópia no processo de transmissão do código genético dos pais para a sua prole. Em um universo com diferencial de entropia positivo, erros de replicação são inevitáveis e a otimização evolutiva torna-se inevitável em qualquer população que se reproduz em uma arena limitada (ATMAR, 1994).

Com a criação da nova população de genótipos g_1', uma geração está completa. A adaptação evolutiva ocorre em sucessivas iterações destes mapeamentos.

O biólogo Sewell Wright propôs, em 1931, o conceito de superfície de adaptação para descrever nível de adaptação de indivíduos e espécies (FOGEL, 1999). Uma população de genótipos é mapeada em seus respectivos fenótipos que por sua vez são mapeados na superfície de adaptação. Cada pico (máximo local) da superfície de adaptação corresponde a uma coleção de fenótipos otimizada, e portanto a um ou mais conjuntos de genótipos otimizados. A evolução é um processo que conduz, de forma probabilística, populações em direção a picos da superfície, enquanto que a seleção elimina variantes fenotípicas menos apropriadas. Outros pesquisadores propõem uma visão invertida da superfície de adaptação: populações avançam descendo picos da superfície de adaptação até que um ponto de mínimo seja encontrado.

Qualquer que seja o ponto de vista, a evolução é inerentemente um processo de otimização. A seleção produz fenótipos tão próximos do ótimo quanto possível, dadas condições iniciais, restrições ambientais e parâmetros evolutivos. Observe, no entanto, que em sistemas biológicos reais, não existem superfícies de adaptação estáticas. O ambiente está em constante mudança, fazendo com que populações estejam em constante evolução em direção a novos pontos de ótimo. Neste caso, assumindo que as mudanças ambientais são significativas, a taxa evolutiva deve ser suficientemente elevada para acompanhar as mudanças ambientais.

5 A computação evolutiva[2]

Os sistemas baseados em computação evolutiva mantêm uma população de soluções potenciais, aplicam processos de seleção baseados na adaptação de um indivíduo e também empregam outros operadores "genéticos". Diversas abordagens para sistemas baseados em evolução foram propostas, sendo que as principais diferenças entre elas dizem respeito aos operadores genéticos empregados, que serão melhor detalhados na seqüência. As principais abordagens propostas na literatura são:

- algoritmos genéticos;
- estratégias evolutivas;
- programação evolutiva.

Os *algoritmos genéticos* foram introduzidos por Holland em 1975 (HOLLAND, 1975) com o objetivo de formalizar matematicamente e explicar rigorosamente processos de adaptação em sistemas naturais e desenvolver sistemas artificiais (simulados em computador) que retenham os mecanismos originais encontrados em sistemas naturais. Os algoritmos genéticos empregam os operadores de *crossover* e mutação (a serem apresentados mais adiante).

Uma extensão dos algoritmos genéticos, denominada *programação genética*, foi introduzida por KOZA (1992)[3], e tem por objetivo básico evoluir programas de computador usando os princípios da evolução natural. Atualmente a programação genética tem sido aplicada a uma grande variedade de problemas, como na síntese de circuitos elétricos analógicos (KOZA *et al.*, 1997) e na definição de arquiteturas de redes neurais artificiais (GRUAU, 1994).

Estratégias evolutivas (RECHENBERG, 1973; SCHWEFEL, 1995) foram inicialmente propostas com o objetivo de solucionar problemas de otimização de parâmetros, tanto

[2] Esta seção é uma adaptação autorizada de textos publicados em IYODA (2000), capítulo 3.
[3] J. R. Koza detém uma patente sobre programação genética.

discretos como contínuos. Em virtude de empregarem apenas operadores de mutação, grandes contribuições em relação a análise e síntese destes operadores foram elaboradas.

A *programação evolutiva*, introduzida por FOGEL *et al.* (1966), foi originalmente proposta como uma técnica para criar inteligência artificial através da evolução de máquinas de estado finito. A programação evolutiva também emprega apenas mutação. Recentemente, a programação evolutiva tem sido aplicada a problemas de otimização e é, neste caso, virtualmente equivalente às estratégias evolutivas; apenas pequenas diferenças no que diz respeito aos procedimentos de seleção e codificação de indivíduos estão presentes nas duas abordagens atualmente (FOGEL, 1994).

Apesar das abordagens acima citadas terem sido desenvolvidas de forma independente, seus algoritmos possuem uma estrutura comum. Usaremos o termo *algoritmo evolutivo* como uma denominação comum a todas elas. A estrutura de um algoritmo evolutivo pode ser dada na forma (MICHALEWICZ, 1996):

```
procedimento programa evolutivo
início
    t ← 0
    inicialize P(t)
    avalie P(t)
    enquanto (não condição de parada) faça
    início
        t ← t + 1
        selecione P(t) a partir de P(t – 1)
        altere P(t)
        avalie P(t)
    fim
fim
```

Um algoritmo evolutivo mantém uma população de indivíduos $P(t) = \{x_1^t, \ldots, x_n^t\}$ na iteração (geração) t. Cada indivíduo representa um candidato à solução do problema em questão e, em qualquer implementação computacional, assume a forma de alguma estrutura de dados S. Cada solução x_i^t é avaliada e produz alguma medida de adaptação, ou *fitness*. Então, uma nova população é formada na iteração $t + 1$ pela seleção dos indivíduos mais adaptados. Alguns indivíduos da população são submetidos a um processo de alteração por meio de operadores genéticos para formar novas soluções. Existem transformações unárias m_i (mutação) que criam novos indivíduos através de pequenas modificações de atributos em um indivíduo ($m_i : S \rightarrow S$), e transformações de ordem superior c_j (*crossover*), que criam novos indivíduos através da combinação de dois ou mais indivíduos ($c_j : S \times \ldots \times S \rightarrow S$). Após um número de gerações, a condição de parada deve ser atendida, a qual geralmente indica a existência, na população, de um indivíduo que represente uma solução aceitável para o problema, ou quando o número máximo de gerações foi atingido.

As abordagens evolutivas apresentadas nesta seção diferem em diversos aspectos, dentre os quais se destacam: estruturas de dados utilizadas para codificar um indivíduo, operadores genéticos empregados, métodos para criar a população inicial e métodos para selecionar indivíduos para a geração seguinte. Entretanto, elas compartilham o mesmo princípio comum: uma população de indivíduos sofre algumas transformações e durante a evolução os indivíduos competem pela sobrevivência. Iremos nos concentrar a seguir na descrição mais detalhada dos algoritmos genéticos.

5.1 Algoritmos Genéticos

Os algoritmos genéticos empregam uma terminologia originada da teoria da evolução natural e da genética. Um indivíduo da população é representado por um único cromossomo, o qual contém a codificação (genótipo) de uma possível solução do problema (fenótipo). Cromossomos são usualmente implementados na forma de listas de atributos ou vetores, onde cada atributo é conhecido como *gene*. Os possíveis valores que um determinado gene pode assumir são denominados *alelos*.

O processo de evolução executado por um algoritmo genético corresponde a um procedimento de busca em um espaço de soluções potenciais para o problema. Como enfatiza MICHALEWICZ (1996), esta busca requer um equilíbrio entre dois objetivos aparentemente conflitantes: o aproveitamento das melhores soluções e a exploração do espaço de busca (*exploitation* × *exploration*). Este equilíbrio está muito longe de ocorrer quando se considera outras técnicas de busca:

- métodos de otimização clássicos, como o método do gradiente, são exemplos de métodos que apenas aproveitam a melhor solução na busca de possíveis aprimoramentos, sem realizar uma exploração do espaço de busca.
- métodos de busca aleatória, por sua vez, são exemplos típicos de métodos que exploram o espaço de busca ignorando o aproveitamento de regiões promissoras do espaço.

Algoritmos genéticos constituem, assim, uma classe de métodos de busca de propósito geral que apresentam um balanço notável entre aproveitamento de melhores soluções e exploração do espaço de busca. Embora apresentem etapas não-determinísticas em seu desenvolvimento, os algoritmos genéticos não são métodos de busca puramente aleatórios, pois combinam variações aleatórias com seleção, polarizada pelos valores de adequação (*fitness*) atribuído a cada indivíduo. Outra propriedade importante dos algoritmos genéticos (assim como de todos os algoritmos evolutivos) é que eles mantêm uma população de soluções candidatas enquanto que os métodos alternativos, como *simulated annealing* (AARTS & KORST, 1989), processam um único ponto no espaço de busca a cada instante.

O processo de busca é, portanto, multi-direcional, através da manutenção de soluções candidatas, e encorajando a troca de informação entre as direções. A cada geração, soluções relativamente "boas" se reproduzem, enquanto que soluções relativamente "ruins" são eliminadas. Para fazer a distinção entre diferentes soluções, é empregada uma função de avaliação ou de adaptabilidade (*fitness*) que simula o papel da pressão exercida pelo ambiente sobre o indivíduo. Seguindo a estrutura básica de um algoritmo evolutivo, podemos descrever um algoritmo genético como segue (MICHALEWICZ, 1996):

- durante a iteração t, um algoritmo genético mantém uma população de soluções potenciais (indivíduos, cromossomos, lista de atributos ou vetores) $P(t) = \left\{\mathbf{x}_1^t, \ldots, \mathbf{x}_n^t\right\}$;
- cada solução $\mathbf{x}_i^t$ é avaliada e produz uma medida de sua adaptação, ou *fitness*;
- uma nova população (iteração $t+1$) é então formada privilegiando a participação dos indivíduos mais adaptados
- alguns membros da nova população passam por alterações, por meio de *crossover* e mutação, para formar novas soluções potenciais;
- este processo se repete até que um número pré-determinado de iterações seja atingido, ou até que um nível de adaptação esperado seja alcançado.

Um algoritmo genético para um problema particular deve ter os seguintes componentes:

- uma representação genética para soluções candidatas ou potenciais (processo de codificação);
- uma maneira de criar uma população inicial de soluções candidatas ou potenciais;
- uma função de avaliação que faz o papel da pressão ambiental, classificando as soluções em termos de sua adaptação ao ambiente (ou seja, sua capacidade de resolver o problema);
- operadores genéticos;
- valores para os diversos parâmetros usados pelo algoritmo genético (tamanho da população, probabilidades de aplicação dos operadores genéticos, etc.).

5.1.1 Codificação de Indivíduos

Cada indivíduo de uma população representa um candidato em potencial à solução do problema em questão. No algoritmo genético clássico, proposto por HOLLAND (1975), as soluções candidatas são codificadas em arranjos binários de tamanho fixo. A motivação para o uso de codificação binária vem da teoria dos esquemas (*schemata theory*), utilizada com relativo sucesso para explicar por que os algoritmos genéticos funcionam. HOLLAND (1992) argumenta que seria benéfico para o desempenho do algoritmo maximizar o paralelismo implícito inerente ao algoritmo genético, e prova que um alfabeto binário maximiza o paralelismo implícito.

Entretanto, em diversas aplicações práticas a utilização de codificação binária leva a um desempenho insatisfatório. Em problemas de otimização numérica com parâmetros reais, algoritmos genéticos com representação inteira ou em ponto flutuante freqüentemente apresentam desempenho superior à codificação binária. MICHALEWICZ (1996) argumenta que a representação binária apresenta desempenho pobre quando aplicada a problemas numéricos com alta dimensionalidade e onde alta precisão é requerida. Suponha por exemplo, que temos um problema com 100 variáveis com domínio no intervalo [−500, 500] e que precisamos de 6 dígitos de precisão após a casa decimal. Neste caso precisaríamos de um cromossomo de comprimento 3000, e teríamos um espaço de busca de dimensão aproximadamente 10^{1000}. Neste tipo de problema o algoritmo genético clássico apresenta desempenho pobre. MICHALEWICZ (1996) apresenta também simulações computacionais comparando o desempenho de algoritmos genéticos com codificação binária e com ponto flutuante, aplicados a um problema de controle. Os resultados apresentados mostram uma clara superioridade da codificação em ponto flutuante.

A argumentação de MICHALEWICZ (1996), de que o desempenho de um algoritmo genético com codificação binária é pobre quando o espaço de busca é de dimensão elevada, não é universalmente aceita na literatura referente a algoritmos genéticos. FOGEL (1994) argumenta que o espaço de busca por si só (sem levar em conta a escolha da representação) não determina a eficiência do algoritmo genético. Espaços de busca de dimensão elevada podem às vezes ser explorados eficientemente, enquanto que espaços de busca de dimensão reduzida podem apresentar dificuldades significativas. FOGEL (1994), entretanto, concorda que a maximização do paralelismo implícito nem sempre produz um desempenho ótimo.

Fica claro, portanto, que a codificação é uma das etapas mais críticas na definição de um algoritmo genético. A definição inadequada da codificação pode levar a problemas de convergência prematura do algoritmo genético. A estrutura de um cromossomo deve representar uma solução como um todo, e deve ser a mais simples possível.

Em problemas de otimização restrita, a codificação adotada pode fazer com que indivíduos modificados por *crossover*/mutação sejam inválidos. Nestes casos, cuidados especiais devem ser tomados na definição da codificação e/ou dos operadores.

5.1.2 Definição da População Inicial

O método mais comum utilizado na criação da população é a inicialização aleatória dos indivíduos. Se algum conhecimento inicial a respeito do problema estiver disponível, pode ser utilizado na inicialização da população. Por exemplo, no caso de codificação binária, se é sabido que a solução final vai apresentar mais 0's do que 1's, então esta informação pode ser utilizada, mesmo que não se saiba exatamente a proporção. Já em problemas com restrição, deve-se tomar cuidado para não gerar indivíduos inválidos na etapa de inicialização.

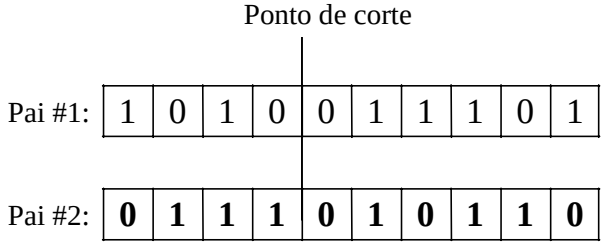
5.1.3 Operadores Genéticos

Os operadores genéticos mais freqüentemente utilizados em algoritmos genéticos são o *crossover* e a mutação. Nesta seção, apresentamos os principais aspectos relacionados a estes operadores.

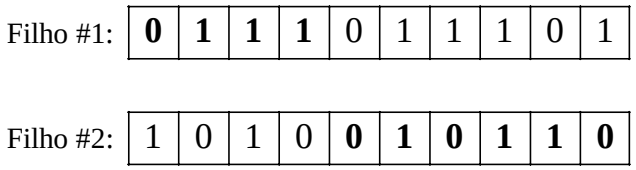
5.1.3.1 O Operador de *Crossover*

O operador de *crossover* ou recombinação cria novos indivíduos através da combinação de dois ou mais indivíduos. A idéia intuitiva por trás do operador de *crossover* é a troca de informação entre diferentes soluções candidatas. No algoritmo genético clássico é atribuída uma probabilidade de *crossover* fixa aos indivíduos da população.

O operador de *crossover* mais comumente empregado é o *crossover* de um ponto. Para a aplicação deste operador, são selecionados dois indivíduos (pais) e a partir de seus cromossomos são gerados dois novos indivíduos (filhos). Para gerar os filhos, seleciona-se um mesmo ponto de corte aleatoriamente nos cromossomos dos pais, e os segmentos de cromossomo criados a partir do ponto de corte são trocados. Considere, por exemplo, dois indivíduos selecionados como pais a partir da população inicial de um algoritmo genético e suponhamos que o ponto de corte escolhido (aleatoriamente) encontra-se entre as posições 4 e 5 dos cromossomos dos pais:



Após o *crossover*, teremos os seguintes indivíduos-filho:



Muitos outros tipos de *crossover* têm sido propostos na literatura. Uma extensão simples do *crossover* de um ponto é o *crossover* de dois pontos, onde dois pontos de corte são escolhidos e material genético são trocados entre eles. Outro tipo de *crossover* muito comum é o *crossover uniforme* (SYSWERDA, 1989): para cada bit no primeiro filho é decidido (com alguma probabilidade fixa *p*) qual pai vai contribuir com seu valor para aquela posição. Como o *crossover* uniforme troca bits ao invés de segmentos de bits (que aqui fazem o papel dos genes), ele pode combinar características independentemente da sua posição relativa no cromossomo. No entanto, não há nenhum operador de *crossover* que claramente apresente um

desempenho superior aos demais. Uma conclusão a que se pode chegar é que cada operador de *crossover* é particularmente eficiente para uma determinada classe de problemas e extremamente ineficiente para outras.

Os operadores de *crossover* descritos até aqui também podem ser utilizados em cromossomos com codificação em ponto flutuante. Entretanto existem operadores de *crossover* especialmente desenvolvidos para uso com codificação em ponto flutuante. Um exemplo é o chamado *crossover aritmético* (MICHALEWICZ, 1996). Este operador é definido como uma combinação linear de dois vetores (cromossomos): sejam $\mathbf{x}_1$ e $\mathbf{x}_2$ dois indivíduos selecionados para *crossover*, então os dois filhos resultantes serão $\mathbf{x}'_1 = a\mathbf{x}_1 + (1-a)\mathbf{x}_2$ e $\mathbf{x}'_2 = (1-a)\mathbf{x}_1 + a\mathbf{x}_2$ onde a é um número aleatório pertencente ao intervalo [0, 1]. Este operador é particularmente apropriado para problemas de otimização numérica com restrições, onde a região factível é convexa. Isto porque, se $\mathbf{x}_1$ e $\mathbf{x}_2$ pertencem à região factível, combinações convexas de $\mathbf{x}_1$ e $\mathbf{x}_2$ serão também factíveis. Assim, garante-se que o *crossover* não gera indivíduos inválidos para o problema em questão.

5.1.3.2 O Operador de Mutação

O operador de mutação modifica aleatoriamente um ou mais genes de um cromossomo. A probabilidade de ocorrência de mutação em um gene é denominada *taxa de mutação*. Usualmente, são atribuídos valores pequenos para a taxa de mutação. A idéia intuitiva por trás do operador de mutação é criar uma variabilidade extra na população, mas sem destruir o progresso já obtido com a busca.

Considerando codificação binária, o operador de mutação padrão simplesmente troca o valor de um gene em um cromossomo (HOLLAND, 1992). Assim, se um gene selecionado para mutação tem valor 1, o seu valor passará a ser 0 após a aplicação da mutação, e vice-versa.

No caso de problemas com codificação em ponto flutuante, os operadores de mutação mais populares são a mutação uniforme e a mutação gaussiana (MICHALEWICZ & SCHOENAUER, 1996). O operador para mutação uniforme seleciona aleatoriamente um componente $k \in \{1, 2, \ldots, n\}$ do cromossomo $\mathbf{x} = [x_1 \ldots x_k \ldots x_n]$ e gera um indivíduo $\mathbf{x}' = [x_1 \ldots x'_k \ldots x_n]$, onde x'_k é um número aleatório (com distribuição de probabilidade uniforme) amostrado no intervalo [*LB*, *UB*] e *LB* e *UB* são, respectivamente, os limites inferior e superior para o valor do alelo x_k. Já no caso da mutação gaussiana, todos os componentes de um cromossomo $\mathbf{x} = [x_1 \ldots x_n]$ são modificados na forma:

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

onde $N(0, \sigma)$ é um vetor de variáveis aleatórias gaussianas independentes, com média zero e desvio padrão σ. Outro operador de mutação, especialmente desenvolvido para problemas de otimização com restrições e codificação em ponto flutuante, é a chamada mutação não-uniforme, destinada a realizar a sintonia fina junto aos indivíduos da população. Este e outros exemplos de operadores de mutação para problemas de otimização numérica podem ser encontrados em MICHALEWICZ (1996) e MICHALEWICZ & SCHOENAUER (1996).

5.1.4 Seleção de Indivíduos para a Próxima Geração

O algoritmo genético clássico utiliza um esquema de seleção de indivíduos para a próxima geração chamado *roulette wheel* (MICHALEWICZ, 1996). O *roulette wheel* atribui a cada indivíduo de uma população uma probabilidade de passar para a próxima geração proporcional ao seu *fitness* medido, em relação à somatória do *fitness* de todos os indivíduos da população. Assim, quanto maior o *fitness* de um indivíduo, maior a probabilidade dele

passar para a próxima geração. Sendo assim, a seleção de indivíduos por *roulette wheel* pode fazer com que o melhor indivíduo da população seja perdido, ou seja, não passe para a próxima geração. Uma alternativa é escolher como solução o melhor indivíduo encontrado em todas as gerações do algoritmo. Outra opção é simplesmente manter sempre o melhor indivíduo da geração atual na geração seguinte, estratégia essa conhecida como seleção elitista (FOGEL, 1994; MICHALEWICZ, 1996).

Outro exemplo de mecanismo de seleção é a seleção baseada em rank (BÄCK *et al.*, 1997). Esta estratégia utiliza as posições dos indivíduos quando ordenados de acordo com o *fitness* para determinar a probabilidade de seleção. Podem ser usados mapeamentos lineares ou não-lineares para determinar a probabilidade de seleção. Para um exemplo de mapeamento não-linear, veja MICHALEWICZ (1996). Uma variação deste mecanismo é simplesmente passar os *N* melhores indivíduos para a próxima geração.

A seguir, citamos alguns outros possíveis mecanismos de seleção:

- Seleção por diversidade: são selecionados os indivíduos mais diversos da população.
- Seleção bi-classista: são selecionados os *P*% melhores indivíduos e os (100 – *P*)% piores indivíduos.
- Seleção aleatória: são selecionados aleatoriamente *N* indivíduos da população. Podemos subdividir este mecanismo de seleção em:
  - Salvacionista: seleciona-se o melhor indivíduo e os outros aleatoriamente.
  - Não-salvacionista: seleciona-se aleatoriamente todos os indivíduos.

Estes mesmos mecanismos de seleção podem ser adaptados para selecionar também os indivíduos que irão sofrer *crossover* e mutação. Por exemplo, usando a seleção bi-classista, é possível selecionar os indivíduos que, ao se reproduzirem, irão gerar os indivíduos da próxima geração. Como será visto em um estudo de caso, o número de indivíduos selecionados para *crossover* pode ser bem menor que o total de indivíduos da população, indicando que só alguns terão chance de gerar descendentes, e em grande número.

6 Estudo de casos

Serão tratados a seguir dois problemas de otimização para os quais não existem soluções dedicadas que garantam a obtenção da solução ótima utilizando uma quantidade viável de recursos computacionais. Ambos os enunciados poderiam ser enriquecidos com restrições adicionais e novos requisitos de forma a produzir problemas ainda mais complexos, mas optou-se por mantê-los o mais simples possível, por razões didáticas.

O aspecto essencial a ser levando em conta é o fato de que, mesmo que hoje existam soluções dedicadas mais eficazes do que as que serão apresentadas com base em técnicas de computação evolutiva, estas soluções dedicadas não são facilmente extensíveis a novas situações e certamente seria necessário uma solução dedicada para cada problema. Via computação evolutiva, estaremos empregando o mesmo algoritmo evolutivo, mais especificamente um algoritmo genético, para ambos os problemas, tomando o cuidado apenas de definir apropriadamente a representação (codificação genética) a ser adotada em cada caso, assim como os operadores genéticos correspondentes.

6.1 Otimização de parâmetros de uma caixa preta

Considere o seguinte problema de otimização:

Sabendo que cada botão pode ser colocado em 16 posições distintas, encontre a melhor combinação de posições para os 9 botões disponíveis na superfície da caixa preta apresentada na figura 3 de modo que o sinal de saída assuma o valor máximo.

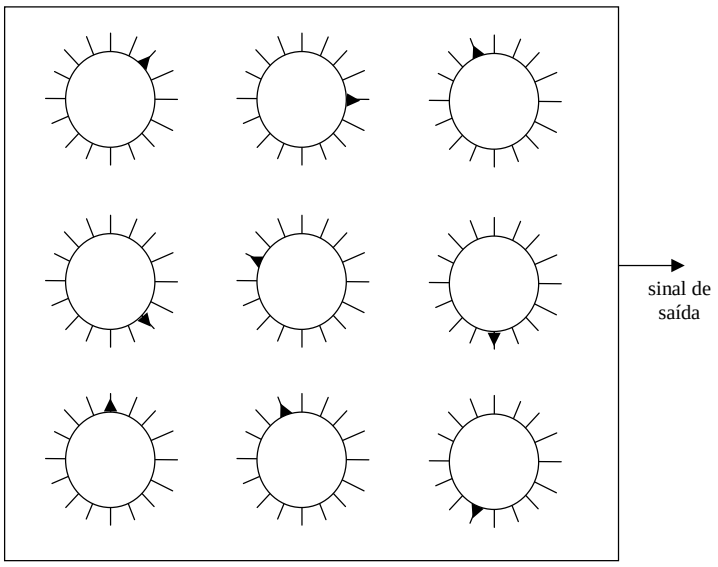
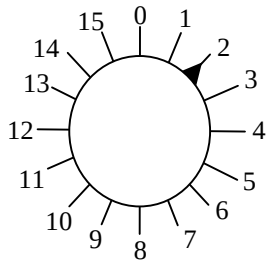


Figura 3 - Problema de otimização combinatória: definir as posições dos 9 botões de tal modo que o sinal de saída seja máximo

Codificação

- representação binária: existem 16 posições possíveis para cada um dos 9 botões, de modo que 4 bits são suficientes para representar cada uma das 16 posições, na forma:



Posição	Representação	Posição	Representação
0	0000	8	1000
1	0001	9	1001
2	0010	10	1010
3	0011	11	1011
4	0100	12	1100
5	0101	13	1101
6	0110	14	1110
7	0111	15	1111

Posição atual: 0010

- cromossomo associado à solução candidata apresentada na figura 3:

  001001001111011011011000000011111001
- número de possíveis configurações de botões (soluções candidatas): $2^{36} \cong 68{,}72*10^{9}$
- operadores genéticos: mutação simples e *crossover* uniforme
- mapeamento, assumido desconhecido, entre as 2^{36} posições possíveis dos botões e o sinal de saída:

$$\text{sinal de saída} = 9 + b_2b_5 - b_{23}b_{14} + b_{24}b_4 - b_{21}b_{10} + b_{36}b_{15} - b_{11}b_{26} + b_{16}b_{17} + b_3b_{33} + b_{28}b_{19} + b_{12}b_{34} - b_{31}b_{32} - b_{22}b_{25} + b_{35}b_{27} - b_{29}b_7 + b_8b_{13} - b_6b_9 + b_{18}b_{20} - b_1b_{30} + b_{23}b_4 + b_{21}b_{15} + b_{26}b_{16} + b_{31}b_{12} + b_{25}b_{19} + b_7b_8 + b_9b_{18} + b_1b_{33}$$

  Esta vai ser a função de adaptação ou fitness, a ser maximizada. Esta função de mapeamento é assumida desconhecida (razão para a denominação de caixa preta), sendo que só se tem acesso ao valor do sinal de saída para cada configuração de botões (associada à seqüência de bits do cromossomo).

- solução ótima (determinada a partir da análise da função que produz o sinal de saída, portanto também assumida desconhecida):

  11111011100110111111101111110010111 1

  com um valor de fitness de 27.
- valores arbitrados pelo usuário:
  - probabilidade de bits 1 nos cromossomos da população inicial: 50%
  - taxa de mutação: 3%
  - taxa de *crossover*: 60%
  - tipo de seleção para aplicação de *crossover*: bi-classista (50% dos melhores e 10% dos piores indivíduos)
- resultados obtidos (observe que foi possível encontrar a solução ótima):

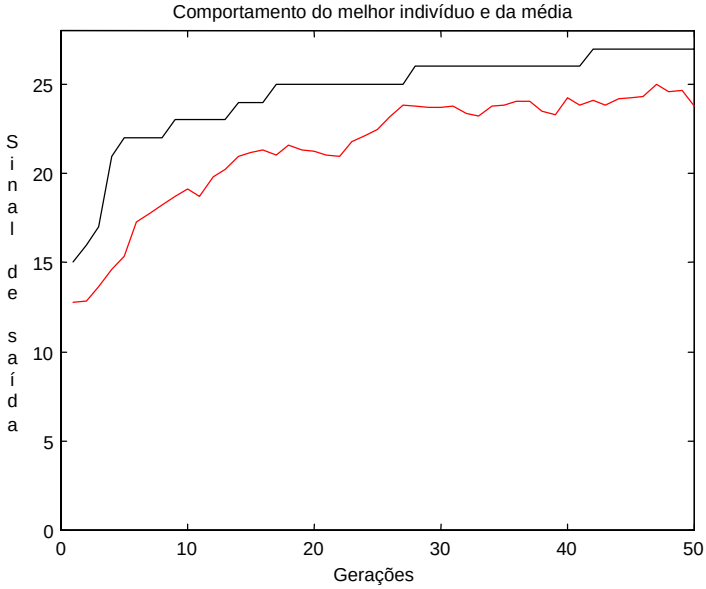


Figura 4 - Resultado de desempenho do processo evolutivo

- número de indivíduos testados: 1500 (dentre os possíveis $68{,}72 \times 10^9$ candidatos)
- tempo de simulação em um Pentium III 450 MHz: 0,38 segundos

6.2 Problema do Caixeiro Viajante

Considere o seguinte problema de otimização (a escolha do número de 100 cidades foi feita simplesmente para tornar o problema intratável. A solução ótima para este problema não é conhecida):

> *Suponha que um caixeiro deva partir de sua cidade, visitar clientes em outras 99 cidades diferentes, e então retornar à sua cidade. Dadas as coordenadas das 100 cidades, descubra o percurso de menor distância que passe uma única vez por todas as cidades e retorne à cidade de origem.*

Codificação

- representação inteira: cada cromossomo conterá todos os números de 1 a 100 (cada número associado a uma cidade, e a ordem de aparecimento dos números no cromossomo vai indicar o percurso, sendo necessário fechar o percurso da última para a primeira cidade. Detalhe: como trata-se de um percurso fechado, a origem do percurso pode ser qualquer uma das cidades, ao menos para efeito da implementação computacional.

- número de possíveis percursos (soluções candidatas): $99! \cong 9{,}33 \times 10^{155}$
- função de adequação (*fitness*): o inverso da distância associada a cada percurso.
- solução ótima: desconhecida, em razão da impossibilidade de testar todas as soluções candidatas (único meio existente para se garantir a obtenção da solução ótima);
- valores arbitrados pelo usuário:
  - tipo de mutação: sorteio de duas cidades para troca de posição
  - taxa de mutação: 1%
  - tipo de *crossover*: OX (uma espécie de *crossover* de um ponto, caracterizado pela junção de uma parte de um cromossomo com a parte de um outro, mas com a substituição das cidades repetidas pelas ausentes, na seqüência)
  - taxa de *crossover*: 60%
  - tipo de seleção: rank ou torneio (50% dos melhores)
- resultados obtidos:

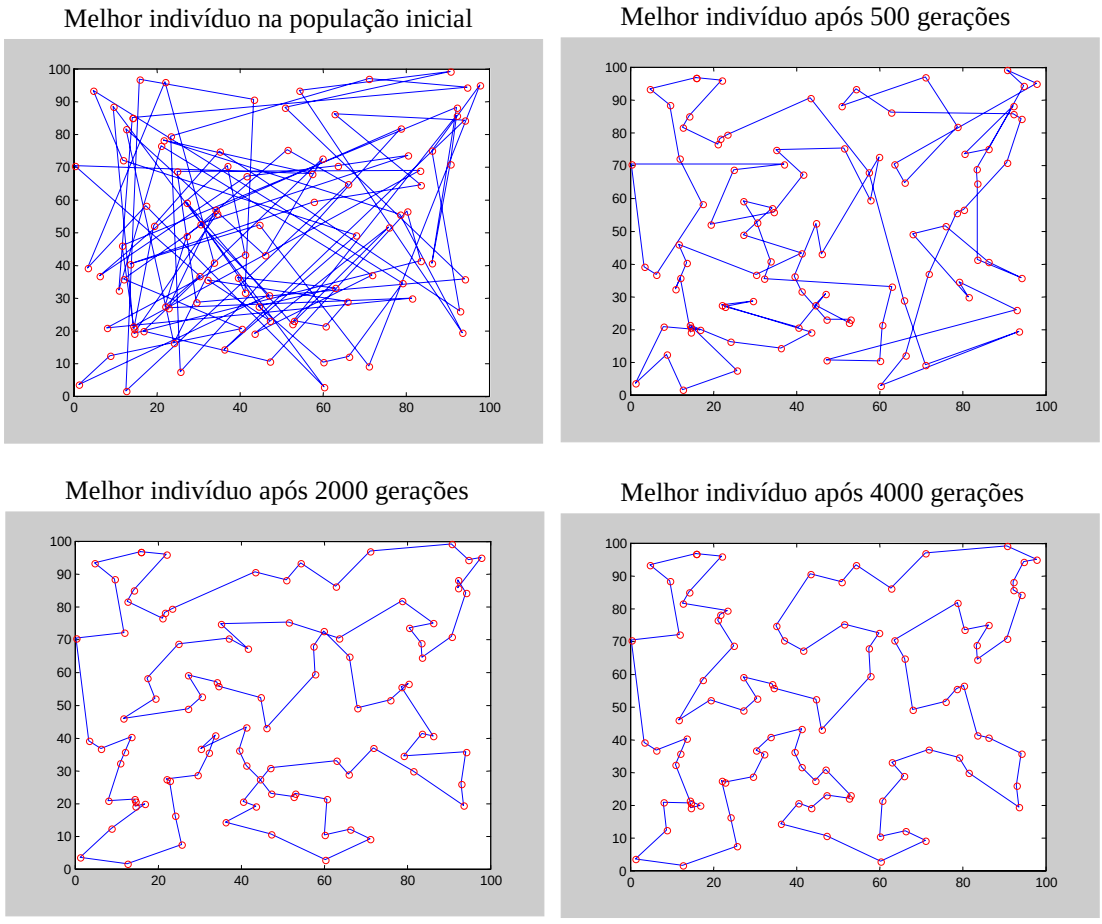


Figura 5 - Resultado do processo evolutivo

- número de indivíduos testados: 400000 (dentre os possíveis $9{,}33 \times 10^{155}$ candidatos)
- tempo de simulação em um Pentium III 450 MHz: 287 segundos

7 Tópicos avançados

Alguns temas relacionados à computação evolutiva, mencionados ou não ao longo do texto, merecem um estudo à parte, e serão apenas mencionados a seguir de modo a alertar para as contribuições que progressos junto a estes tópicos podem trazer para a área de computação evolutiva, e vice-versa:

- geradores de números aleatórios;
- diversidade populacional;
- operadores adaptativos;
- complexidade computacional;
- busca local;
- co-evolução;
- problemas de otimização multi-objetivo e com múltiplas restrições;
- superfícies de adaptação não-estacionárias;
- abordagem de processos criativos em computador via computação evolutiva.

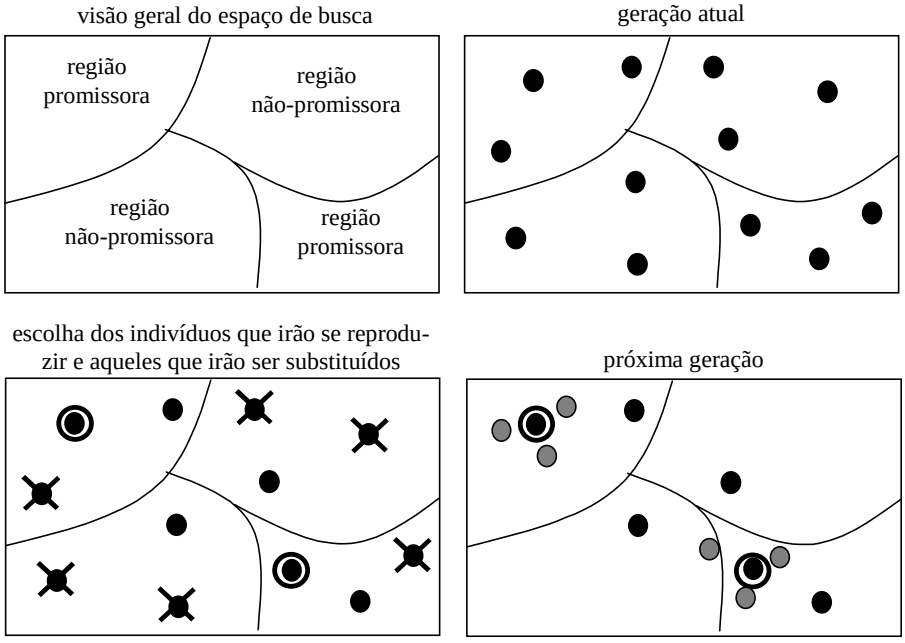


Figura 6 - Visão pictórica da força evolutiva

8 Conclusões

A computação evolutiva representa uma iniciativa de implementação computacional de regras de evolução muito simples: os indivíduos sofrem variações aleatórias de geração em geração (via operadores genéticos responsáveis pela implementação dos processos de busca), estando sujeitos à seleção natural sob recursos limitados. Os indivíduos mais adaptados (exemplares localizados em regiões promissoras do espaço de busca) sobrevivem e se reproduzem, propagando seu material genético às próximas gerações. Uma visão pictórica da força evolutiva está ilustrada na figura 6, onde existem regiões promissoras e não-promissoras no espaço de busca. Assumindo que o tamanho da população é fixo ao longo das gerações (simulação da existência de recursos limitados), então os indivíduos mais adaptados da geração atual terão maior chance de transmitir seu código genético para a próxima geração. Existe portanto um compromisso entre mecanismos de exploração global do espaço de busca (manutenção de diversidade na população) e de exploração local das regiões promissoras detectadas (aplicação de operadores genéticos em conjunto com processos de seleção baseadas no nível de adaptação de cada indivíduo).

Este tutorial procurou formalizar os principais conceitos envolvidos no estudo e aplicação da computação evolutiva, privilegiando aspectos computacionais e procurando ilustrar o potencial de aplicação com enfoque na solução de problemas de otimização via

algoritmos genéticos. Foram apresentados dois exemplos de aplicação, envolvendo um problema de ajuste de parâmetros em sistemas estacionários (no caso, configuração de botões em uma caixa preta) e um problema de otimização combinatória, sabido ser intratável a partir do uso de técnicas de busca que garantam a obtenção da solução ótima (no caso, o problema do caixeiro viajante).

Se houve alguma preocupação constante ao longo do texto, esta foi certamente a iniciativa de mostrar que existem algoritmos superiores aos algoritmos evolutivos para resolver certos tipos de problemas. No entanto, para se chegar a estas soluções dedicadas e muito eficientes, esforços incalculáveis foram devotados e as técnicas então elaboradas estão fortemente vinculadas a conhecimentos específicos associados a cada problema, ou seja, representam soluções dedicadas. Há o sacrifício da generalidade para um ganho em desempenho, caracterizando o método de solução como forte. Perante um método forte de solução, a computação evolutiva não é competitiva e portanto não deve ser considerada como alternativa. Mas a quantidade de problemas de interesse que hoje admitem uma solução a partir de métodos fortes é desprezível diante da quantidade de problemas para os quais não existe ainda uma solução dedicada. E é este o reino da computação evolutiva, tão vasto que as suas fronteiras se estendem além de qualquer horizonte que possa ser vislumbrado.

9 Sugestão para leituras complementares

- introdução às técnicas de geração de algoritmos evolutivos devotados para solução de problemas do mundo real e de interesse prático, e comparação com técnicas clássicas de otimização, podem ser encontradas em MICHALEWICZ & FOGEL (2000);
- avanços recentes em computação evolutiva podem ser encontrados em conferências de abrangência internacional, dentre as quais se destacam a *IEEE International Conference on Evolutionary Computation* e a *Genetic and Evolutionary Computation Conference*;
- já existem também periódicos especificamente devotados ao tema da computação evolutiva, dentre os quais se destacam o *IEEE Transactions on Evolutionary Computation*, editado pelo *IEEE* (*The Institute of Electrical and Electronics Engineers*) e o *Evolutionary Computation Journal*, editado por *The MIT Press*;
- buscas através da internet podem conduzir a *sites* interessantes e ricos em informação organizada na forma de hipertextos, mas nem todos os *sites* merecem esta classificação, sendo que o controle de qualidade deve ser feito pelo próprio usuário. Sugestões de palavras-chave para busca estão apresentadas na primeira página deste tutorial;
- é possível participar também de listas de discussão moderadas, muito úteis para se manter informado acerca de eventos já programados e meios de acesso a novas publicações na área, além de acompanhar debates envolvendo questões ainda em aberto. As inscrições podem ser solicitadas através dos seguintes e-mails: GA-List-Request@aic.nrl.navy.mil e EP-List-Request@magenta.me.fau.edu.

10 Referências

AARTS, E. & KORST, J. "Simulated Annealing and Boltzmann Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing", John Wiley & Sons, 1989.

ATMAR, W. "On the Rules and Nature of Simulated Evolutionary Programming", *Proc. of the First Ann. Conf. On Evolutionary Programming*, pp. 17-26, 1992.

ATMAR, W. "Notes on the Simulation of Evolution", *IEEE Transactions on Neural Networks*, vol. 5, no. 1, pp. 130-148, 1994.

BÄCK, T. "Evolutionary Algorithms in Theory and Practice", Oxford University Press, 1996.

BÄCK, T., FOGEL, D.B. & MICHALEWICZ, Z. (eds.) "Handbook of Evolutionary Computation", Institute of Physics Publishing and Oxford University Press, 1997.

BÄCK, T., FOGEL, D.B. & MICHALEWICZ, Z. (eds.) "Evolutionary Computation 1: Basic Algorithms and Operators", Institute of Physics Publishing, 2000a.
BÄCK, T., FOGEL, D.B. & MICHALEWICZ, Z. (eds.) "Evolutionary Computation 2: Advanced Algorithms and Operators", Institute of Physics Publishing, 2000b.
BOOKER, L.B., GOLDBERG, D.E. & HOLLAND, J.H. "Classifier Systems and Genetic Algorithms", *Artificial Intelligence*, vol. 40, pp. 235-282, 1989.
BREMERMANN, H.J. "Optimization through evolution and recombination", *in* M.C. Yovits, G.T. Jacobi & G.D. Goldstine (eds.) *Self-Organizing Systems*, pp. 93-106, Spartan Books, 1962.
BURNS, G.W. & BOTTINO, P.J. "The Science of Genetics", Prentice Hall, 1988.
DARWIN, C. "The Origin of Species", John Murray, 1859 (Penguin Classics, 1985).
DAVIS, L. (ed.) "Handbook of Genetic Algorithms", Van Nostrand Reinhold, 1991.
DE JONG, K.A. "An analysis of the behavior of a class of genetic adaptive systems", Ph.D. Dissertation, University of Michigan, Ann Arbor, 1975.
FOGEL, D.B. "An Introduction to Simulated Evolutionary Computation", *IEEE Transactions on Neural Networks*, vol. 5, no. 1, pp. 3-14, 1994.
FOGEL, D.B. (ed.) "Evolutionary Computation: The Fossil Record", The IEEE Press, 1998.
FOGEL, D.B. "Evolutionary Computation: Toward a New Philosophy of Machine Intelligence", 2nd edition, The IEEE Press, 1999.
FOGEL, L.J. "Autonomous automata", *Industrial Research*, vol. 4, pp. 14-19, 1962.
FOGEL, L.J., OWENS, A.J. & WALSH, M.J. "Artificial Intelligence through Simulated Evolution", Wiley, 1966.
FRASER, A.S. "Simulation of genetic systems by automatic digital computers: I. Introduction", *Austral. J. Biol. Sci.*, vol. 10, pp. 484-491, 1957.
GOLDBERG, D.E. "Genetic Algorithms in Search, Optimization, and Machine Learning", Addison-Wesley, 1989.
GRUAU, F. "Genetic Micro Programming of Neural Networks", *in* Kinnear, K.E. (ed.), *Advances in Genetic Programming*, pp. 495-518, The MIT Press, 1994.
HARTL, D.L. & CLARK, A.G. "Principles of Population Genetics", Sinauer, 1989.
HOLLAND, J.H. "Outline for a logical theory of adaptive systems", *J. Assoc. Comput. Mach.*, vol. 3, pp. 297-314, 1962.
HOLLAND, J.H. "Adaptation in Natural and Artificial Systems", University of Michigan Press, 1975.
HOLLAND, J.H. "Adaptation in Natural and Artificial Systems", 2nd edition, The MIT Press, 1992.
IYODA, E.M. "Inteligência Computacional no Projeto Automático de Redes Neurais Híbridas e Redes Neurofuzzy Heterogêneas", Tese de Mestrado, Faculdade de Engenharia Elétrica e de Computação (Unicamp), 2000.
KINNEAR, K.E. (ed.) "Advances in Genetic Programming", The MIT Press, 1994.
KOZA, J.R. "Genetic Programming: On the Programming of Computers by means of Natural Selection", The MIT Press, 1992.
KOZA, J.R., BENNET III, F.H., ANDRE, D., KEANE, M.A. & DUNLAP, F. "Automated Synthesis of Analog Electrical Circuits by Means of Genetic Programming", *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 2, pp. 109-128, 1997.
MAYR, E. "Toward a New Philosophy of Biology: Observations of an Evolutionist", Belknap Press, 1988.
MICHALEWICZ, Z. "Genetic algorithms + Data Structures = Evolution Programs", 3rd edition, Springer-Verlag, 1996.
MICHALEWICZ, Z. & FOGEL, D. B. "How to solve it: Modern Heuristics", Springer-Verlag, 2000.
MICHALEWICZ, Z. & SCHOENAUER, M. "Evolutionary Algorithms for Constrained Parameter Optimization Problems", *Evolutionary Computation*, vol. 4, no. 1, pp. 1-32, 1996.
MITCHELL, M. "An Introduction to Genetic Algorithms", The MIT Press, 1996.
RECHENBERG, I. "Cybernetic solution path of an experimental problem", *Royal Aircraft Establishment*, Library Translation no. 1122, 1965.
RECHENBERG, I. "Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution", Frommann-Holzboog, 1973.
SCHWEFEL, H.-P. "Kybernetische Evolution als Strategie der experimentellen Forschung in der Strömungstechnik", Diplomarbeit, Hermann Föttinger Institut für Strömungstechnik, Technische Universität, Berlin, 1965.
SCHWEFEL, H.-P. "Evolutionsstrategie und numerische Optimierung", Dissertation, Technische Universität, Berlin, 1975.
SCHWEFEL, H.-P. "Evolution and Optimum Seeking", Sixth-Generation Computer Technology Series, Wiley, 1995.
SYSWERDA, G. "Uniform Crossover in Genetic Algorithms", *in* Schaffer, J.D. (ed.), *Proceedings of the Third International Conference on Genetic Algorithms*, Morgan Kaufmann Publishers, pp. 2-9, 1989.