CAP

MINISTÉRIO DA CIÊNCIA E TECNOLOGIA
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

UM MODELO DE COMPUTAÇÃO EVOLUTIVA PARA UMA ARQUITETURA CLIENTE-SERVIDOR

Rubens Cruz Gatto

Dissertação de Mestrado em Computação Aplicada,
orientada pelo Dr. Guilherme Bittencourt,
aprovada em Março de 1994.

INPE
São José dos Campos
Dezembro de 1992

Aprovada pela Banca Examinadora
em cumprimento a requisito exigido
para a obtenção do Título de Mestre
em Computação Aplicada

Dr. Luiz Antonio Nogueira Lorena
Presidente

Dr. Guilherme Bittencourt
Orientador

Dr. Antonio Miguel Vieira Monteiro
Membro da Banca

Dr. Paulo Ouvera Simoni
Membro da Banca

Dr. Antonio Carlos Roque da Silva Filho
Membro da Banca
- Convidado -

Candidato: Rubens Cruz Gatto

São José dos Campos, 28 de março de 1994

Resumo

Algoritmos Genéticos – um dos paradigmas de Computação Evolutiva – são algoritmos de busca e otimização cuja característica principal é, a partir de um conjunto inicial de pontos do espaço de busca de um problema a resolver, atingir outros pontos melhores e possivelmente ótimos como solução deste problema, através de um processo evolutivo análogo ao processo evolutivo natural dos seres vivos. Neste tipo de algoritmo identificamos certos aspectos que não necessitam ser alterados em função de sua aplicação a um ou a outro tipo de problema. Esses aspectos considerados básicos ou genéricos, podem assim ser encarados como serviços possíveis de serem prestados por uma máquina a outras de forma que várias aplicações desse tipo de algoritmo possam compartilhar desses recursos para desempenho de seus processos evolutivos, o que é uma característica da arquitetura Cliente-Servidor. O trabalho aqui apresentado trata do desenvolvimento e implementação, sobre uma arquitetura tipo Cliente-Servidor, de um modelo de Computação Evolutiva baseado em Algoritmos Genéticos. O presente trabalho mostra também como podemos explorar o modelo proposto para implementar o que se chama de Co-Evolução: algoritmos genéticos independentes que trocam informacoes entre si para o aprimoramento de um processo evolutivo global.

Para Hélia

Agradecimentos

O autor agradece ao INPE - Instituto Nacional de Pesquisas Espaciais - pela oportunidade de desenvolvimento da pesquisa que resultou na apresentação deste trabalho; a Guilherme Bittencourt por ter concordado em ser o orientador oficial deste trabalho; a José Antonio Gonçalves Pereira, pelo apoio sempre presente; a José Wagner Garcia, pela apresentação, a mim, deste assunto que julgo demasiado fascinante; a Paulo Roberto Martini, pelo incômodo "eventual" da utilização de sua sala; a Edenilse F. E. Orlandi, pelo apoio dentro do departamento; e principalmente a Pedro Paulo Balbi de Oliveira, que, ainda que impossibilitado formalmente, se empenhou pessoalmente na orientação efetiva das pesquisas que resultaram neste trabalho.

Conteúdo

Capítulo 1

Introdução

1.1 Visão Geral Sobre Computação Evolutiva

Computação evolutiva é um termo que foi gerado para designar sistemas computacionais inspirados nos princípios naturais da evolução orgânica. A evolução é a solução apresentada pelos seres vivos aos problemas de sobrevivência impostos pelos ambientes em que habitam. Muitos pesquisadores vêem o poder da evolução como algo que deva ser emulado. A seleção natural não enfrenta um dos grandes desafios que enfrentamos quando nos dedicamos a projetar um determinado sistema de software: especificar com antecedência todos os aspectos de um problema e as ações que o sistema deve tomar para lidar devidamente com eles [Holland 92].

A área da Computação Evolutiva (CE) está apresentando um crescimento considerável. Até recentemente poderíamos dizer que era formada por um conjunto amorfo de grupos de pesquisa sem interação uns com os outros. Ultimamente, pelo aumento do número de conferências, congressos e artigos na área, esse cenário vem se modificando rapidamente. Podemos distinguir três paradigmas em Computação Evolutiva: Estratégias de Evolução, Programação Evolutiva e Algoritmos Genéticos [De Jong and Spears 93] e ([Bukatova and Gulyaev 93]). Estratégias de Evolução foram desenvolvidas para a construção de sistemas capazes de resolver problemas difíceis de otimização de parâmetros. A representação é feita por um vetor de parâmetros manipulados por operadores de mutação desenvolvidos para causar perturbações de uma maneira aproveitável. Programação Evolutiva representa indivíduos como máquinas de estado finito capazes de responder a estímulos do ambiente, e os manipula por operadores (especialmente mutação) para provocar alterações na sua estrutura e comportamento no tempo [Fogel et al. 66]. Essas idéias foram aplicadas a vários problemas, como problemas de predição, de otimização e de aprendizado de máquinas.

Os Algoritmos Genéticos (AGs) levam esse nome devido à ênfase na representação e manipulação de indivíduos nos termos de sua representação codificada (ou genética) e não nos termos da sua expressão evidente [Holland 75]. São o tipo de algoritmo evolutivo mais difundido atualmente. Desenvolvidos a partir dos trabalhos de J. Holland, AGs tornaram possível explorar um domínio muito maior de possíveis soluções de um problema do que programas convencionais. Além disso as tentativas de implementação de princípios naturais em programas, sob condições bem conhecidas e controladas, podem permitir uma compreensão mais clara dos detalhes de como os seres vivos evoluiram na Natureza.

Existem atualmente inúmeras aplicações utilizando AGs para resolução de problemas, tanto computacionais, como de Engenharia, Física e até mesmo em Ciências Sociais [Goldberg 89]. Isso se deve, em termos, ao fato de podermos utilizar um AG para resolução de dois problemas completamente distintos um do outro alterando-se minimamente seu código.

1.2 Arquitetura Cliente-Servidor

A arquitetura Cliente-Servidor é uma filosofia de distribuição de recursos computacionais entre processos residentes em uma mesma máquina ou não. O servidor concentra em si o controle de recursos computacionais de um ambiente e fornece esses recursos a clientes mediante solicitação. Há na atualidade vários exemplos de sistemas que se valem de tal arquitetura, entre eles podemos citar aplicações de bancos de dados, onde uma máquina faz o gerenciamento do banco fornecendo os esquemas de aquisição e depósito de dados para outros sistemas, em geral residentes em máquinas diferentes. Isso possibilita que uma aplicação executada em uma máquina desfrute, via rede, das facilidades de acesso a dados controlados por uma outra máquina, mesmo que estas se encontrem muito distantes uma da outra.

1.3 Relação Entre Algoritmos Genéticos e Arquitetura Cliente-Servidor

A proposta deste trabalho é implementar AGs em uma arquitetura Cliente-Servidor de forma que certas tarefas genéricas de um AG possam ser fornecidas, por um processo servidor, como serviços prestados a aplicações-clientes.

A idéia de relacionar AGs e a arquitetura Cliente-Servidor surgiu a partir de algumas constatações:

- Independentemente do problema no qual seja aplicado um AG, certas tarefas por ele realizadas permanecem imutáveis. Essa pode ser uma das características que justifique o crescimento na utilização de AGs;
- Uma vez que essas tarefas não precisam ser alteradas em função do problema a ser resolvido, seria desejável que, se quiséssemos implementar um AG para resolver um determinado problema, não precisássemos nos preocupar com tais tarefas. Essas tarefas poderiam ser então abstraídas;
- Ultimamente vem crescendo o número de implementações paralelas de AGs, onde estes se comunicam entre si a fim de aprimorar o processo de resolução. Um meio que permitisse essa comunicação de forma facilitada, genérica e eficiente, utilizando inclusive redes, simplificaria significativamente esse tipo de implementação.

Tanto as tarefas fixas de um AG, quanto facilidades de comunicação entre AGs paralelos podem ser encarados como serviços que podem ser prestados por um sistema computacional a aplicações de AGs, as quais só precisariam estar voltadas aos aspectos particulares do problema a ser resolvido. Como filosofia de prestação de serviços, a arquitetura Cliente-Servidor é uma das formas ideais de implementação de tais conceitos.

1.4 Descrição deste Trabalho

Para que o leitor possa ter uma boa noção dos conceitos relacionados neste trabalho, dedicamos o Capítulo 2 aos conceitos biológicos que inspiraram sua teoria. Nele poderão ser encontradas, descritas de forma superficial, as informações básicas necessárias ao entendimento de um AG simples. Na bibliografia fornecida o leitor poderá buscar maiores detalhes sobre os conceitos apresentados.

O Capítulo 3 descreve os AGs, mostrando suas origens e revelando passo a passo as analogias entre o sistema natural e o computacional. Neste capítulo fundamentamos matematicamente a teoria computacional envolvida nos AGs e os comparamos a outros tipos de algoritmos de busca. No final deste capítulo listamos algumas dentre as muitas aplicações que se utilizam desta filosofia.

O Capítulo 4 apresenta o modelo proposto revelando as partes genéricas e específicas de um AG. Dois exemplos distintos são apresentados para ilustrar a independência de aplicação do módulo

implementado como servidor. São apresentadas também as formas de comunicação entre cliente e servidor, e uma seção é reservada à implementação da comunicação entre clientes via servidor.

O Capítulo 5 ilustra o projeto de uma aplicação co-evolutiva utilizando os recursos do modelo apresentado. Aplicações co-evolutivas são aplicações onde dois ou mais agentes iteragem, independentemente um do outro, induzindo a partir de processos evolutivos particulares um processo evolutivo global. Esta aplicação está voltada ao cálculo das raízes de uma função arbitrária, dentro de um intervalo especificado. Neste capítulo serão discutidas estratégias de codificação e de avaliação, bem como as parametrizações que permitem essa generalidade.

O Capítulo 6 conclui este trabalho fazendo uma projeção para o futuro do modelo e comparando-o com outros modelos paralelos conhecidos.

Capítulo 2

Conceitos Biológicos

Este capítulo tem como objetivo apresentar o conhecimento biológico envolvido na fundamentação dos Algoritmos Genéticos. A fim de familiarizar o leitor com essa teoria, bem como com o jargão utilizado, mostraremos nesta seção uma visão superficial dos conceitos biológicos que utilizaremos no embasamento deste trabalho. Não é nossa preocupação detalhar tais conhecimentos de forma rigorosa, pois nos basta apenas apresentar os fenômenos naturais envolvidos, de forma simples para uma boa compreensão da teoria de Algoritmos Genéticos. Maiores detalhes poderão ser encontrados na bibliografia apresentada no final deste trabalho.

2.1 Evolução e Seleção Natural

A Natureza apresenta uma variedade muito grande de ambientes e uma variedade ainda maior de seres vivos que os habitam. Existem diferenças ambientais drásticas entre muitos desses ambientes, tais como: florestas tropicais, desertos, polos, oceanos, etc... Nesses ambientes tão diferenciados podemos encontrar populações de espécies de seres vivos que habitam unicamente determinado ambiente, bem como populações de outras espécies que conseguem habitar ambientes que apresentem diferenças muito grandes.

O que faz com que uma população de uma determinada espécie de ser vivo consiga habitar um determinado ambiente é a adequação do conjunto de características que seus indivíduos possuem - que lhes permitem exercer suas funções vitais - às influências ambientais da região em que habita. A essa adequação damos o nome de ADAPTAÇÃO. Dizemos então que uma espécie está tão melhor adaptada a um ambiente quanto mais as suas características lhe permitirem exercer suas funções vitais. Por funções vitais entendemos as atividades de nascer, crescer, buscar e competir por alimento, se reproduzir, etc... Espécies mais adaptadas a um ambiente reproduzem-se mais facilmente e passam suas características para as próximas gerações.

Na Inglaterra de antes da Revolução Industrial havia uma espécie de maripôsa manchada que podia ser encontrada em duas variedades: uma clara e abundante e outra escura, mais rara (Figura 2.1). Isso se devia ao fato de que as maripôsas claras se confundiam com a coloração da casca das árvores onde pousavam e assim passavam desapercebidas dos seus predadores, os pássaros. A variedade escura, como era mais visível, era a mais comida pelos pássaros e conseqüentemente menos populosa. Com a Revolução Industrial, a fumaça das fábricas escureceu os troncos das árvores favorecendo a camuflagem das maripôsas escuras. Em pouco tempo a população de maripôsas escuras superou a de maripôsas claras e dominou o ambiente tornando-se mais abundante. A mudança ambiental provocada pelo início da Revolução Industrial levou as maripôsas claras, que antes eram as melhores adaptadas ao ambiente, a ficar em desvantagem, e favoreceu a variedade escura que pôde se esconder melhor de seus predadores. Esse fato ilustra claramente como uma característica de um ser vivo pode ser uma solução

satisfatória ou não a um problema de sobrevivência.

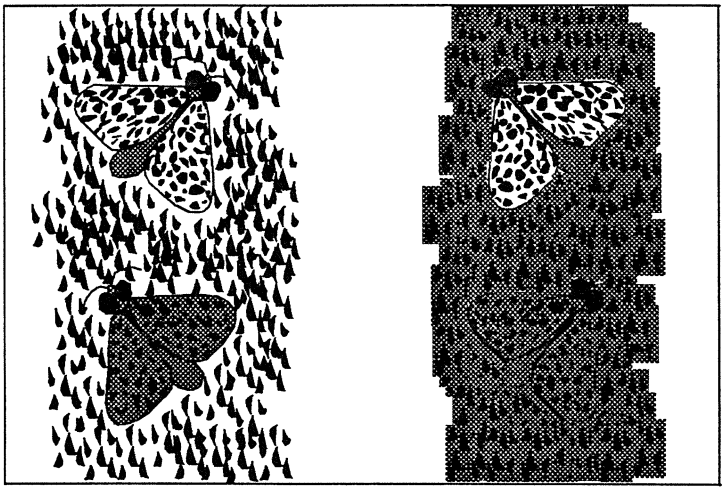


Figura 2.1: A variedade escura foi favorecida pela mudança no ambiente provocada pela Revolução Industrial na Inglaterra.

Um indivíduo de uma geração nunca é perfeitamente idêntico aos seus pais. Sempre há alguma pequena variação em suas características, que ainda assim não é suficiente para distingui-lo como sendo de uma espécie distinta da dos outros indivíduos de sua população. Uma alteração de características que um indivíduo apresente pode ser benéfica ou não ao exercício de suas funções vitais, ou ainda não interferir em nada. Se for benéfica (no caso anterior, cor escura em superfície escura), existirá uma probabilidade maior de que esse indivíduo reproduza-se e conseqüentemente passe suas características a seus descendentes. Caso contrário (idem, cor clara em superfície escura), essa probabilidade será pequena e a tendência será a de desaparecimento dessas características nas próximas gerações.

A esse mecanismo de aprimoramento damos o nome de SELEÇÃO NATURAL, e à sucessão de alterações das características dos seres vivos ao longo do tempo, de gerações em gerações, damos o nome de EVOLUÇÃO. A seleção natural, portanto, promove a evolução, ou seja, o aprimoramento da adaptação dos seres vivos ao ambiente que habitam.

2.2 A Célula

A unidade básica de quase toda forma de vida na Natureza é a célula, isto é, ser formado por uma ou mais células é uma característica comum a indivíduos de populações de uma grande maioria entre todas as espécies de seres vivos.

Todo indivíduo desenvolve-se a partir de uma única célula, chamada ovo. Essa célula duplica-se sucessivamente, gerando novas células do indivíduo, que por sua vez duplicam-se também, promovendo o seu crescimento. Durante o desenvolvimento do indivíduo, do ovo até sua forma final, vai acontecendo um processo de especialização celular: são formados o revestimento, os sistemas orgânicos, etc... É pelo processo de especialização celular que vão sendo estabelecidas as características finais de um indivíduo.

Toda célula - salvo algumas exceções - é constituída por três partes principais: Membrana celular, citoplasma e núcleo (Figura 2.2).

Se ampliarmos uma célula de forma a podermos analisar seu núcleo, veremos que o mesmo parece conter um emaranhado de fios muito longos. Esses fios na verdade são macromoléculas de ácido

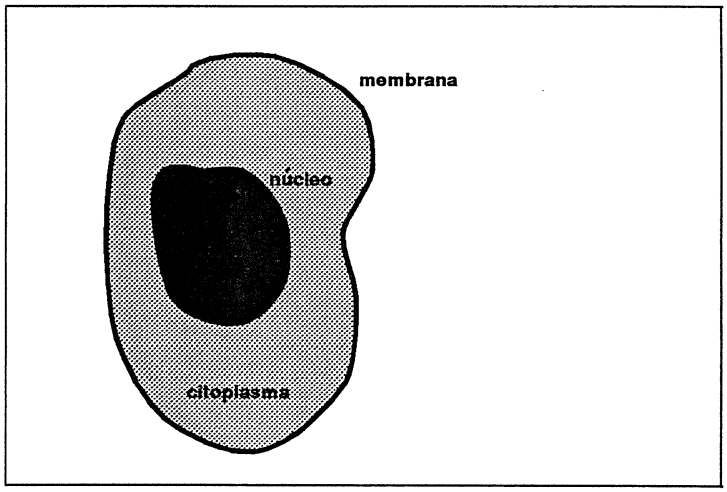


Figura 2.2: Estruturas básicas de uma célula comum.

desoxirribonucléico, ou simplesmente DNA. O DNA é formado pelo encadeamento linear de moléculas menores chamadas nucleotídeos, dos quais há apenas quatro tipos: Adenina, Timina, Citosina e Guanina (A,T,C e G).

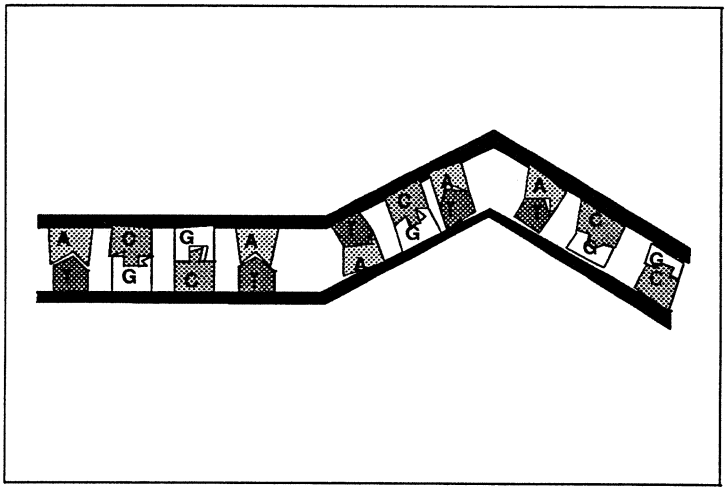


Figura 2.3: A estrutura em forma de escada de uma molécula de DNA.

Apesar do arranjo ser linear, as unidades do DNA são um pouco mais complexas: dispõem-se aos pares. São os seguintes os pares usados como unidades da seqüência formadora do DNA: CG e AT. A melhor maneira de visualizarmos o DNA é como sendo uma escada onde cada degrau seria formado por um desses pares, fazendo diferença qual nucleotídeo está do lado esquerdo e qual está do lado direito do par (Figura 2.3). Desse modo, temos quatro tipos diferentes de degraus: CG, GC, AT e TA. No núcleo das células o DNA pode atingir comprimentos excepcionais, contendo várias centenas de milhões de pares de nucleotídeos [Weisskopf 63].

A molécula de DNA atua como um "guia" para a formação de proteínas, através do código for-

mado pela sucessão de nucleotídeos. Assim, uma determinada porção de DNA, pela forma como seus nucleotídeos se sucedem, determina o aparecimento de uma proteína, enquanto que outra porção determina outra proteína e assim por diante. São justamente as proteínas as substâncias responsáveis pelo processo de especialização celular que determina as características de um indivíduo.

Resumindo, podemos dizer que as características de um indivíduo são determinadas por porções do DNA contido no núcleo de suas células, e que são definidas pelo código representado pela sucessão de pares de nucleotídeos dentro dessas porções. A uma porção do DNA que determina uma característica isolada damos o nome de GENE. À configuração genética que essa porção assume damos o nome de GENÓTIPO e à expressão dessa característica damos o nome de FENÓTIPO.

2.3 Reprodução

É através do processo reprodutivo que as características de uma espécie de ser vivo são passadas para as próximas gerações. Entretanto, antes de falarmos desse processo, detalharemos o processo de divisão celular. São conhecidos dois processos de divisão celular: Mitose e Meiose.

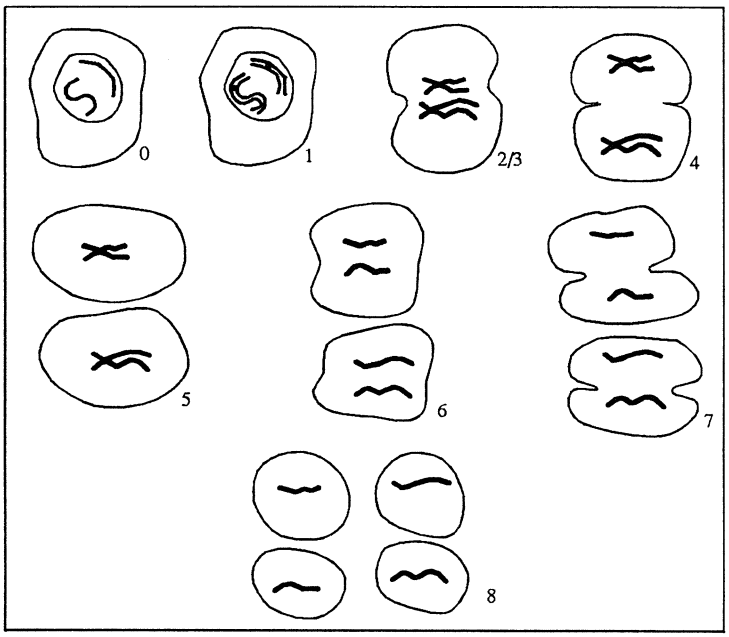


Figura 2.4: Estágios da divisão celular por meiose.

O primeiro é um processo de reprodução celular com fins de constituição do indivíduo, enquanto que o segundo (Figura 2.4) é o processo responsável pela formação das células reprodutivas, também chamadas GAMETAS. Descrevemos abaixo, simplificadamente, como ocorre esse processo:

- estágio 1: as moléculas de DNA do núcleo da célula se auto duplicam;
- estágio 2: as moléculas de DNA condensam-se sob a forma de bastonetes duplos chamados cromossomos. O núcleo desaparece;
- estágio 3: os cromossomos distribuem-se aos pares ao longo de um plano equatorial da célula;
- estágio 4: cada cromossomo de cada par migra para cada um dos polos da célula;

- estágio 5: a célula se divide formando duas novas células;
- estágio 6: em cada uma das novas células geradas os cromossomos rompem sua estrutura dupla;
- estágio 7: cada metade da estrutura rompida migra para cada polo de cada célula;
- estágio 8: cada célula divide-se novamente formando dois gametas;

Podemos observar que, de uma célula normal foram gerados pelo processo de meiose, quatro gametas, apresentando cada um metade da informação genética da célula original. Em relação ao conteúdo genético, dizemos que os gametas são haplóides e que uma célula normal é diplóide.

Durante o processo reprodutivo, o gameta de um indivíduo encontra-se com um gameta de outro indivíduo formando uma célula novamente diplóide cuja informação genética é uma combinação das informações genéticas dos gametas que a formaram. Essa célula é o ovo do qual se desenvolverá um novo indivíduo que apresentará características de cada indivíduo que forneceu os gametas para sua formação.

Como vimos na Figura 2.4, os cromossomos se reúnem aos pares, no estado diplóide. No fim da meiose são gerados gametas haplóides, que ao se juntarem aos gametas de outros indivíduos formam ovos diplóides. As características de um ser vivo na realidade são definidas por pares de genes, dos cromossomos no estado diplóide, e não por um apenas. Um par de genes pode ser homogêneo ou não. Quando um par é heterogêneo, apenas um de seus genes define a característica. Esse gene é dito DOMINANTE e o outro RECESSIVO. Um gene recessivo só define uma característica quando o par a que pertence for homogêneo em relação a esse gene. Por exemplo: seja a característica "cor de olhos". O gene que define a cor castanha (A) é dominante em relação ao gene que define a cor azul (a). Se uma pessoa possuir o par (Aa) ou o par (AA), seus olhos terão a cor castanha. Apenas se possuir o par (aa) a pessoa terá olhos azuis. Aparentemente existe um desperdício genético nessa forma de representação de características, uma vez que apenas a metade dos genes as codificam. Porém os outros genes não expressos representam características já expressas em gerações passadas, e que se encontram em estado de latência podendo ser novamente exploradas e aproveitadas, principalmente no decorrer de mudanças ambientais que modifiquem as condições de adaptação, como no caso das maripôsas na Inglaterra.

Durante o processo da meiose alguns acidentes podem ocorrer. Dois tipos de acidentes são extremamente importantes para a formação das características no novo indivíduo. No terceiro estágio da meiose, quando os cromossomos estão pareados, frequentemente há um entrelaçamento entre as suas partes. Esse entrelaçamento pode provocar um rompimento dos cromossomos e a troca de partes entre os cromossomos do par, causando uma recombinação dos códigos genéticos que pode levar ao aparecimento de novas características no indivíduo. Esse fenômeno é denominado CROSSOVER (Figura 2.5).

Devido à influência de fatores externos, podem haver alterações isoladas na configuração genética podendo também provocar o surgimento de um novo caráter no fenótipo. Esse fenômeno é chamado MUTAÇÃO.

Estes são os conceitos biológicos necessários e suficientes para a fundamentação da teoria que será apresentada nos capítulos a seguir. Outros conceitos, tais como as Leis de Mendel, dominância e recessividade também tem sua aplicabilidade em Algoritmos Genéticos. Porém essa aplicabilidade é restrita e não faz parte do foco do trabalho.O leitor poderá encontrar maiores detalhes sobre esse assunto em [Grobman 62], [Jarman 70], [Storer and Usinger 78] e [Weisskopf 63].

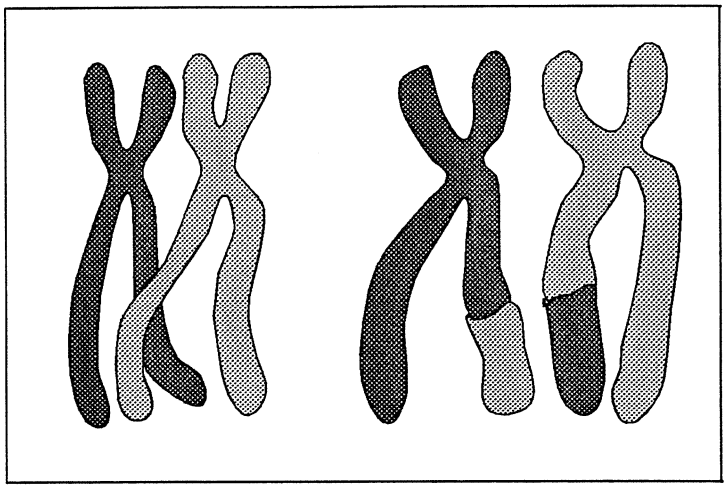


Figura 2.5: O crossover entre um par de cromossomos.

Capítulo 3

Algoritmos Genéticos

3.1 Introdução

A proposta de um algoritmo de busca ou otimização é encontrar uma ou mais soluções ótimas, ou plenamente satisfatórias, para um determinado problema. Isso se faz através da análise, comparação e refinamento sucessivos de pontos dentro de um conjunto de opções mais viáveis ou menos viáveis. Poderíamos fazer a seguinte analogia (Figura 3.1):

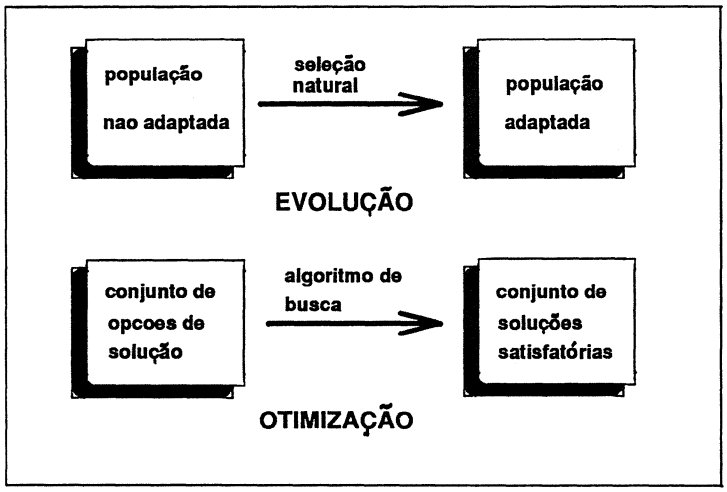


Figura 3.1: Comparação entre o processo evolutivo e o processo de otimização.

- Otimização: busca de soluções ótimas ou plenamente satisfatórias para um problema a ser resolvido, a partir de um espaço de busca finito ou infinito.
- Evolução: aprimoramento das características de indivíduos de espécies de seres vivos que resulta em uma adaptação ao ambiente em que habitam.

Dessa forma poderíamos encarar o processo de refinamento de soluções para um problema a ser resolvido como sendo a evolução de indivíduos de uma população para o aprimoramento da adaptação ao ambiente em que habitam.

ALGORITMOS GENÉTICOS atuam exatamente conforme esse conceito: a partir de uma população inicial (conjunto inicial de opções de solução ao problema), por um mecanismo análogo ao

de seleção natural, um AG promove a evolução dessa população gerando novas populações contendo indivíduos cada vez melhor adaptados ao ambiente, ou seja, gerando opções de solução cada vez mais satisfatórias à resolução do problema (Figura 3.2).

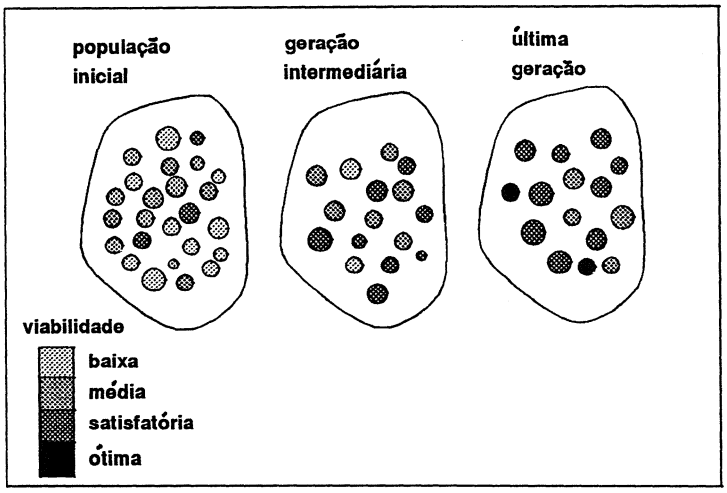


Figura 3.2: Evolução dos graus de adaptação ou viabilidades.

Os primeiros algoritmos desse tipo foram desenvolvidos por John Holland, sua equipe e alunos na Universidade de Michigan. Os objetivos iniciais do grupo eram abstrair e explicar formalmente os processos adaptativos naturais, e projetar sistemas de software que tirassem proveito dos importantes mecanismos dos sistemas naturais. A primeira monografia sobre o assunto, publicada por Holland em 1975, foi "Adaptation in Natural and Artificial Systems". Muitas dissertações a partir desse ponto estabeleceram a importância desta técnica nos campos de otimização de funções e de aplicações em controle. Desde que estabelecidos como uma ferramenta significativa para aplicação em problemas nos quais é necessária uma busca eficiente, AGs estão sendo cada vez mais empregados nas áreas mais diversas como Administração, Ciências e Engenharia. A razão disto é simples: AGs são algoritmos computacionalmente simples embora poderosos no aprimoramento da busca de soluções, e ainda, não são afetados por restrições em relação ao espaço de busca (continuidade, existência de derivadas, unimodalidade, etc...) [Goldberg 89].

Vamos definir aqui um conceito que será abordado freqüentemente até o final do trabalho. Neste trabalho, o conceito de VIABILIDADE terá uma conotação diferente da geralmente usada pela Pesquisa Operacional, onde um ponto do espaço de busca é viável apenas quando é solução do problema. Aqui, a viabilidade de um ponto do espaço de busca de um problema significará o quanto este ponto pode estar próximo da melhor solução possível deste problema. Para diferenciarmos um ponto de outro, atribuimos a eles graus de viabilidade. Dessa forma, um ponto x com um grau de viabilidade maior que o de outro ponto y, estará mais próximo de uma solução do problema do que y.

3.2 Analogias Entre o Modelo Natural e o Artificial

Os aspectos relacionados entre um AG e o modelo natural são então os seguintes:

- Um indivíduo é um ponto no espaço de busca do problema a ser resolvido, isto é, uma opção de solução;
- Uma população é um subconjunto do espaço de busca;

- O cromossomo é a estrutura que codifica as características do indivíduo. Assim um AG não trabalha com os pontos do espaço de busca diretamente, mas sim com uma representação codificada [Goldberg 89];
- O ambiente representa o problema a ser resolvido;
- A adaptação do indivíduo é o grau de viabilidade da opção de solução que ele representa ao problema a ser resolvido, isto é, o quanto esta opção é satisfatória;
- A evolução surge da reprodução sucessiva de populações, cada uma a partir da anterior;
- São formados pares de indivíduos de uma população, que se reproduzirão e gerarão uma nova população de igual tamanho, que poderá ou não sobrepor-se à anterior; portanto um AG não trata apenas um ponto do espaço de busca por vez, mas vários simultaneamente;
- Os pares são escolhidos conforme mecanismo análogo à seleção natural;
- A partir dos dois indivíduos do par, designados "pais", são gerados dois novos indivíduos, designados "filhos", pela atuação de operadores genéticos inspirados nos fenômenos naturais de crossover e mutação;

No modelo natural, a seqüência de nucleotídeos no DNA fornece o código para o desenvolvimento das características do indivíduo, entretanto é sob a forma de cromossomo que o DNA sofre os fenômenos de crossover e mutação. Portanto usamos o cromossomo como estrutura modelar e não o DNA.

3.3 Implementação de um Algoritmo Genético

Suponha o seguinte problema: Dispomos de um dispositivo "caixa-preta" munido de oito chaves do tipo "liga-desliga" e uma fenda (Figura 3.3). De acordo com a configuração aplicada às chaves, o dispositivo fornece pela fenda uma determinada quantia de dinheiro, obedecendo uma função desconhecida por nós. Queremos então descobrir qual a configuração que deveríamos aplicar às chaves para que o dispositivo nos fornecesse a maior quantia de dinheiro possível, ou seja, queremos achar o máximo da função de transição configuração/prêmio.

Inicialmente definimos cada configuração das chaves como uma opção de solução, e o espaço de busca como o conjunto de todas configurações possíveis. Se definirmos para cada chave um status '0' se desligada e '1' se ligada, cada ponto do espaço de busca seria definido por uma seqüência de oito símbolos. Por exemplo, a seqüência 01000111 define a configuração na qual só a segunda e as três últimas chaves estão ligadas. Cada configuração será portanto um indivíduo e qualquer subconjunto do espaço de busca, uma população.

O cromossomo no modelo natural é formado pela seqüência de pares de nucleotídeos, sendo esses pares apenas quatro (AT,TA,CG e GC). O cromossomo então nada mais é que uma "palavra" definida sobre um alfabeto de quatro símbolos. No nosso caso, nossos indivíduos são representados pela seqüência de oito símbolos de um alfabeto $A = \{0, 1\}$. Temos aqui uma relação direta entre a seqüência que define a configuração das chaves e o cromossomo que define o indivíduo (Figura 3.3).

Como vimos acima, codificamos as opções de solução sob a forma de indivíduos definidos por cromossomos de oito posições (ou genes) sobre um alfabeto binário. Nem sempre a codificação pode ser tão direta assim. Como veremos mais adiante existem vários modos de se fazer a codificação.

O problema a ser resolvido deve ser implementado de forma que receba um indivíduo, representado por seu cromossomo, decodifique-o para um ponto de busca, analise esse ponto e retorne um valor maior ou igual a zero que indique quão satisfatório ele é como opção de solução ao problema. Esse valor é o grau de adaptação (viabilidade) ao ambiente (problema) (Figura 3.4). Para o caso do nosso

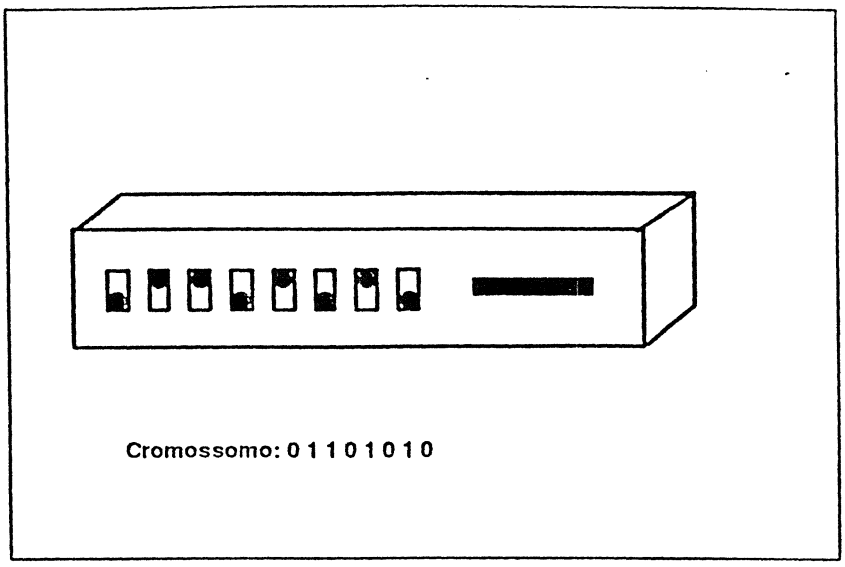


Figura 3.3: Dispositivo tipo "caixa-preta". Conforme a configuração aplicada às chaves um prêmio em dinheiro sai pela fenda.

prêmio fornecido pelo dispositivo ao grau de viabilidade da configuração das chaves, isto é, a adaptação do indivíduo ao ambiente.

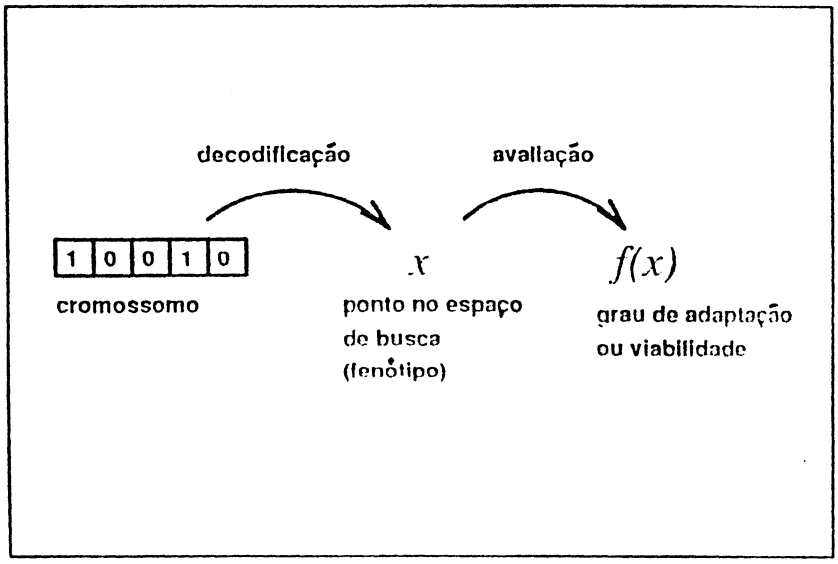


Figura 3.4: Passos para a avaliação da adaptação de um indivíduo.

Dada uma população, para que possamos gerar a próxima população, precisamos formar os pares para reprodução. Como vimos no capítulo 2, o mecanismo de seleção natural faz com que a probabilidade de um indivíduo se reproduzir no ambiente que habita seja tão maior quanto melhor estiver adaptado a esse ambiente.

1. Para cada indivíduo x_i, calculamos o seu grau de adaptação $f(x_i)$;
2. Calculamos a soma total do grau de adaptação de todos indivíduos da população por,

$$S = \sum_{i=1}^{t} f(x_i)$$

onde t é o tamanho da população;

3. Para cada indivíduo calculamos sua probabilidade de seleção como sendo

$$P(x_i) = \frac{f(x_i)}{\sum_{j=1}^{t} f(x_j)}$$

4. Finalmente selecionamos cada indivíduo de cada par segundo sua probabilidade de seleção.

Conforme o esquema acima, fica possibilitada a ocorrência do mesmo indivíduo em mais de um par.

Uma vez escolhido um par, geramos dois cromossomos "filhos" a partir da aplicação dos operadores genéticos crossover e mutação sobre os cromossomos dos pais.

A operação de crossover ocorre da seguinte forma: de acordo com uma probabilidade definida, é sorteada aleatoriamente uma posição para ruptura entre 0 e o tamanho dos cromossomos; os dois cromossomos são quebrados em duas partes na posição sorteada; são gerados os dois novos cromossomos filhos pela cópia cruzada das partes dos dois cromossomos dos pais. Convém notar que, embora na Natureza o crossover ocorra durante a meiose, no caso de um AG tudo se passa como se ocorresse no encontro dos gametas. O CROSSOVER é um operador extremamente importante no aproveitamento de características de indivíduos já analisados para a geração de novos indivíduos com novas características surgidas da recombinação dos cromossomos de seus pais. Por exemplo: do problema do dispositivo; se tivéssemos escolhido dois cromossomos A e B para reprodução, $A = 01100100$ e $B = 10101111$, e a posição sorteada fosse $p = 5$, geraríamos novos cromossomos conforme a Figura 3.5.

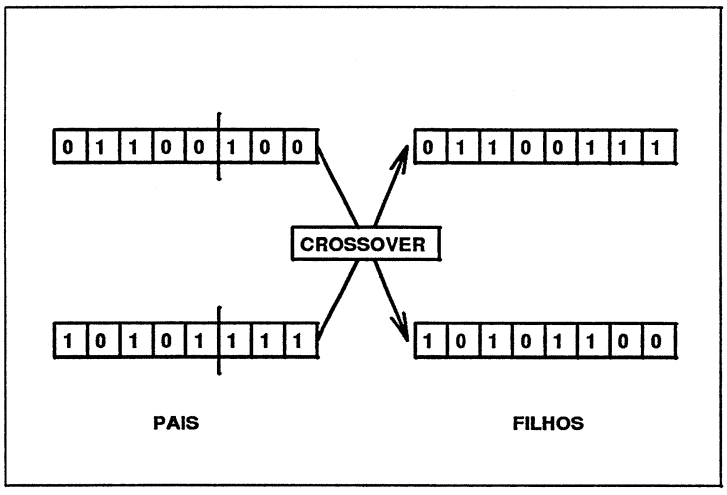


Figura 3.5: O crossover simples ou de 1 ponto.

Sozinho entretanto, o crossover não é suficiente para gerar qualquer outra configuração genética. Se nenhum dos pais possuir os símbolos, por exemplo, "01", em suas posições quatro e cinco, nenhum

dos dois filhos gerados possuirão essas posições preenchidas com os respectivos símbolos. O operador de mutação resolve esse problema. Durante o processo de cópia cruzada dos cromossomos, também de acordo com uma probabilidade pré-definida, podemos manter o símbolo correntemente copiado ou trocá-lo por outro símbolo qualquer do alfabeto. Essa troca deve ocorrer com uma probabilidade muito pequena, pois se ocorresse frequentemente destruiria todas as configurações selecionadas e evoluídas até sua ocorrência [Goldberg 89]. Ocorrendo com uma probabilidade pequena, pode ser gerada uma configuração melhor que poderá ser propagada para outras gerações (Figura 3.6).

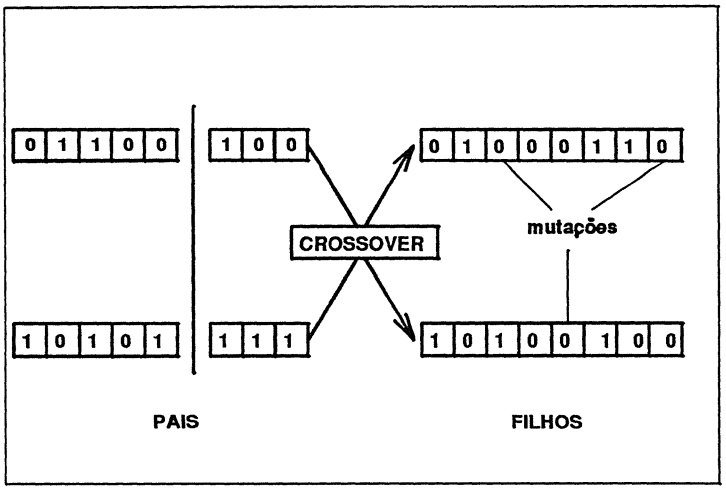


Figura 3.6: A mutação.

A condição de parada de um AG fica a critério da aplicação. Pode ser tanto a ocorrência de um indivíduo que represente uma solução ótima ou satisfatória, como um número finito de gerações previamente estabelecido.

3.4 Exemplo de aplicação

Vamos aqui exemplificar a implementação de um AG para o problema do dispositivo apresentado anteriormente (Figura 3.3). Vamos supor que a função de cálculo do prêmio seja

$$f(x) = x^2$$

onde x é um número inteiro traduzido do cromossomo, considerado como representação binária. Como o dispositivo possui oito chaves,

$$0 \leq x \leq 2^8 - 1$$

Assim, se o conjunto de chaves tiver a configuração 00100100, o valor de x será 36 e $f(x) = x^2 = 1296$.

Trabalharemos com uma população de 20 indivíduos. Este número é arbitrário. Quanto maior a população maior é a diversidade de indivíduos, e conseqüentemente menos gerações serão necessárias para atingirmos soluções satisfatórias. A população inicial será gerada ao acaso e as probabilidades de ocorrência de crossover e mutação são de 1.0 e 0.01 respectivamente. No apêndice A temos a listagem das três primeiras gerações dessa aplicação. Cada linha representa um indivíduo da população e cada coluna é definida como:

- Coluna 1: Mostra o número do indivíduo e seu cromossomo.

- Coluna 2: Mostra o valor numérico representado pelo cromossomo.
- Coluna 3: Mostra o grau de adaptação do indivíduo, que é o valor pago pelo dispositivo quando submetido à configuração codificada pelo cromossomo.
- Coluna 4: Mostra a probabilidade de seleção do indivíduo.
- Coluna 5: Mostra o número esperado de vezes que o indivíduo deve ser selecionado para reprodução, definido por

$$Numesp = p(x_i)t$$

onde t é o tamanho da população.

- Coluna 6: Mostra os números dos pais do indivíduo na população anterior.
- Coluna 7: Mostra a posição nos cromossomos dos pais em que houve o crossover que gerou o indivíduo.

Ao final de cada população estão apresentadas as estatísticas respectivas.

Como podemos observar, a população inicial, da geração 000, já possui indivíduos satisfatórios: pela ordem os indivíduos 05, 20, 09, e 10. As probabilidades de seleção estão na casa dos 15%, enquanto que as dos outros não chegam a 10%.

Na geração 001, esses indivíduos participaram de todos os pares para reprodução. Foi gerado um indivíduo ainda mais satisfatório que os da geração 000. Este indivíduo, de número 07, filho dos indivíduos 10 e 05 está mais próximo de um valor ótimo que seria o máximo da função. Notamos ainda que os graus de adaptação mínimo e médio cresceram significativamente, e que restou apenas um indivíduo com um grau de adaptação relativamente baixo.

Na geração seguinte, 002, há a ocorrência de um indivíduo considerado ótimo, ou seja o máximo da função, que já conhecemos de antemão. Verificamos também que o grau de adaptação médio cresceu mais ainda, atingindo aproximadamente 83% do valor máximo.

Para ilustrar como ocorreu a evolução até atingir este indivíduo mostramos na Figura 3.7 a sua "árvore genealógica".

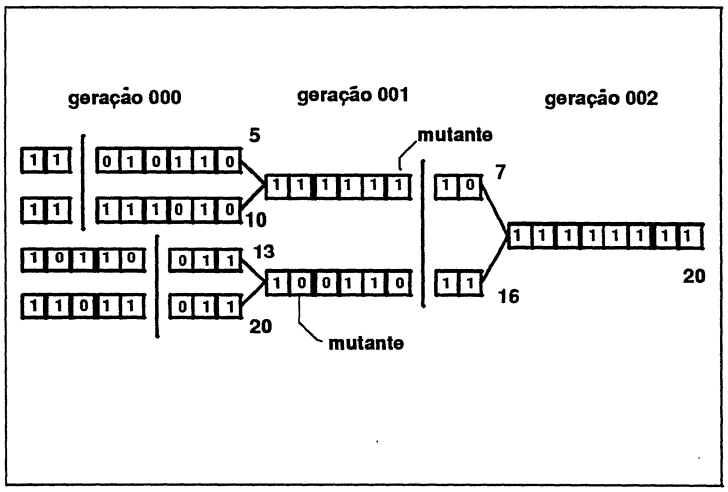


Figura 3.7: Árvore genealógica do indivíduo 20 na geração 002.

3.5 Aspectos Matemáticos dos Algoritmos Genéticos

Para justificar matematicamente a eficiência de um AG, vamos inicialmente estabelecer o conceito de ESQUEMA. Um esquema é a formalização do padrão de similaridade entre cromossomos de um subconjunto de indivíduos de uma população [Goldberg 89]. Para definir um esquema usaremos uma cadeia de símbolos do mesmo comprimento dos cromossomos sobre o mesmo alfabeto acrescido do símbolo de indeterminação "*", que pode representar qualquer outro símbolo do alfabeto. Dessa forma, para o exemplo do dispositivo, "*" poderia assumir tanto "0" como "1". Dizemos que um determinado esquema H mapeia os cromossomos, por exemplo, C_1, C_2 e C_3, se para toda posição de H ocupada por um determinado símbolo que não seja "*", a mesma posição em cada cromossomo for ocupada pelo mesmo símbolo. Para as outras posições de H, ocupadas por "*", as mesmas posições dos cromossomos podem conter qualquer símbolo do alfabeto. Exemplo:

Sejam os cromossomos:

$$C_1 = 11010, C_2 = 10001, C_3 = 00111$$

e os esquemas:

$$H_1 = 1 * 0 * *, H_2 = *0 * *1, H_3 = * * 11*, H_4 = *11 * *$$

Pela definição de esquema, verificamos que H_1 mapeia C_1 e C_2, H_2 mapeia C_2 e C_3, H_3 mapeia apenas C_3 e H_4 não mapeia nenhum dos cromossomos.

O esquema é um meio compacto e poderoso para explorar as similaridades entre cromossomos [Goldberg 89]. O número de esquemas possíveis para cromossomos de tamanho L definidos sobre um alfabeto de cardinalidade K é dado por $(K+1)^L$, devido à inclusão do símbolo "*". O número de esquemas que mapeiam um determinado cromossomo é 2^L uma vez que cada posição, ou é fixada ou é ocupada por "*". Para não tornar os cálculos muito complexos, usaremos um alfabeto binário $V = \{0, 1\}$.

Dizemos que um esquema H é viável se todos os cromossomos mapeados por H representam indivíduos viáveis. Não ocorrendo crossover, o número de cromossomos mapeados por um esquema viável H aumentará a cada geração segundo sua probabilidade de seleção que é calculada pela soma das probabilidades de seleção dos indivíduos cujos cromossomos são mapeados por H. Podemos definir dois atributos para um esquema: especificidade e abrangência. Um esquema H será tão específico quantas forem suas posições fixas. Denominamos "ordem" de um esquema H, e denotamos por $O(H)$, o número de posições fixas do esquema H. Ex: o esquema $H_1 = 011 * 1 * **$ tem ordem $O(H_1) = 4$ e o esquema $H_2 = * * *1 * * * *$ tem ordem $O(H_2) = 1$. Portanto o esquema H_1 é mais específico do que H_2. Desse modo podemos concluir que quanto mais específico for um esquema menos cromossomos são mapeados por ele, pois o número de cromossomos mapeados por um determinado esquema H deve ser calculado por $K^{L-O(H)}$. Um esquema H é tão mais abrangente quanto maior for a distância entre a primeira e a última posição fixa, percorrendo-se o esquema da esquerda para a direita. A essa distância damos o nome de comprimento de H, denotado por $d(H)$. Ex: se $H_1 = 011 * 1 * *$, a primeira posição fixa é a posição 1 ocupada por "0" e a última posição fixa é a posição 5, ocupada por "1". Assim $d(H_1) = 5 - 1 = 4$.

Sejam $m(H,t)$ o número esperado de cromossomos mapeados pelo esquema H na geração t e $r(H,t)$ o número de cromossomos efetivamente mapeados pelo esquema H na geração t. Seja $f(H)$ a média do grau de adaptação dos indivíduos cujos cromossomos são mapeados pelo esquema H e n o tamanho da população. O número esperado de cromossomos mapeados pelo esquema H na geração $t+1$ é dado por:

$$m(H, t+1) = r(H,t).n.\frac{f(H)}{\sum_{j=1}^{n} f(x_j)} .$$

Embora [Goldberg 89] não faça a distinção entre o número esperado e o numero efetivo de esquemas mapeados, introduzimos este conceito para uma melhor justificação matemática da propagação dos esquemas.

Se definirmos f* como a média total do grau de adaptação dos indivíduos da população:

$$f^* = \frac{\sum_{j=1}^{n} f(x_j)}{n},$$

$$m(H, t+1) = r(H, t).\frac{f(H)}{f^*} .$$

Em palavras, o número esperado de cromossomos mapeados na geração $t+1$ por um determinado esquema H aumenta ou diminui conforme a razão entre a média do grau de adaptação dos cromossomos mapeados por H e a média do grau de adaptação de todos indivíduos da população. Supondo que a diferença entre $f(H)$ e f^* seja sempre $c.f^*$, c constante, de geração a geração, e que $r(H,t) = m(H,t)$ sempre, temos:

$$m(H, t+1) = m(H, t).\frac{(f^* + c.f^*)}{f^*} ,$$

$$m(H, t+1) = m(H, t).(1+c) ,$$

$$m(H, t) = m(H, 0).(1+c)^t .$$

Como podemos observar, nessas condições, a propagação de um esquema aumenta ou diminui segundo uma projeção geométrica de razão $1+c$.

A propagação dos esquemas sofre interferência dos operadores de crossover e de mutação. O crossover pode ou não destruir um esquema dependendo do seu comprimento. Destruição de um esquema é a separação de suas posições fixas extremas pela ocorrência de um corte entre elas. Por exemplo: o esquema $1***0$ tem maior probabilidade de ser destruído pelo crossover do que o esquema $*11**$. Isso porque para o primeiro existem quatro posições possíveis para corte, que separariam as suas posições fixas extremas, enquanto que para o último existe apenas uma. Assim, esquemas que mapeiam cromossomos de indivíduos com maior grau de adaptação e que tenham menor comprimento têm maior chance de serem propagados geração a geração.

Seja o cromossomo $C = 0111000$ de tamanho $L = 7$ e os dois esquemas que o mapeiam, $H_1 = *1****0$ e $H_2 = ***10**$. Seja $r = 3$ o local selecionado para o crossover (Figura 3.8).

A menos que o indivíduo selecionado para reprodução com o indivíduo portador de C possua um cromossomo com os mesmos símbolos nas posições definidas por H_1, o esquema H_1 será destruído, pois suas posições fixas irão cada uma para cada novo cromossomo gerado. O esquema H_2 será propagado pois suas posições irão juntas para o mesmo cromossomo. Notamos que $d(H_1) = 7-2 = 5$, $d(H_2) = 5-4 = 1$, e que r é selecionado entre $L-1 = 7-1 = 6$ possíveis locais. A probabilidade de H_1 ser destruído é $P_d = d(H_1)/(L-1) = 5/6$ e a probabilidade de sobrevivência $P_s = 1-5/6 = 1/6$. A probabilidade de H_2 ser destruído é $P_d = d(H_2)/(L-1) = 1/6$ e a probabilidade de sobreviver é $P_s = 1-1/6 = 5/6$. Devemos levar em conta também a probabilidade da ocorrência do crossover P_c. Portanto, levando em consideração apenas o crossover, o número esperado de cromossomos mapeados na geração $k+1$ passa a ser:

$$m(H, t+1) = r(H, t).\frac{f(H)}{f^*}.(1 - \frac{P_c.d(H)}{L-1})$$

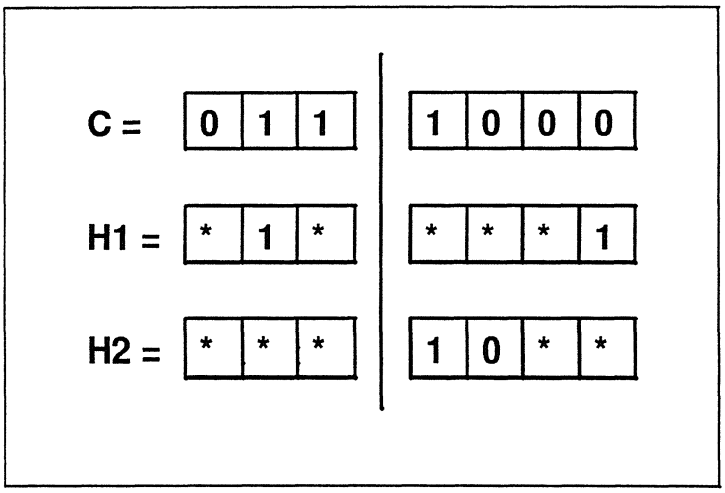


Figura 3.8: Efeito do crossover. O esquema H1 será destruido enquanto que o esquema H2 será propagado.

Seja P_m a probabilidade de ocorrer uma mutação. Para que um esquema H sobreviva à mutação, todas as suas posições fixas devem ser mantidas. A probabilidade de uma posição ser mantida é $1-P_m$, e um esquema de ordem $O(H)$ sobrevive à mutação quando todas as suas posições fixas permanecem inalteradas. Portanto $P_s = (1 - P_m)^{O(H)}$. Podemos escrever agora a equação final da propagação esperada para um esquema H:

$$m(H, t+1) = r(H, t).\frac{f(H)}{f^*}.(1 - \frac{P_c.d(H)}{L-1}).(1 - p_m)^{O(H)}$$

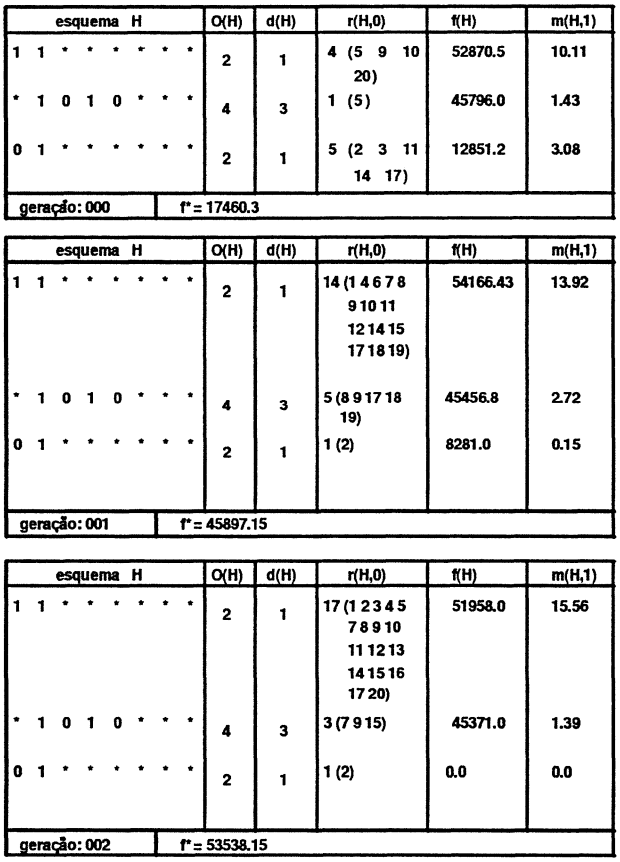
Vejamos um exemplo de propagação de esquemas: Usando novamente o exemplo do dispositivo "caixa-preta" e a listagem fornecida no apêndice A, obtemos a propagação de alguns esquemas escolhidos, mostrada na Figura 3.9.

3.6 Variações

3.6.1 Escalonamento do Grau de Adaptação

Em muitas aplicações de AGs, especialmente naquelas onde a população utilizada tem um tamanho reduzido, é comum encontrarmos logo nas primeiras gerações indivíduos minimamente satisfatórios no meio de outros medíocres. Se deixarmos a evolução correr ao sabor da regra normal de seleção apresentada, esses indivíduos satisfatórios dominam rapidamente uma grande parte da população, o que não é desejável pois causa uma convergência prematura para um indivíduo, que domina a população e impede a exploração de outras configurações possivelmente viáveis [Goldberg 89].

Outro problema que surge é, depois de algumas gerações, a média dos graus de adaptação ficar muito próxima do grau máximo de adaptação da população, apesar de poder haver uma diversidade significativa na população. Permitindo isso, as probabilidades de seleção dos indivíduos ficam distribuídas em um intervalo muito estreito reduzindo o processo seletivo a um sorteio sem favoritos, isto é, os melhores graus de adaptação, por serem muito próximos dos outros graus, pouco contribuem para o aproveitamento dos cromossomos para as próximas gerações [Goldberg 89].



esquema H	O(H)	d(H)	r(H,0)	f(H)	m(H,1)
1 1 * * * * * *	2	1	4 (5 9 10 20)	52870.5	10.11
* 1 0 1 0 * * *	4	3	1 (5)	45796.0	1.43
0 1 * * * * * *	2	1	5 (2 3 11 14 17)	12851.2	3.08
geração: 000	f* = 17460.3				

esquema H	O(H)	d(H)	r(H,0)	f(H)	m(H,1)
1 1 * * * * * *	2	1	14 (1 4 6 7 8 9 10 11 12 14 15 17 18 19)	54166.43	13.92
* 1 0 1 0 * * *	4	3	5 (8 9 17 18 19)	45456.8	2.72
0 1 * * * * * *	2	1	1 (2)	8281.0	0.15
geração: 001	f* = 45897.15				

esquema H	O(H)	d(H)	r(H,0)	f(H)	m(H,1)
1 1 * * * * * *	2	1	17 (1 2 3 4 5 7 8 9 10 11 12 13 14 15 16 17 20)	51958.0	15.56
* 1 0 1 0 * * *	4	3	3 (7 9 15)	45371.0	1.39
0 1 * * * * * *	2	1	1 (2)	0.0	0.0
geração: 002	f* = 53538.15				

Figura 3.9: A propagação de esquemas através de gerações.

Um artifício usado para evitar esses problemas é o escalonamento dos graus de adaptação da população. Uma vez calculados todos os graus de adaptação da população e feitas as estatísticas, fazemos um escalonamento desses valores, por exemplo, de forma linear (Figura 3.10).

Os valores de $f(x)$ é são os graus de adaptação dos indivíduos x e os valores de $g(x)$, seus novos graus de adaptação escalonados. Um escalonamento linear comumente usado consiste em calcular $g(x)$ de forma que a média não seja alterada e o novo máximo seja o dobro dessa média. Contudo devemos ter o cuidado de garantir que esse escalonamento não gere graus de adaptação negativos.

3.6.2 Tipos de Crossover

A operação de crossover gera novas configurações de cromossomos através da combinação de dois cromossomos originais. Existem várias formas de implementar essa operação. A que descrevemos anteriormente é a forma mais simples, chamada "crossover de um ponto".

O "crossover de dois pontos" é semelhante ao de um ponto, mas ao invés tratar os cromossomos como cadeias de símbolos, trata-os como anéis que devem ser cortados em dois pontos para separação dos segmentos que serão trocados na reprodução (Figura 3.11).

O "crossover multiponto" é uma extensão natural do cromossomo de dois pontos. Os segmentos gerados devem ser copiados alternadamente para os filhos.

Um outro tipo de crossover mais genérico é o "crossover uniforme" [Syswerda 89]. Cada vez que for realizada uma reprodução cria-se aleatoriamente uma máscara binária de tamanho igual ao dos

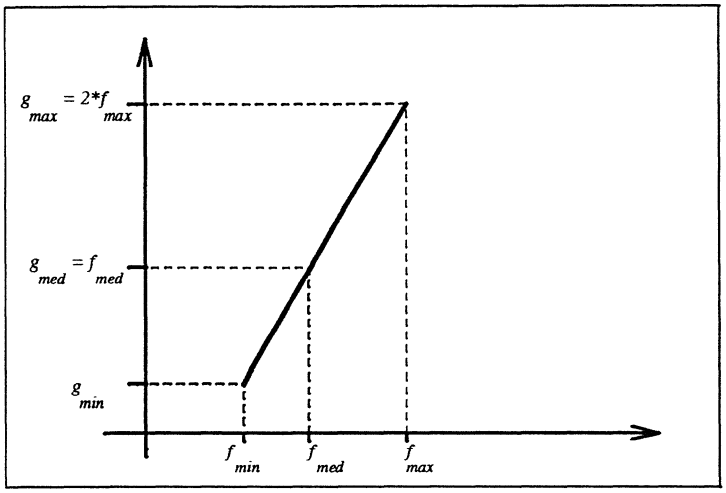


Figura 3.10: Escalonamento da função de avaliação $f(x)$ para uma nova função $g(x)$, mantendo a média e multiplicando o valor máximo por 2.

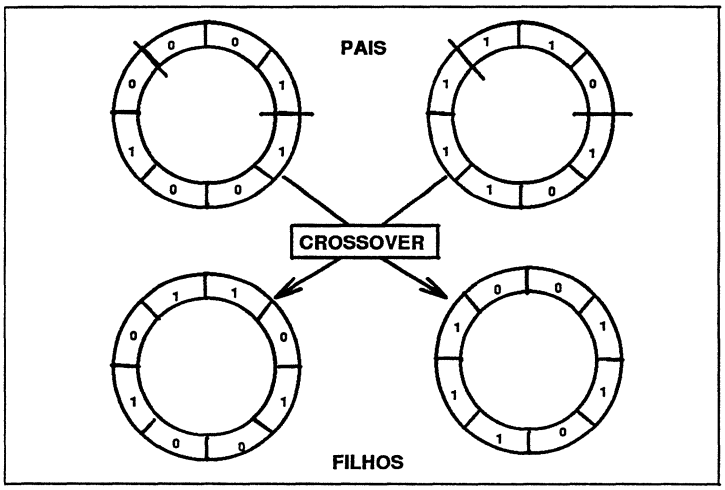


Figura 3.11: O crossover de dois pontos ou de "anel".

cromossomos e procedemos da seguinte forma: Sejam $A = a_1a_2...a_n$ e $B = b_1b_2...b_n$ os cromossomos pais, $C = c_1c_2...c_n$ e $D = d_1d_2...d_n$ os cromossomos filhos e $M = m_1m_2...m_n$ uma máscara binária, onde n é o comprimento dos cromossomos. Se $m_i = 0$, c_i recebe a_i e d_i recebe b_i, caso contrário c_i recebe b_i e d_i recebe a_i (Figura 3.12) [Syswerda 89]. Com o crossover uniforme podemos reproduzir os outros tipos de crossover através de uma máscara apropriada (Figura 3.13).

O que faz diferença na utilização de um tipo de crossover ou de outro é o seu poder exploratório [Schaffer et al. 89] e [Syswerda 89]. O poder exploratório de um tipo de crossover pode ser medido pelo número máximo de configurações alcançáveis pela realização de uma única operação. Supondo L o tamanho do cromossomo, como o crossover de um ponto só pode ocorrer em $L-1$ posições, seu poder exploratório é $2.(L-1)$. A multiplicação por dois é devida ao fato de que uma operação gera dois cromossomos e não apenas um. Já para o caso do crossover uniforme, o número de máscaras

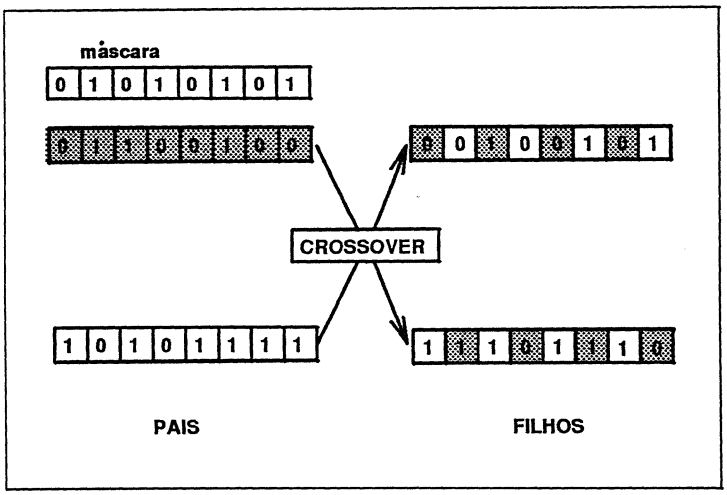


Figura 3.12: O crossover uniforme.

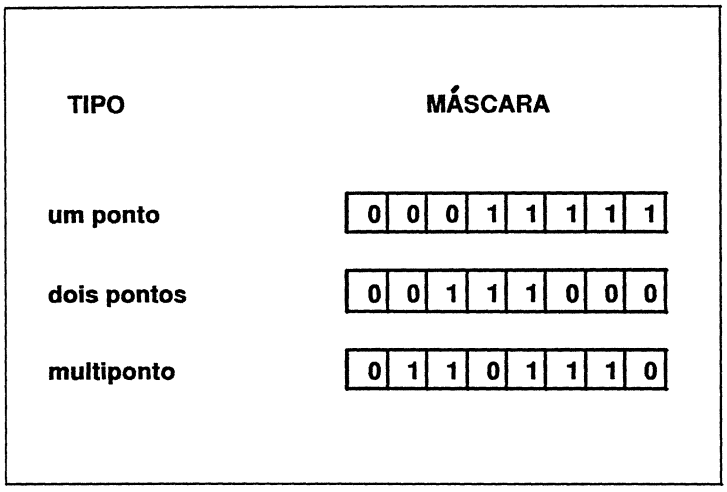


Figura 3.13: Mapeamento de outros tipos de crossover com o crossover uniforme.

binárias de tamanho L possíveis é 2^L, mas pela definição do crossover uniforme, tanto faz usar uma máscara M como o seu complemento $\overline{M}$ que o resultado da operação não se altera. Portanto, para cada operação possível temos duas máscaras equivalentes. Assim, o poder exploratório desse tipo de crossover fica sendo $2^{L+1}/2 = 2^L$, o que define um poder exploratório muito maior para esse tipo de crossover. O número efetivo de configurações alcançáveis depende do número de posições coincidentes entre os cromossomos originais.

O tipo de implementação do crossover também afeta a propagação dos esquemas. Enquanto que para o crossover de um ponto a probabilidade de sobrevivência de um esquema H é $P_s = 1 - d(H)/(L-1)$, para o crossover uniforme será $P_s = 1/2^{O(H)-1}$.

3.6.3 Outras Variações

Na literatura recente há uma classificação das implementações de AGs em três categorias: modelos globais, modelos de ilhas e algoritmos genéticos celulares [Whitley and Gordon 93]. A implementação que apresentamos é a de um Algoritmo Genético Simples, pertencente à categoria dos modelos globais. Outras implementações pertencentes a essa categoria são:

- O Algoritmo Genético "Elitista", onde o melhor indivíduo da geração t é sempre copiado para a geração $t+1$, sem precisar passar por seleção, o que justifica o nome.
- O Algoritmo Genético de Composição, onde os n melhores indivíduos das gerações t e $t+1$ juntas são escolhidos para formar a nova geração.

Na categoria de modelos de ilhas se enquadram as implementações nas quais se executam em paralelo vários AGs com populações simples. Cada "ilha" é um AG simples com sua própria subpopulação. Indivíduos podem migrar de uma "ilha" para outra seguindo uma topologia estabelecida. Essa migração pode ser implementada por processos de seleção como o que foi descrito, ou por processos de seleção "elitistas".

A última categoria, a dos Algoritmos Genéticos Celulares, apresenta implementações onde a população é disposta em uma grade bidimensional e cada indivíduo se reproduz apenas com o melhor dentro de uma vizinhança definida. Os filhos gerados substituem os pais se possuirem graus de adaptação superiores [Whitley and Gordon 93], [Gordon et al. 92].

Não será objetivo do trabalho descrever detalhadamente essas implementações, nem fazer comparações com o modelo apresentado.

3.7 Nichos e Formação de Espécies

No tratamento de problemas multimodais, AGs simples convergem para um único "pico", mesmo havendo outros de igual qualidade (Figura 3.14). Em situações análogas a essa, a Natureza forma subpopulações de organismos em nichos separados forçando que indivíduos similares compartilhem os recursos disponíveis [Deb and Goldberg 89].

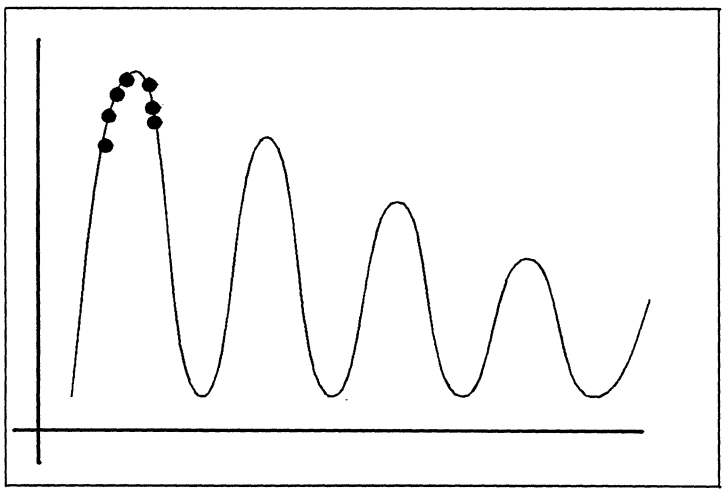


Figura 3.14: Convergência para um único "pico".

Muitas modificações foram tentadas nos AGs para implementação de uma forma análoga a esse compartilhamento. Entre elas podemos citar o método de povoamento proposto por [De Jong 75] e os métodos de compartilhamento propostos por [Goldberg and Richardson 87].

3.7.1 Métodos para Formação de Nichos

Em um problema de otimização de funções multimodais, um AG simples não consegue controlar a competição entre esquemas viáveis correspondentes a diferentes "picos", e fatalmente a convergência se dá para um deles apenas. Apesar da convergência para o melhor "pico" existir, geralmente queremos determinar a localização de outros pontos viáveis (Figura 3.15).

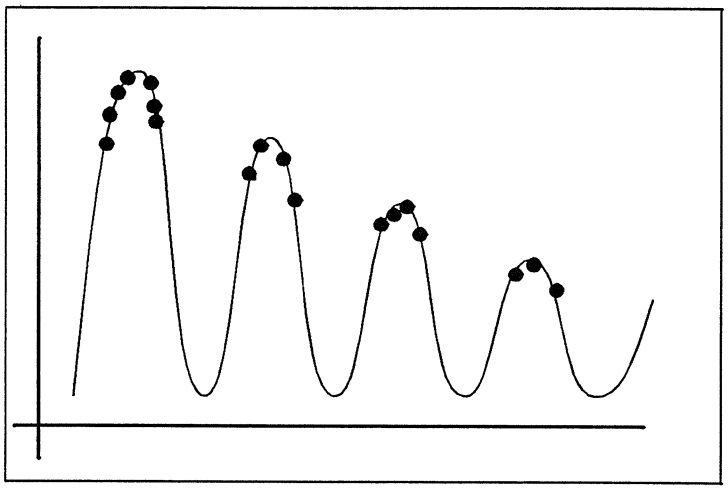


Figura 3.15: Formação de nichos.

Na Natureza, uma espécie é uma coleção de organismos com características similares. A dispersão dos seres vivos nos ambientes reduz a competição entre espécies por recursos ambientais e colabora para a formação de subpopulações em nichos.

3.7.2 Esquema de Povoamento

Em [De Jong 75] nichos separados são criados substituindo-se indivíduos existentes por outros de acordo com um padrão de similaridade. São definidos dois parâmetros para esse propósito: uma taxa de "gap" de geração (G) e um fator de povoamento (CF). O parâmetro G indica a proporção da população que será permitida reproduzir-se. Indivíduos da população serão substituídos (ou "mortos") pelos novos indivíduos gerados, de acordo com o parâmetro CF. Para selecionar um indivíduo para morrer, tomamos aleatoriamente CF indivíduos e escolhemos o mais semelhante, em termos de cromossomo, ao novo indivíduo. De Jong usou esse esquema com sucesso determinando $G = 0.1$ e $CF = 2$ ou 3.

3.7.3 Esquema de Compartilhamento

[Goldberg and Richardson 87] dividiram a população em diferentes subpopulações de acordo com a similaridade de indivíduos sob dois aspectos: fenótipo e genótipo. A formação de nichos se dá pela forma que é feita a avaliação. O grau de adaptação de um indivíduo é calculado em função da presença de outros indivíduos na sua vizinhança definida no espaço dos fenótipos ou no dos genótipos.

Não entraremos nos detalhes desses métodos pois não é esse o objetivo do modelo. O leitor poderá encontrar em [Goldberg and Richardson 87], [Goldberg 89] e [Deb and Goldberg 89] os conceitos mais aprofundados.

3.8 Comparações de AGs com Outros Métodos de Busca

Segundo [Goldberg 89], os algoritmos de busca podem ser agrupados em três classes: algoritmos baseados em cálculo, algoritmos enumerativos e algoritmos aleatórios.

Algoritmos de busca baseados em cálculo necessitam da existência de certas condições tais como continuidade, existência de derivadas, etc..., o que impede a sua aplicação a problemas que não apresentem tais características. São também algoritmos de escopo local, isto é, sua busca por um ponto ótimo, por exemplo a raiz de uma equação, sempre é baseada em um ponto melhor da vizinhança de um ponto conhecido.

Algoritmos enumerativos calculam valores de uma função de mérito para cada ponto de um espaço de busca finito ou infinito discretizado. Apesar da atração que causa por sua semelhança como o modo humano de realizar buscas (em espaços diminutos), tais esquemas não satisfazem por uma razão simples: a maioria dos problemas reais apresenta espaços de busca muito extensos para se pesquisar ponto a ponto.

A primeira classe é de aplicabilidade restrita, pois a maioria dos problemas reais não apresentam as condições requeridas. Portanto, falta-lhe robustez. A segunda, mais robusta, não é satisfatória quando o espaço de busca assume grandes dimensões. A terceira classe, algoritmos de busca aleatória, vêm ganhando popularidade. Mas não satisfazem por completo no tocante à eficiência. Buscas aleatórias que memorizam os melhores pontos encontrados, em uma execução longa, tendem a ter um desempenho não muito diferente das buscas enumerativas [Goldberg 89].

Devemos fazer uma distinção entre métodos de busca essencialmente aleatória e métodos de busca que se valem de técnicas baseadas em aleatoriedade, tais como AGs e "simulated annealing". É importante salientar que técnicas aleatorizadas não implicam necessariamente em busca sem direção. Estas técnicas são heurísticas que utilizam escolhas aleatórias como ferramenta de guia para uma busca eficiente. "Simulated annealing" usa processos aleatórios na sua busca através de mínimos estados de energia. O livro [Davis and Steenstrup 87], citado por [Goldberg 89], explora as conexões entre "simulated annealing" e AGs (Figura 3.16).

A qualidade mais evidente de um AG é sua robustez. Para podermos aplicar um AG a diversos problemas devemos apenas estabelecer os critérios de codificação, interpretação e análise de pontos do espaço de busca em cromossomos. O restante, seleção, reprodução e evolução não se alteram. Um processo de otimização deve procurar aprimorar o desempenho na direção de algum ponto ótimo do espaço de busca. Há uma clara distinção entre o processo de aprimoramento e o ponto ótimo em si. Quando analisamos um algoritmo de busca, em geral o que nos interessa é apenas se ele alcança ou não um ponto ótimo, deixando para um segundo plano o seu desempenho para alcançá-lo.

Consideremos, por exemplo, uma pessoa que deve elaborar testes para um determinado sistema. Que critério poderíamos usar para julgar se essa pessoa fez bem ou mal o seu trabalho? Em geral dizemos que a pessoa fez bem uma tarefa quando ela fez as escolhas adequadas dentro das restrições de tempo e recursos. Nós, portanto, não a julgamos sob o ponto de vista da perfeição absoluta (que equivaleria ao ponto ótimo para o problema), e sim em termos de melhor ou pior, comparativamente. Assim, convergência para um ponto ótimo nem sempre é o mais importante na resolução de muitos dos problemas reais. Para muitos problemas a meta melhor seria a de atingir um nível satisfatório rapidamente. Atingir um ponto ótimo seria, então, um aspecto secundário (Figura 3.17).

AGs podem atuar na resolução de problemas complexos onde outros métodos não apresentam resultados satisfatórios. Por exemplo, um trabalho recente [De Jong and Spears 89] apresenta a utilização de AGs na resolução de problemas NP-Completos. Embora haja consenso que todo problema

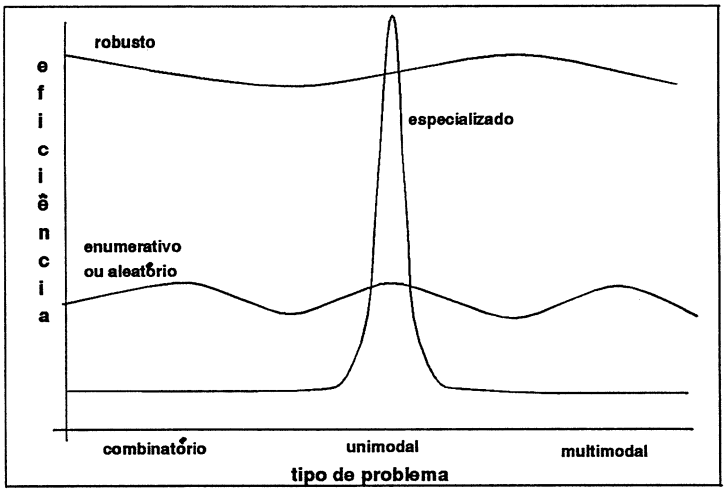


Figura 3.16: Comparação da eficiência de alguns tipos de algoritmos de busca, segundo [Goldberg 89].

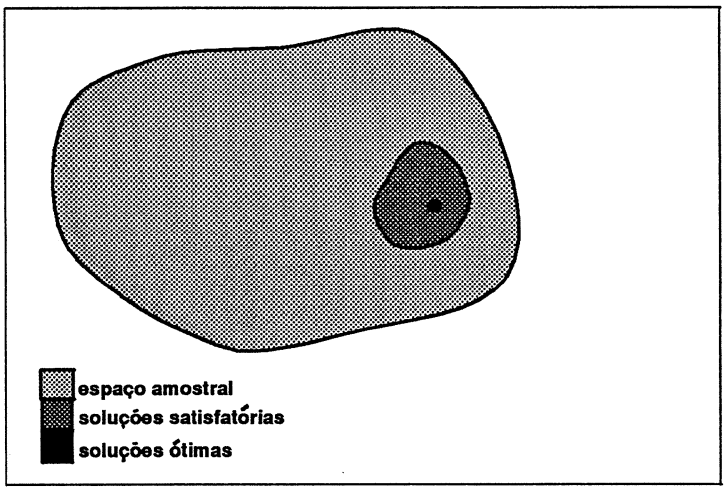


Figura 3.17: Proporção genérica entre o espaço amostral, um conjunto de soluções viáveis e o conjunto de soluções ótimas.

NP-Completo apresente a mesma dificuldade computacional de resolução, um AG pode atuar melhor sobre alguns tipos de problemas NP-Completos do que em outros. Uma vez que qualquer problema NP-Completo pode ser convertido para outro problema do mesmo tipo em tempo polinomial, a estratégia descrita naquele trabalho consiste em identificar um problema NP-Completo canônico no qual um AG atue satisfatoriamente, e dessa forma resolver outros problemas indiretamente por redução a este problema canônico.

Contudo, levando em conta as dificuldades de codificação e avaliação que muitos problemas podem apresentar, AGs podem ainda não ser a melhor alternativa para resolução de certos problemas onde métodos exatos ou heurísticas mais especializadas possam ter desempenho superior.

3.9 Aplicações de Algoritmos Genéticos

Relacionamos aqui uma série de aplicações citadas por [Goldberg 89], identificadas pela área, ano, autor(es) e descrição:

BIOLOGIA

- 1967 - Rosemberg - Simulação da evolução de populações de organismos unicelulares.

CIÊNCIA DA COMPUTAÇÃO

- 1985 - Rendell - Busca de função de avaliação em jogos.

ENGENHARIA E PESQUISA OPERACIONAL

- 1982 - Etter, Hicks e Cho - Projeto de filtros adaptativos usando AG simples.
- 1983 - Goldberg - Otimização de gasodutos.
- 1985 - Davis e Smith - Layout de circuitos VLSI com AGs.
- 1986 - Minga - Otimização de transporte de cargas aéreo com AG.

PROCESSAMENTO DE IMAGENS E RECONHECIMENTO DE PADRÕES

- 1970 - Cavicchio - Seleção de detetores para reconhecimento de padrões binários.
- 1984 - Fitzpatrick e outros - Registro de imagens com AG minimizando diferenças

CIÊNCIAS SOCIAIS

- 1979 - Reynolds - Adaptação do modelo de comportamento caçador-coletor pré-histórico.
- 1981 - Smith e De Jong - Calibração do modelo de migração de populações usando busca genética.

Capítulo 4

Um Modelo Computacional Evolutivo Baseado em uma Arquitetura Cliente-Servidor

4.1 A Arquitetura "CLIENTE-SERVIDOR"

Suponha que vários computadores estejam conectados entre si por uma rede de comunicação de dados, e que um desses computadores concentre em si recursos computacionais necessários a sistemas residentes nos outros computadores.

Quando temos uma situação como essa, em que uma máquina provê recursos computacionais para outras máquinas, conectadas em rede, atendendo a solicitações, dizemos que esse conjunto de máquinas está organizado sob uma arquitetura "CLIENTE-SERVIDOR". A máquina que fornece os recursos é o SERVIDOR e as outras, os CLIENTES (Figura 4.1).

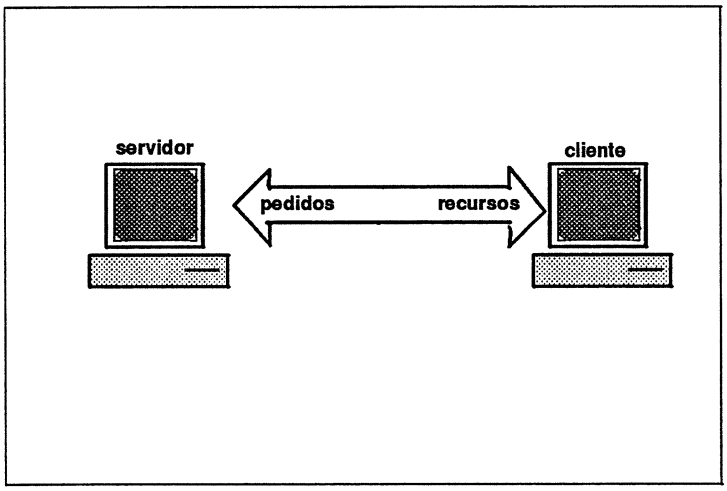


Figura 4.1: Relacionamento de duas máquinas em uma arquitetura cliente-servidor.

O objetivo de uma arquitetura desse tipo é concentrar em uma máquina os recursos computacionais de um determinado ambiente e diminuir a carga de outras máquinas, eliminando também uma redundância desnecessária. Entre os vários sistemas que fazem uso dessa arquitetura podemos citar o

"correio eletrônico" e alguns sistemas de banco de dados multi-usuário.

Qualquer máquina pode ser servidor ou cliente, sendo até comum as duas coisas acontecerem ao mesmo tempo. Na máquina servidora deve existir um processo atento às solicitações da rede e que ativa os processos fornecedores dos recursos solicitados. Esses processos ficam em hibernação até serem ativados. Ao fim do atendimento voltam a hibernar. O cliente para usar o recurso que necessita, deve conectar-se ao servidor e solicitar o recurso.

4.2 Algoritmos Genéticos e a Arquitetura "Cliente - Servidor"

As principais tarefas de um AG são: codificação, avaliação, estatística populacional, seleção e reprodução. Dentre essas tarefas algumas são independentes do contexto da aplicação e outras são afetadas pelos aspectos particulares de cada aplicação. As tarefas dependentes são:

- Codificação/Tradução: mapeamento dos pontos dos espaço de busca do domínio relativo à aplicação (fenótipos) para seqüências finitas de símbolos de um alfabeto (cromossomos) manipuláveis geneticamente. Para cada problema a ser resolvido os critérios para realização desta tarefa são alterados;

- Avaliação: análise do ponto traduzido a partir dos cromossomos do ponto de vista do problema a ser resolvido. Nesta tarefa está a implementação dos aspectos do problema propriamente dito;

- Estatística: embora seja sempre realizada da mesma forma, independente da aplicação, é através dela que se obtém a condição de parada da aplicação;

- Escolha de parâmetros: determinação de tamanhos de população, de cromossomos e taxas de ocorrência de operações genéticas a serem fornecidas para as outras tarefas;

As tarefas independentes são:

- Seleção: escolha dos cromossomos que formarão os pares para reprodução. Para realização desta tarefa basta a população com os seus indivíduos devidamente avaliados. Não interessando os aspectos de codificação;

- Reprodução: manipulação genética dos cromossomos (crossover e mutação). Só necessita das taxas de ocorrência definidas e do tamanho dos cromossomos.

- Comunicação: implementação de facilidades que permitam a troca de informações entre as tarefas dependentes e as independentes, bem como entre aplicações de AGs em paralelo.

Podemos portanto dividir um AG em dois módulos funcionais: um genérico para manipulação populacional, e outro específico, que reflete os aspectos intrínsecos da aplicação (Figura 4.2). Os métodos de manipulação populacional são portanto recursos computacionais possíveis de serem fornecidos por um servidor a quaisquer aplicações que deles necessitem. Assim podemos ter em uma máquina um servidor que manipula populações, e em outra, uma aplicação que decodifica, avalia e atribui graus de adaptação. O servidor envia ao cliente uma população e este retorna o grau de adaptação de seus indivíduos para que o servidor possa gerar uma nova população.

Seguindo essa arquitetura podemos multiplexar os clientes mantendo apenas um servidor. Assim esse servidor pode controlar concomitantemente mais do que uma aplicação. Para nosso uso chamaremos de CENTRO o processo servidor, e de APLICAÇÃO o processo cliente (Figura 4.3).

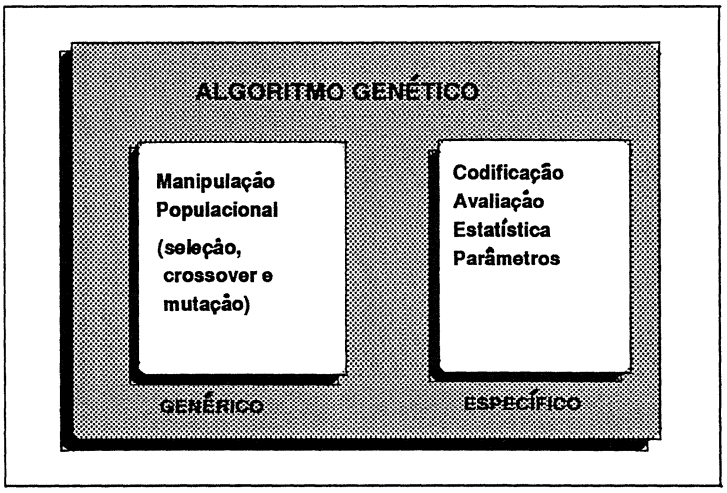


Figura 4.2: Divisão modular de um Algoritmo Genético.

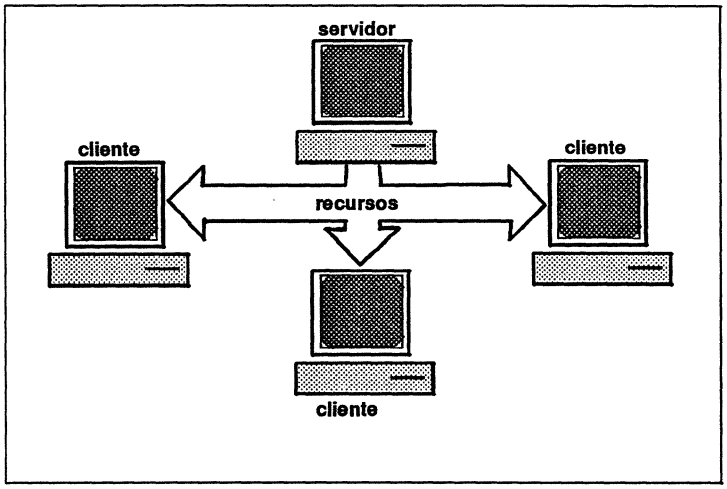


Figura 4.3: Arquitetura cliente-servidor multiplexada.

4.3 Relacionamento CENTRO-APLICAÇÃO

Vamos descrever aqui como seria a execução de uma sessão de uma única APLICAÇÃO no modelo proposto (Figura 4.4).

1. O processo CENTRO está hibernando, esperando ser ativado quando chegar alguma solicitação pela rede.
2. A APLICAÇÃO adquire os parâmetros necessários tais como: alfabeto, tamanho dos cromossomos, tamanho da população e probabilidades de crossover e de mutação; conecta-se ao CENTRO e faz um pedido de registro enviando-lhe os parâmetros adquiridos.

3. Se não houver qualquer impedimento o CENTRO aceita o pedido de registro, cadastra os parâmetros e retorna à APLICAÇÃO um identificador. Caso contrário retorna um status de erro. A partir deste ponto o identificador retornado deverá ser utilizado pela APLICAÇÃO sempre que esta se comunicar com o CENTRO.
4. A APLICAÇÃO recebe a resposta do CENTRO. Se não houve erro, gera uma população inicial.
5. A APLICAÇÃO analisa a população. Se alguma condição de parada for satisfeita, a APLICAÇÃO termina e notifica o CENTRO. Caso contrário, calcula e atribui os graus de adaptação aos indivíduos e envia a população, com a avaliação, ao CENTRO junto com seu identificador.
6. Caso receba uma notificação de término, o CENTRO remove o cadastro da aplicação, libera seu identificador e volta ao passo 1. Caso contrário, prossegue para o passo 7.
7. O CENTRO recebe a população, consulta os parâmetros indexados pelo identificador fornecido e, com base no grau de adaptação dos indivíduos, usando os operadores de seleção, crossover e mutação, gera uma nova população e envia-a para a APLICAÇÃO. Volta ao passo 5.

No caso de haver mais de uma aplicação, depois de enviar uma nova população para uma APLICAÇÃO X, enquanto não recebe de volta a população avaliada, o CENTRO pode atender outras APLICAÇÕES distintas de X. Isso é possível graças ao identificador fornecido, pelo qual o CENTRO pode consultar univocamente os parâmetros relativos a cada aplicação cadastrada.

4.4 Protocolo de Comunicação CENTRO-APLICAÇÃO

Vimos no ítem anterior que há três tipos de comunicação da APLICAÇÃO para o CENTRO: pedido de cadastro, envio de população avaliada e notificação de fim da APLICAÇÃO. Para que o CENTRO possa tratar cada caso corretamente, foi criado um pequeno protocolo (Figura 4.6). A APLICAÇÃO, ao se comunicar com o CENTRO deve incluir um código que define qual a finalidade da comunicação (Figura 4.5). Junto com o pedido de cadastro "CAD", a APLICAÇÃO envia os parâmetros adequados aos seus objetivos. Na fase operacional a APLICAÇÃO usa o código "AVL" para enviar uma população avaliada ao CENTRO, identificada pelo seu número. Para finalizar seu processo, a APLICAÇÃO envia um código "FIM" e seu identificador ao CENTRO para que este possa liberar as áreas de memória que utilizou no cadastro e relacionamentos com a APLICAÇÃO.

Só o CENTRO precisa desses códigos para tratamento correto, pois ele nunca sabe que tipo de solicitação virá da rede no caso de haver mais de uma aplicação "no ar". Já a APLICAÇÃO não necessita disso pois sua implementação reflete a expectativa do que deve receber do CENTRO.

A emissão e a recepção de populações e avaliações entre cliente e servidor poder ser definida de duas formas:

1. O servidor, após enviar uma população ao cliente, libera a área de memória usada por essa população para uso em relacionamentos com outros clientes. O cliente, ao retornar a avaliação, inclui também a população que recebeu para avaliar, permitindo ao servidor gerar nova população.
2. O servidor mantém a população mesmo após o envio ao cliente. O cliente retorna ao servidor apenas a avaliação dos indivíduos da população que recebeu.

A primeira opção mantém o servidor menos carregado, permitindo a este atender um número maior de clientes sem afetar seu desempenho. Em contrapartida, a rede ficará mais carregada. A segunda

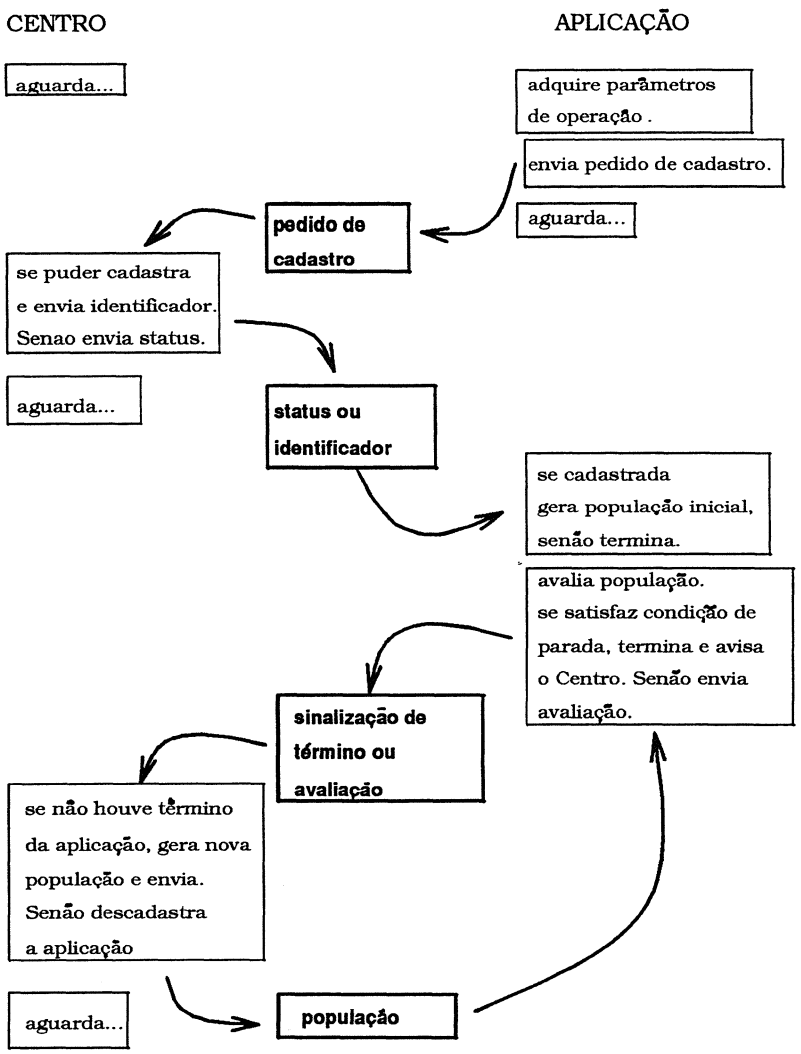


Figura 4.4: Relacionamento entre o CENTRO e uma APLICAÇÃO.

opção representa o oposto da primeira, carregando mais o servidor e liberando mais a rede. A escolha de uma forma ou de outra depende dos critérios de cada tipo de aplicação. Por exemplo: se o tempo de resposta de um cliente ao servidor for muito grande, convém usar a primeira opção. Se, por outro lado, houver muita soilicitação da rede, a segunda opção é mais adequada.

4.5 Exemplos

Para ilustrar a separação dos módulos genérico e específico de um AG, vamos mostrar como duas aplicaç oes completamente distintas podem ser codificadas de modo muito semelhante de forma a

código	função
CAD	Pedido de cadastro de aplicação
AVL	Envio de Avaliação
FIM	Sinalização de término de aplicação

Figura 4.5: Códigos de comunicação cliente-servidor.

CAD
nome da aplicação
alfabeto
tamanho da população
tamanho dos cromossomos
probabilidade de crossover
probabilidade de mutação

AVL	
identificador	
indivíduo	grau
indivíduo	grau
indivíduo	grau
.	
.	
.	
indivíduo	grau

FIM
identificador

Figura 4.6: Protocolos de comunicação cliente-servidor.

possibilitar sua resolução no ambiente proposto. Como veremos, a codificação para o primeiro exemplo será direte, enquanto que para o segundo, um pouco mais elaborada.

Usaremos duas aplicações que utilizarão compartilhadamente o CENTRO: uma resolução de equação de segundo grau, onde ilustramos uma forma de codificação simples e direta, e também a estratégia de avaliação para um problema simples; e um problema de caixeiro viajante, no qual procuramos mostrar como um problema de codificação mais complexa pode ser adaptado para os recursos oferecidos pelo servidor.

4.5.1 Aplicação I : Busca do Valor Máximo de uma Equação

Vamos projetar aqui uma aplicação para a busca do valor máximo da equação

$$y = -x^2 + 35.x - 250$$

no intervalo $[0, 31]$ (Figura 4.7). Utilizando um alfabeto binário e um cromossomo de cinco posições representando um número inteiro, poderemos cobrir o espaço amostral proposto (Figura 4.8).

A atribuição do grau de adaptação será feita da seguinte forma:

1. Para cada indivíduo, calculamos o seu fenótipo, ou seja traduzimos o conteúdo de seu cromossomo para um número inteiro x no intervalo $[0, 31]$.
2. Com o valor de x traduzido, calculamos o valor de x segundo a equação que queremos resolver.
3. Atribuímos ao indivíduo um grau de adaptação $N(x)$ estritamente positivo e proporcional a y (Figura 4.9). Cada tipo de $N(x)$ tem uma influência distinta sobre o processo evolutivo, promovendo uma convergência mais rápida ou mais lenta.

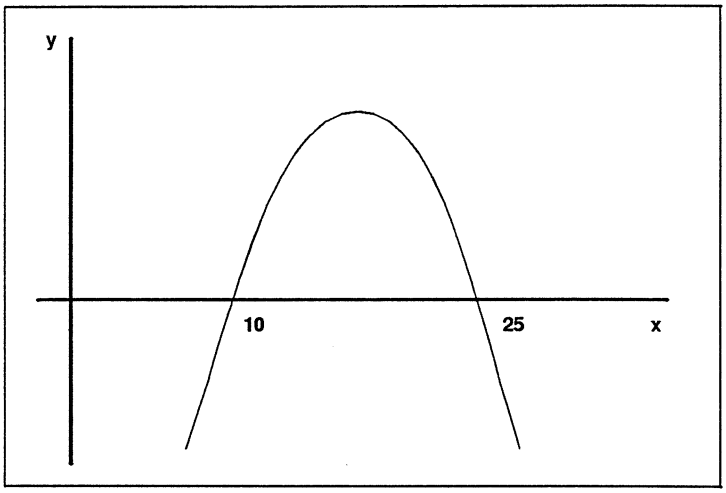


Figura 4.7: Gráfico da função $y = -x^2 + 35.x - 250$.

cromossomo	fenótipo
00000	0
00001	1
00010	2
.....	...
.....	...
.....	...
11101	29
11110	30
11111	31

Figura 4.8: Tradução cromossomo-fenótipo para a aplicação I.

4.5.2 Aplicação II: Resolução de um Problema de Caixeiro Viajante

O problema do caixeiro viajante consiste em percorrer um determinado número de cidades e voltar à cidade de origem, pelo menor percurso, sem passar duas vezes pela mesma cidade. É um problema cujo espaço amostral tem uma dimensão da ordem de $(N-1)!$, onde N é o número total de cidades. Em termos matemáticos o problema consiste em: dado um grafo com um certo número de vértices, todos conectados entre si por uma aresta, achar um ciclo hamiltoniano mínimo neste grafo (Figura 4.10). Ciclo hamiltoniano é um caminho partindo de um determinado vértice e chegando a esse mesmo vértice passando por todos os outros vértices uma única vez.

Essa aplicação apresenta uma complexidade maior para codificação. A primeira idéia que nos vem à mente é adotar um alfabeto onde cada símbolo corresponda a uma cidade, e fazer com que cada cromossomo represente um caminho possível. Porém teríamos um problema com essa codificação: Nem sempre o crossover de dois cromossomos que representam ciclos hamiltonianos originaria novos ciclos hamiltonianos, podendo gerar pontos fora do espaço amostral (Figura 4.11).

Existem várias implementações em AGs para o problema do caixeiro viajante, mas que utilizam particularizações nos métodos de reprodução como forma de garantir a geração de ciclos hamiltonianos

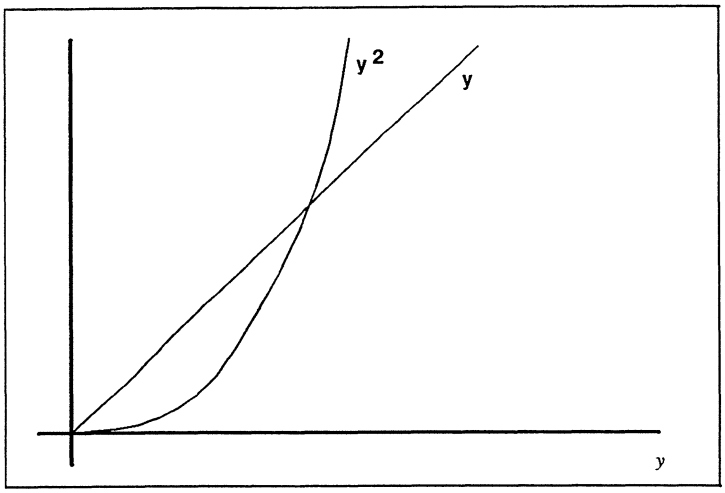


Figura 4.9: Exemplos de funções de atribuição do grau de adaptação para a aplicação I.

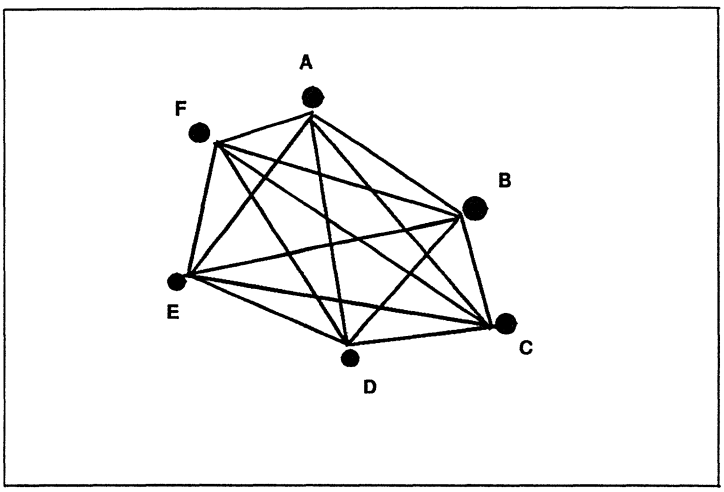


Figura 4.10: Grafo completo mostrando todos caminhos possíveis entre um conjunto de cidades.

[Hamaifar et al. 93] e [Whitley 89]. Essas particularizaç oes procuram garantir o uso das cidades como símbolos do alfabeto cromossômico. Entretanto, seu uso descaracteriza a robustez atribuída ao AG, tornando-o assim um esquema mais especializado. Uma forma de codificação adequada ao serviços supridos pelo CENTRO, e que resolve esse problema, é a adoção de uma notação posicional, que ilustraremos através do seguinte exemplo:

Seja $C_0 = \{A, B, C, D, E\}$ o conjunto de cidades a percorrer. O caminho será de seis posições, com a primeira e a última posição ocupadas pela mesma cidade. Supondo que o caixeiro está inicialmente em A, a primeira e a última posição do caminho já estão naturalmente resolvidas. A penúltima posição também estará automaticamente resolvida quando as anteriores já tiverem sido ocupadas, pois será a última cidade que sobrar. Os cromossomos devem codificar apenas o preenchimento das três posições intermediárias, destacadas na Figura 4.13. Temos para o preenchimento dessas posições o conjunto $C_1 = \{B, C, D, E\}$. A segunda posição poderá ser ocupada por qualquer um dos elementos de C_1,

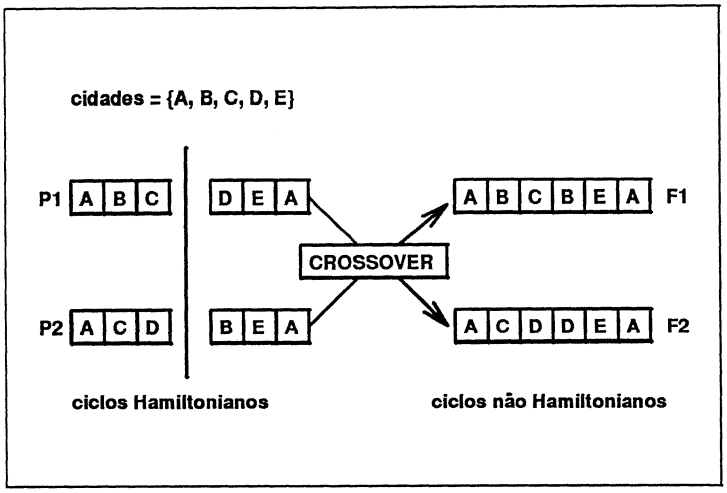


Figura 4.11: Crossover em cromossomos definidos sobre um alfabeto formado pelas cidades a serem percorridas.

suponhamos que seja pelo terceiro elemento D. Ficam sobrando as cidades do conjunto $C_2 = \{B, C, E\}$. Colocamos na terceira posição o primeiro elemento de C_2, B, e sobra $C_3 = \{C, E\}$. O segundo elemento de C_3, E, vai para a quarta, e o que sobrou, C, para a quinta (penúltima) posição. Pela figura 4.12, podemos verificar que os mesmos dois ciclos hamiltonianos da figura figura 4.11, sob essa representação, quando submetidos ao crossover produzem, agora, outros dois ciclos hamiltonianos.

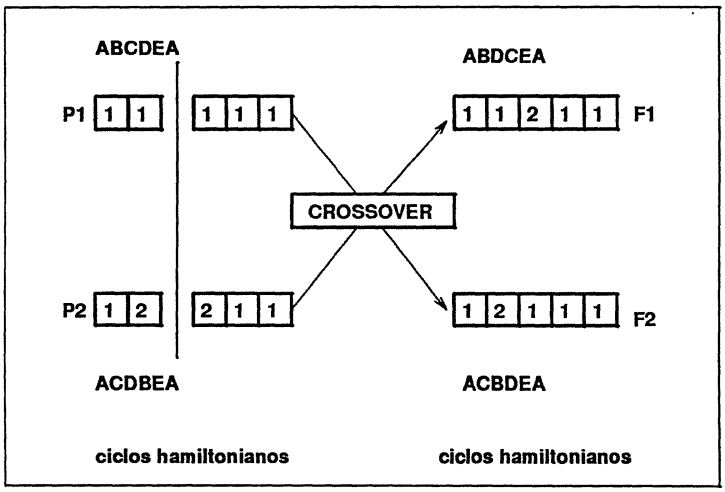


Figura 4.12: Crossover de dois ciclos hamiltonianos em representação posicional.

Poderíamos utilizar como alfabeto o conjunto de símbolos $V = \{1, 2, 3, 4\}$, que representaria os índices dos elementos do conjunto C_i restante do preenchimento da i-ésima posição. Porém, a cada posição preenchida certos símbolos do alfabeto ficariam incompatíveis por representar índices de cidades não disponíveis para preenchimento da posição. A operação de mutação precisaria ter noção da posição em que está atuando para trocar o símbolo corrente por um outro símbolo compatível para

posição	C_i	índice	cidade	caminho
1	{A, B, C, D, E}	1	A	A ? ? ? A
2	{B, C, D, E}	3	D	A D ? ? A
3	{B, C, E}	1	B	A D B ? ? A
4	{C, E}	2	E	A D B E ? A
5	{C}	1	C	A D B E C A

Figura 4.13: Montagem de um ciclo hamiltoniano com base nos índices de elementos dos conjuntos C_i.

a posição. Por exemplo: a posição 2 pode ser ocupada pelos índices 1, 2, 3 ou 4 por haver quatro cidades disponíveis; mas a posição 4 só pode ser ocupada pelos índices 1 ou 2 porque, como a segunda e a terceira já estão ocupadas, sobram apenas duas cidades disponíveis.

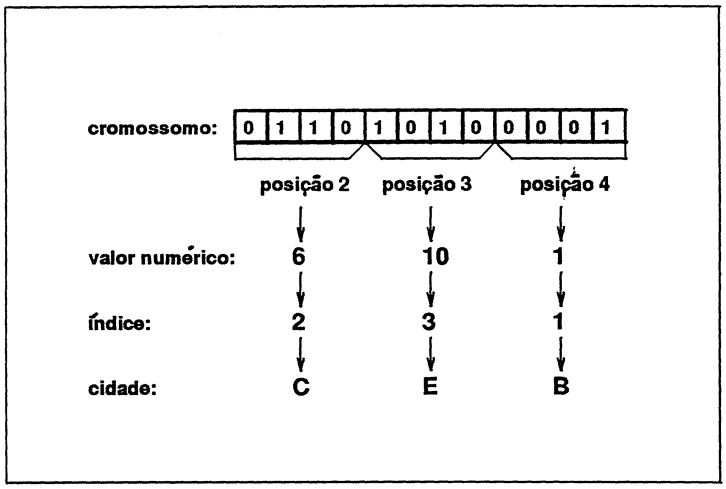


Figura 4.14: Esquema de codificação de índices através de segmentos binários.

Para evitar esse problema, utilizamos um alfabeto binário e cromossomos subdivididos em 3 segmentos de igual tamanho, um para cada posição a ser preenchida (Figura 4.14). Para o valor numérico traduzido de cada um dos segmentos do cromossomo, faremos uma proporção para o intervalo de índices possíveis para a posição que o segmento representa (Figura 4.15). Por exemplo, se usarmos uma seqüência de quatro símbolos binários, para a segunda posição teríamos os valores numéricos (0,1,2,3) correspondendo ao índice 1, os valores (4,5,6,7) ao índice 2, (8,9,10,11) ao índice 3 e (12,13,14,15) ao índice 4. Para a terceira posição, porém, os valores (0,1,2,3,4) corresponderiam ao índice 1, (5,6,7,8,9) ao índice 2 e (10,11,12,13,14,15) ao índice 3, o que daria uma probabilidade um pouco maior para ocorrências desse índice. Essa probabilidade tenderá a se igualar à dos outros índices na medida que aumentarmos o tamanho das seqüências.

Resumindo e generalizando: Se o conjunto de cidades possui tamanho N, os cromossomos precisam representar apenas $N-2$ cidades do caminho, pois a primeira já é conhecida pois é a cidade de partida do caixeiro, a última não conta pois deve ser igual à primeira, e a penúltima é ocupada pela cidade que sobrar do preenchimento das outras posições. Devemos escolher o tamanho l dos segmentos dos cromossomos tal que $2^l \geq N$. Por exemplo: pela figura 4.14, o cromossomo 011010100001, com três segmentos de quatro símbolos, representaria o seguinte caminho: o primeiro segmento tem valor

config. segmento	valor	índices		
		posição 2	posição 3	posição 4
0 0 0 0	0	1	1	1
0 0 0 1	1	1	1	1
0 0 1 0	2	1	1	1
0 0 1 1	3	1	1	1
0 1 0 0	4	2	1	1
0 1 0 1	5	2	2	1
0 1 1 0	6	2	2	1
0 1 1 1	7	2	2	1
1 0 0 0	8	3	2	2
1 0 0 1	9	3	2	2
1 0 1 0	10	3	3	2
1 0 1 1	11	3	3	2
1 1 0 0	12	4	3	2
1 1 0 1	13	4	3	2
1 1 1 0	14	4	3	2
1 1 1 1	15	4	3	2

Figura 4.15: Conversão de valores numéricos de segmentos em índices de cidades segundo a posição a ser preenchida.

numérico 6, que, pela proporção, indica o elemento 2 do conjunto $C_1 = \{B, C, D, E\}$, isto é, a cidade C será a segunda cidade do caminho. O segundo segmento tem valor 10, que, pela proporção, indica o elemento 3 do conjunto $C_2 = \{B, D, E\}$ portanto a terceira cidade do caminho será E. A terceira porção tem valor 1, que indica o primeiro elemento de $C_2 = \{B, D\}$, e assim a quarta cidade será B. Conseqüentemente D será a penúltima cidade do caminho, porque foi a que sobrou, e A a última.

Finalmente, a atribuição dos graus de viabilidade dos indivíduos se faz em proporção inversa ao tamanho do caminho descrito por eles . Assim podemos utilizar funções de avaliação que tenham um perfil tal como o sugerido pelas curvas descritas na figura 4.16.

Convém salientar que essa implementação do problema de Caixeiro Viajante é original do autor, não tendo sido encontrada nenhuma referência a algum método semelhante na bibliografia mencionada.

4.5.3 Comentários sobre as aplicações

Como vimos através desses dois exemplos, um AG pode ser portado de um problema para o outro apenas com alterações de codificação e avaliação.

O estabelecimento de critérios para a utilização de um AG para um determinado problema, bem como para codificação de pontos de um espaç de busca, é um aspecto importante, porém, por requerer uma descrição muito extensa e a análise de diversos tipos de problema, e também por ser uma questão ainda em aberto, não será abordado neste trabalho, Podemos dizer que, de um modo genérico, para problemas com grande espaço de busca, ou para os quais inexiste uma heurística apropriada, os AGs têm se mostrado uma ferramenta adequada. Entretento, para alguns problemas, as tarefas de codificação e avaliação podem tornar-se tão intrincadas, que pode vir a ser aconselhável a descaracterização de alguns aspectos genéricos do AG e torná-lo mais especializado, ou hibridizá-lo com outras técnicas; ou usar definitivamente uma heurística especialista.

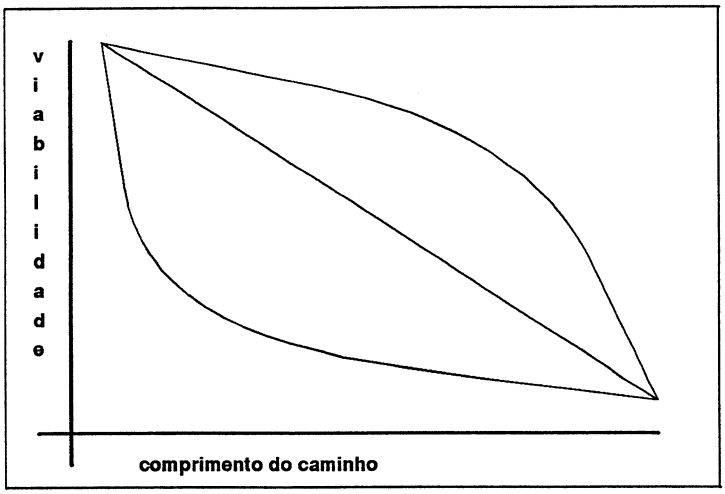


Figura 4.16: Curvas de avaliação possíveis para o problema do caixeiro viajante.

4.6 Comunicação Inter-Clientes

Se quisermos utilizar este modelo para resolver um determinado problema, usando para isso duas ou mais aplicações que compartilharão os recursos do CENTRO, seria interessante que fosse provida alguma forma de comunicação entre essas aplicações (Figura 4.17). Essa comunicação até poderia ficar a cargo da implementação das aplicações pois não teria nenhuma afinidade com as atividades do CENTRO. Porém, para que essa comunicação entre aplicações seja possível, tanto para o caso de aplicações sendo executadas dentro da mesma máquina, como para o caso de serem executadas em máquinas diferentes, seria necessária a implementação de um outro sistema cliente-servidor semelhante a um sistema de "correio eletrônico".

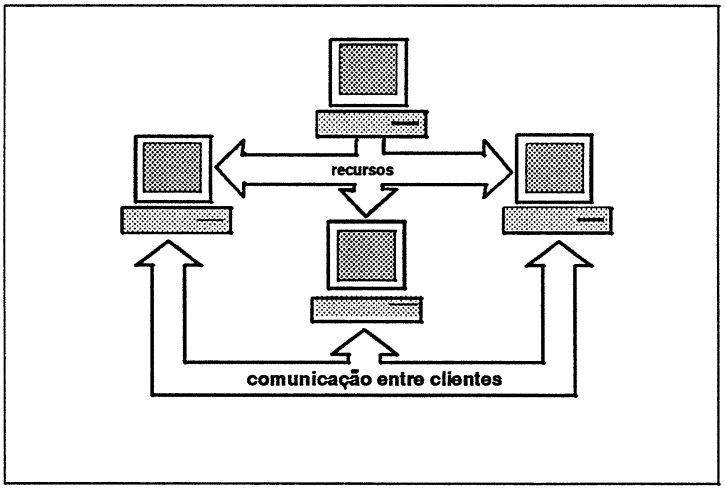


Figura 4.17: Troca de informações entre clientes.

Com a inclusão de dois novos tipos de solicitação ao CENTRO e uma pequena mudança no formato da mensagem do CENTRO para a APLICAÇÃO, podemos prover essa facilidade no próprio CENTRO.

Esses dois novos tipos de solicitação são: o pedido de consulta da lista de aplicações correntes (para que a aplicação que queira enviar uma mensagem possa conhecer o identificador da aplicação para a qual deseja enviar essa mensagem) e o pedido de envio da mensagem propriamente dita (com os identificadores das aplicações origem e destino). A alteração a ser feita no formato de mensagem do CENTRO/APLICAÇÃO é a inclusão de um campo de comunicação a seguir ao campo de população. Esse campo serve para o CENTRO informar à APLICAÇÃO a lista de aplicações correntes, bem como para colocar a mensagem enviada a disposição da aplicação destino. Os códigos dos novos pedidos são: "LIS" para solicitação da lista de aplicações e "MSG" para envio de mensagem (Figura 4.18)

código	função
LIS	Pedido de lista de aplicações.
MSG	Pedido de envio de mensagem.

Figura 4.18: Códigos para implementação da comunicação entre clientes.

Essa comunicação entre clientes é útil na implementação de aplicações co-evolutivas, isto é, um conjunto de aplicações dedicadas à resolução de um único problema através da interferência de uma no processo evolutivo das outras.

Com a inclusão dos novos tipos de comunicação, o protocolo de operação sofre uma pequena alteração e é criado um novo protocolo para o envio de mensagem (Figura 4.19).

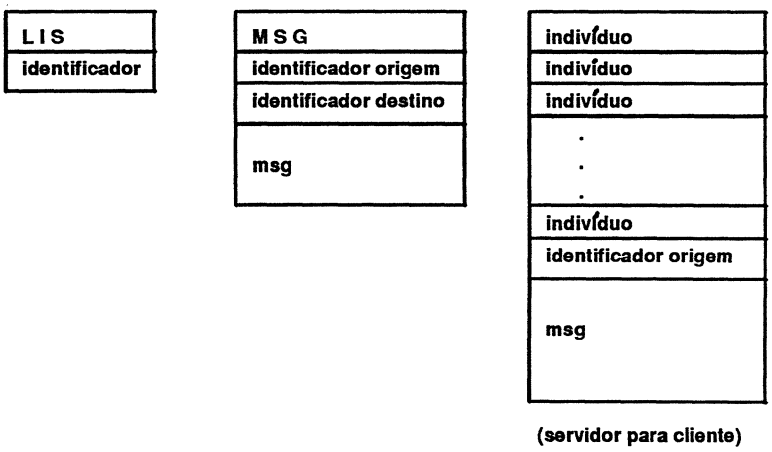


Figura 4.19: Protocolos para comunicação entre clientes.

Capítulo 5

Aplicações Co-Evolutivas

5.1 Introdução

Como foi citado anteriomente, um AG busca soluções de um problema promovendo, geração a geração, a evolução de um subconjunto inicial do espaco de busca até que seu objetivo seja atingido. Na busca de soluções a um determinado problema, podemos utilizar dois ou mais AGs em paralelo, permitindo a comunicação de dados entre eles.

Aplicações de AGs co-evolutivas são aplicações nas quais um conjunto de AGs está dedicado à resolução de um determinado problema e cada AG individualmente possui seu próprio processo evolutivo, comunicando aos outros a ocorrência de determinados eventos durante sua execução e induzindo um processo evolutivo global entre os elementos da aplicação [Husbands et al. 91]. Como exemplo de comunicação podemos citar a transmissão, entre os AGs, dos melhores indivíduos de cada geração de cada AG.

Cada AG dentro de uma aplicação co-evolutiva é um agente autônomo com população e processo evolutivo próprios. O processo evolutivo pode ser aprimorado através do conhecimento de situações ocorridas nos processos evolutivos dos outros AGs presentes na aplicação.

Uma aplicação co-evolutiva não precisa ser necessariamente homogênea, isto é, os AGs que a compõem não precisam ser idênticos nem precisam realizar as mesmas tarefas. Um AG pode estar atento a um determinado aspecto do problema enquanto que outros podem estar atentos a outros aspectos. Ainda mais, podem haver inclusive casos híbridos onde alguns agentes da aplicação sejam AGs e outros não; por exemplo, uma aplicação onde um de seus agentes receba os melhores indivíduos das gerações dos outros agentes, faça uma análise e comunique de volta a eles alguma informação para correção dos seus processos evolutivos.

5.2 Exemplo de Aplicação Co-Evolutiva

Para ilustrar o conceito de co-evolução vamos mostrar aqui uma aplicação - desenvolvida originalmente pelo autor - dedicada à busca de raízes de uma função $f(x)$ qualquer dentro de um intervalo especificado. A única informação que precisaremos a respeito da função para a qual calcularemos as raízes é: para um dado x codificado no cromossomo de um indivíduo, qual o valor retornado por $f(x)$.

Métodos tradicionais de cálculo de raízes, além de calcularem apenas uma raiz, necessitam de mais informações, tais como:

- Método de Newton: continuidade, existência de derivadas em todos os pontos do intervalo pesquisado e, para um dado x pesquisado, a derivada da função nesse ponto não pode ser nula;
- Método das Secantes: continuidade;

- Método das Aproximações Sucessivas: continuidade, diferenciabilidade e pode divergir para certos casos.

Também, como não sabemos com antecedência quantas raízes há no intervalo especificado, não poderemos saber quantas vezes teremos que executar tais métodos.

A aplicação apresentada aqui, a depender dos critérios escolhidos, possivelmente não será mais eficiente do que os métodos acima citados, para funções que obedeçam às restrições por eles impostas. Para outras funções não existe base para comparação.

5.2.1 Codificação dos Indivíduos

Seja $f(x)$ uma função cujas raízes queremos descobrir no intervalo $[0, 4095]$. A codificação mais imediata que nos ocorre é a representação binária convencional com 12 símbolos. Entretanto essa forma de codificação apresenta um problema.

Seja $C_r = 100000$ $(x = 32)$ um cromossomo que represente uma raiz a ser encontrada. Suponha que em determinada geração apareça um indivíduo cujo cromossomo $C_1 = 011111$ $(x = 31)$. Sem dúvida alguma C_1 terá uma ótima avaliação por se tratar de um indivíduo vizinho ao indivíduo raiz, tanto em fenótipo como em genótipo, e conseqüentemente, poderá aparecer em vários pares de reprodução. Como não existem posições coincidentes entre C_r e C_1, uma única aplicação de crossover, sobre C_1 e qualquer outro cromossomo, não será capaz de gerar C_r sem que haja um número muito grande de mutações. Igualmente improvável seria o surgimento de C_r a partir de outro tipo de par de cromossomos, uma vez que sua formação seria prejudicada pela alta viabilidade de C_1.

Uma codificação melhor seria a que na transição de um genótipo para seu vizinho ocorresse a alteração de apenas um símbolo. O código de Gray utilizado em Eletrônica Digital se encaixa exatamente nessa descrição. A Figura 5.1 mostra uma tabela de transição entre código binário convencional e código de Gray. Usaremos então essa forma de codificação para representar os indivíduos x candidatos à raiz da função $f(x)$.

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

Figura 5.1: Tabela de conversão Gray-Binário.

5.2.2 Avaliação dos Indivíduos

Utilizaremos para atribuição de grau de viabilidade uma função de avaliação $N(x)$,

$$N(x) = \frac{1}{(1+k)^{|f(x)|}}$$

que forneça graus de viabilidade tão mais próximos de 1.0 quanto $f(x)$ estiver próximo de 0, e tão mais próximo de 0.0 quanto maior for $\mid f(x) \mid$. Dessa forma promove-se a convergência para os indivíduos que representem valores de x que façam $f(x) = 0$, ou seja, suas raízes.

Como podemos observar no gráfico da Figura 5.2, existe uma diferença muito grande entre os valores de $N(x)$ para x próximo de uma raiz e os valores para x distante de uma raiz (estes com viabilidades praticamente iguais entre si). Para um k fixo, essa diferença varia de acordo com $\mid f(x) \mid$, determinando picos mais largos ou mais estreitos.

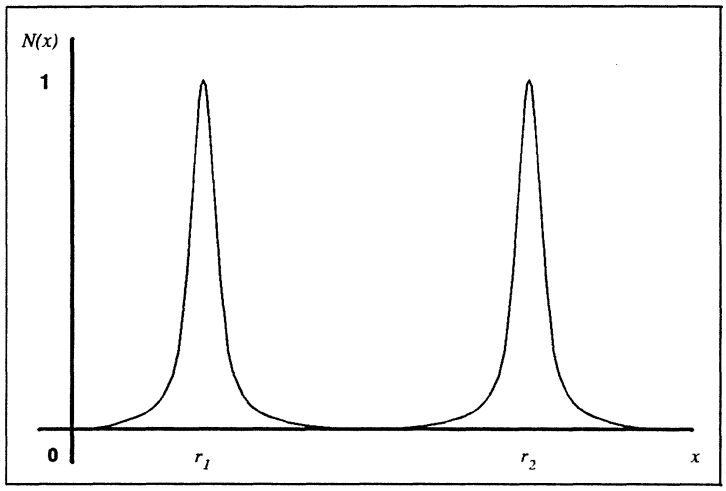


Figura 5.2: Função de avaliação com picos estreitos.

Para o início da busca, picos muito estreitos não são desejáveis pois a probabilidade de haver indivíduos fora dos picos é alta. A convergência desses indivíduos para a região das raízes torna-se, não um processo evolutivo, mas sim um processo aleatório, visto que as probabilidades de seleção para casamentos ficam iguais. Picos muito largos também não são desejáveis pois provocam uma diversidade de graus de viabilidade muito pequena, e recaímos no problema anterior. A solução para isso é ajustar um valor de k para cada função cujas raízes quisermos calcular.

Para uma mesma função, o valor de k determina a largura dos picos em torno das raízes (Figura 5.3). Quanto mais próximo k estiver de 0.0, mais largos serão os picos (se $k = 0.0$, $N(x) = 1.0$ para qualquer x). O valor de k pode ser definido nas primeiras gerações dos AGs. Inicialmente, fazemos $k = 0.0$ e refinamos sucessivamente seu valor em função da média m dos valores máximos de $\mid f(x) \mid$ encontrados a cada geração pelos agentes da aplicação. Se desejarmos que indivíduos para os quais $\mid f(x) \mid > m$ tenham, por exemplo, grau de viabilidade menor que 0.2, poderemos calcular k da seguinte forma:

$$\frac{1}{(1+k)^m} = 0.2 \ ,$$

$$(1+k)^m = 5 \ ,$$

$$k = 5^{\frac{1}{m}} - 1 \ .$$

O propósito do parâmetro k é moldar a função de cálculo de viabilidade para uma busca eficiente, conforme requerido pelos métodos de busca evolutiva conforme discutido em [Harvey 92] e

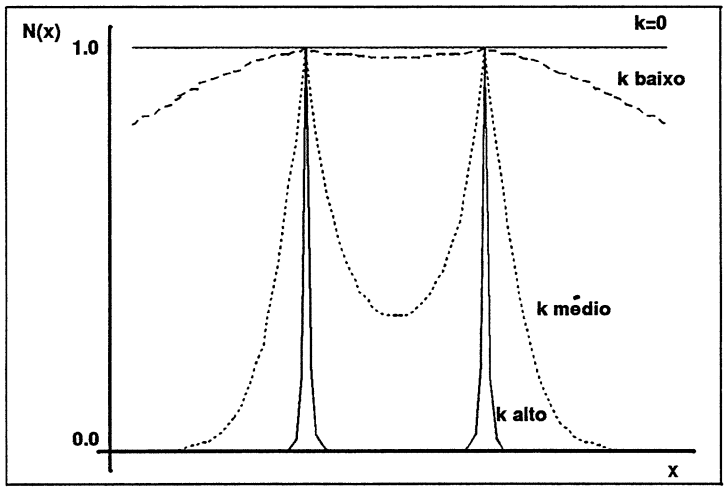


Figura 5.3: Influência do parâmetro k na largura dos picos de $N(x)$.

[Kauffman 89]. Isto é, deve-se moldar o perfil de viabilidade do espaço de busca de forma a se conseguir um grau de correlacionamento adequado entre pontos vizinhos do espaço. k é, portanto, um índice de correlacionamento do perfil de viabilidade do espaço de busca para um problema a resolver. Uma outra forma de obter valores para k seria dispormos de um processo evolutivo diferenciado, dedicado apenas ao refinamento sucessivo desses valores, pela busca a valores máximos da equação.

Uma vez encontrada uma raiz r_1, não é desejável que continue havendo convergência para esse valor e para suas proximidades. Por isso, ao detectarmos uma raiz devemos alterar $N(x)$, incluindo um fator de redução $H(x)$ que assuma valores entre 0.0 e 1.0 para $x \in [r_1 - \Delta_r, r_1 + \Delta_r]$ e 1.0 caso contrário (Figuras 5.4 e 5.5):

$$H(x) = \begin{cases} 0.0 < h < 1.0 & \text{se } x \in [r_1 - \Delta_r, r_1 + \Delta_r] \\ 0.0 & \text{se } x = r_1 \\ 1.0 & \text{outros} \end{cases}$$

5.2.3 Os Polinômios de Hermite

Para implementar a função de redução $H(x)$ utilizaremos uma técnica extraída da Computação Gráfica.

Quando dispomos de dois pontos P_1 e P_2 no espaço e queremos ajustar uma curva fazendo-a passar por esses pontos, utilizamos polinômios cúbicos de interpolação. Esses polinômios são da forma $Q(t) = [x(t)\ y(t)\ z(t)]$, parametrizados por t, $0 \leq t \leq 1$. Existem vários tipos de interpolações que utilizam polinômios desse tipo. Uma delas é a técnica de Hermite, que utiliza dois vetores V_1 e V_2 tangentes à curva nos pontos P_1 e P_2, respectivamente (Figura 5.6). Quando $t = 0$ a curva passa por P_1 e quando $t = 1$ a curva passa por P_2. A formulação para este tipo de polinômio é:

$$Q(t) = \left[\begin{array}{cccc} x(t) & y(t) & z(t) \end{array} \right] = T.M_H.G_H \ , onde$$

$$T = \left[\begin{array}{cccc} t^3 & t^2 & t & 1 \end{array} \right] ,$$

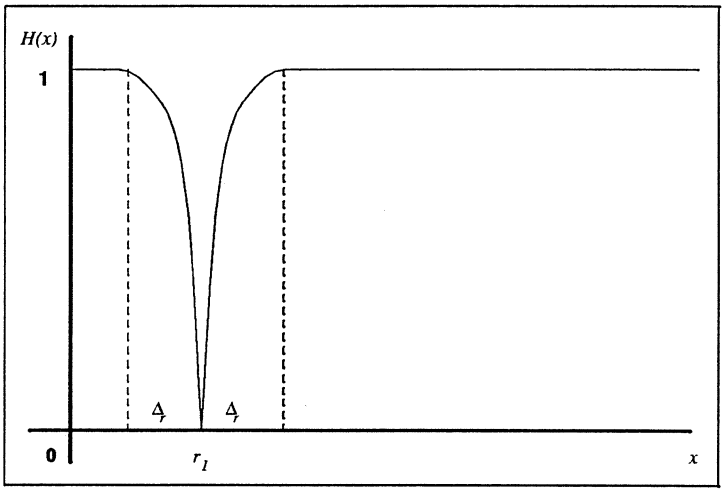


Figura 5.4: Função de compensação após uma raiz encontrada.

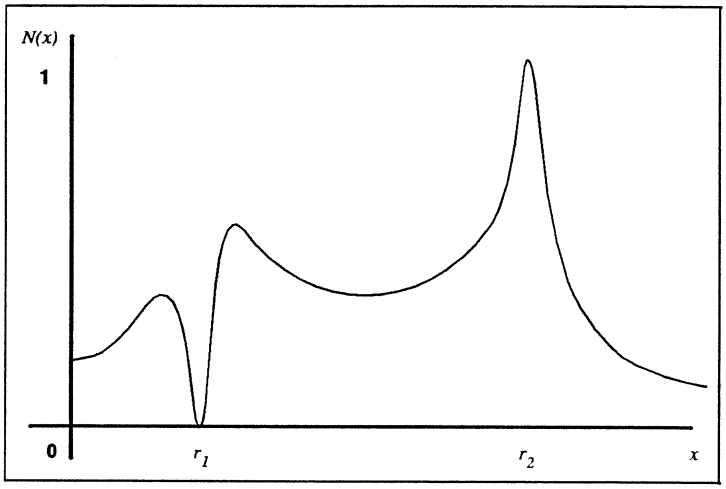


Figura 5.5: Função de avaliação modificada após uma raiz encontrada.

$$M_H = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix},$$

$$G_H = \begin{bmatrix} P_1 \\ P_2 \\ V_1 \\ V_2 \end{bmatrix}.$$

O cálculo mostrado acima é dirigido a curvas no espaço. O nosso caso se restringe a uma curva no

plano e, devido a isso, apenas $y(t)$ nos interessa. Não mostraremos os detalhes de como se chegou às relações acima. Em [Foley et al. 90] essa teoria está descrita em maiores detalhes.

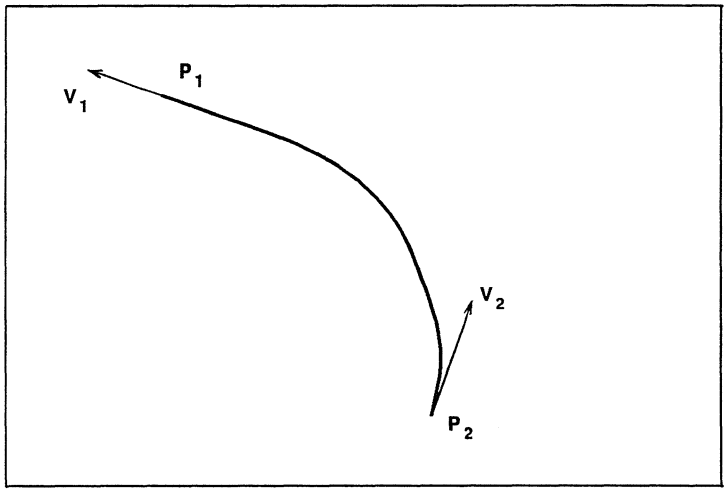


Figura 5.6: Curva de Hermite definida pelos pontos P_1 e P_2 e pelos vetores V_1 e V_2.

Para que um valor de x próximo a uma raiz r já encontrada tenha sua viabilidade reduzida, multiplicamos $N(x)$ por $H(x)$, que é o valor da curva de Hermite calculada para os pontos $P_1 = (r - \Delta_r, 1)$, $P_2 = (r, 0)$, e os vetores tangentes $V_1 = (10, 0)$, horizontal para a direita e $V_2 = (0, 10)$, vertical para baixo (seus valores não precisam ser alterados). Fazemos isso da seguinte forma:

$$t = \frac{x - (r - \Delta_r)}{\Delta_r}$$

$$H(x) = y(t)$$

Assim o valor do polinômio de Hermite $y(t)$ é calculado para os pontos $r - \Delta_r \leq x \leq r$. Os pontos $r \leq x \leq r + \Delta_r$ são calculados pelo seu simétrico em relação a r.

5.2.4 Pontos Sub-Ótimos

Como mencionamos anteriormente, um AG não necessariamente atinge os pontos ótimos em uma busca. Entretanto, no caso de busca de raízes de uma equação, isso se faz necessário. Consideremos os cromossomos C_1 e C_2, vizinhos em relação ao genótipo, e x_1 e x_2, seus fenótipos. Pode acontecer de existir uma raiz r tal que $x_1 < r < x_2$, ou seja, não existir um cromossomo que codifique a raiz. Para se descobrir r, toda vez que ocorrer um indivíduo com viabilidade suficientemente alta para que haja a possibilidade de seu fenótipo estar muito próximo de uma raiz, devemos fazer uma inspeção em torno do seu fenótipo. Para isso determinamos um limiar de inspeção l, $0.0 < l < 1.0$ (Figura 5.7). Quando um indivíduo possuir um grau de viabilidade superior a l, faremos uma inspeção na sua vizinhança no domínio do fenótipo.

5.2.5 Comunicação Entre os AGs

Para implementar a comunicação entre os agentes em uma aplicação co-evolutiva como a descrita acima podemos obedecer a vários critérios. Entre eles podemos citar os seguintes:

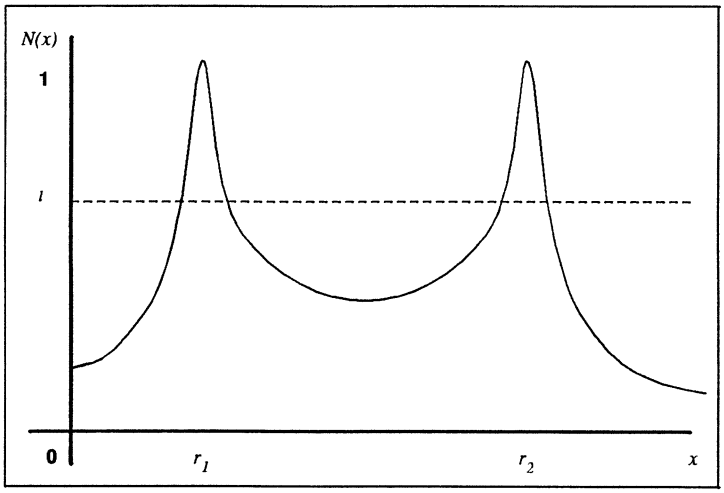


Figura 5.7: Limiar de inspeção e pontos sub-ótimos.

Quanto ao conteúdo das mensagens:

- Raízes encontradas: cada vez que um agente encontrar uma raiz deve comunicar aos outros agentes para que estes alterem suas funções de avaliação;
- Melhores indivíduos encontrados: os agentes trocam entre si os seus indivíduos de melhor grau de viabilidade;
- Novos valores de k: cada fez que for encontrado um valor de k mais apropriado, deve ser disseminado entre os agentes;

Quanto à disseminação:

- Total: um agente envia suas mensagens para todos os outros agentes;
- Seqüencial: um agente envia suas mensagens ao próximo agente;
- Aleatória: um agente escolhe aleatoriamente um dos agentes para os quais enviará suas mensagens.

Quanto à freqüência:

- Sempre: a cada geração o agente envia suas mensagens aos outros agentes;
- Intervalos: a cada determinado número de gerações um agente envia suas mensagens aos outros;
- Probabilidade: o envio de mensagems de um agente está associado a uma probabilidade de ocorrência;

Em especial, com relação à comunicação de indivíduos, podemos ter ainda os sequintes critérios:

Quanto à escolha do indivíduo:

- Elitista: um agente escolhe os n melhores indivíduos da sua atual geração para enviá-los aos outros agentes. O valor de n pode ser fixo ou não;
- Probabilístico: um agente escolhe os n indivíduos da sua atual geração para enviá-los aos outros agentes seguindo o mesmo critério de escolha para formação dos pares reprodutivos.

Quanto à recepção dos indivíduos:

- Substituição: O agente substitui os n piores indivíduos da sua atual geração pelos n indivíduos que receber;
- Competição: O agente complementa sua população com os n indivíduos que receber para que possam competir com os que já possui (esta opção necessita de aviso de aumento de população ao servidor);
- Análise: O agente analisa os indivíduos que recebeu e decide quais irá considerar e quais serão descartados.

5.3 Avaliação da Busca de Raízes de uma Equação Determinada

Usaremos a aplicação co-evolutiva descrita acima para buscar as raízes da função

$$f(x) = x^3 - 2857.x^2 + 1785962.x - 151026200$$

cujas raízes são: 100, 754 e 2003, no intervalo [0,4096]. Os critérios utilizados foram os seguintes:

- A aplicação foi executada várias vezes para 1,2,3,4,5 e 6 agentes;
- A comparação do desempenho de execução para cada número de agentes é feita pela média de gerações, média de tempo gasto e média de indivíduos processados;
- A aplicação é homogênea, isto é, os agentes são iguais entre si e trabalham sob os mesmos parâmetros;
- A aplicação foi executada inteiramente na mesma plataforma do servidor, isto é, o tempo de rede não está sendo considerado.
- Os agentes comunicam entre si as raízes encontradas, bem como o melhor indivíduo de cada geração (elitista). A comunicação de indivíduos ocorre sob uma probabilidade de 70%;
- Cada agente envia suas mensagens a todos os outros agentes;
- Ao receber um novo indivíduo, o agente substitui o pior indivíduo de sua geração atual pelo novo;
- A população para cada agente é composta de 20 indivíduos escolhidos aleatoriamente;
- O valor de k foi pré-determinado por inspeção e fixado em 0.007;
- Os valores de $\Delta_r = 100.0$ e de $l = 0.5$ foram escolhidos arbitrariamente;
- O tipo de crossover é o de um ponto;
- Seleção e mutação são realizadas da maneira como foram descritas no capítulo 3;

O desempenho dessa aplicação é comparado ao desempenho da execução de um AG simples para o mesmo problema, sob os mesmos parâmetros, para populações de tamanho 10, 20, 30, 40, 50, 60, 70 e 80. A execução de um AG simples para esse problema é ilustrada pelos gráficos das Figuras 5.8, 5.9 e 5.10. Como podemos observar, o número de gerações necessário para busca das três raízes decresce significativamente com o aumento do tamanho da população utilizada. Esse comportamento é esperado, uma vez que populações maiores permitem uma maior probabilidade de aparecimento de bons indivíduos logo nas primeiras gerações. O número de indivíduos processados também apresenta comportamento decrescente, uma vez que é função do tamanho da população e do número de gerações. O tempo gasto para a busca das raízes tem um comportamento decrescente até populações de 40 a 50 indivíduos, tornando a crescer para populações maiores, o que significa que a partir destes tamanhos de população, a diminuição do número de gerações para solução do problema começa a não compensar o tempo gasto, pois o tempo envolvido na manipulação de cada geração será cada vez maior.

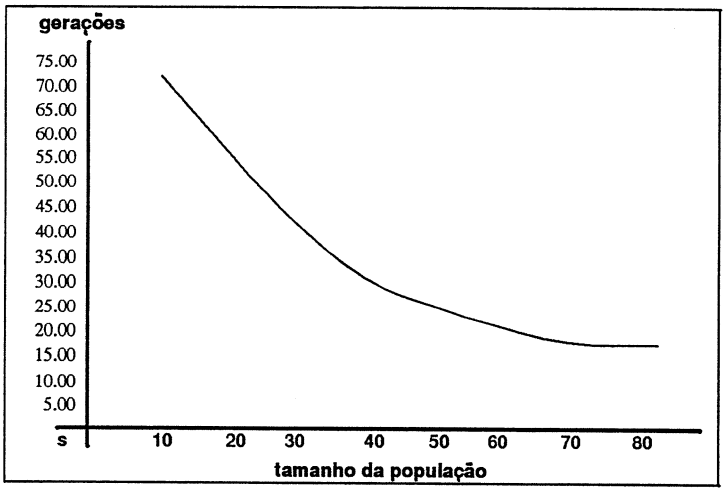


Figura 5.8: Gráfico do número de gerações em função do tamanho populacional para um AG simples.

Na execução da aplicação co-evolutiva, pelos gráficos mostrados nas Figuras 5.11, 5.12 e 5.13, podemos observar uma redução considerável para as três medidas tomadas. Uma vez que cada agente trabalha com uma população de 20 indivíduos devemos comparar a aplicação co-evolutiva e o AG simples em termos da dimensão populacional equivalente, por exemplo, uma execução co-evolutiva com 2 agentes deve ser comparada com a execução do AG simples para uma população de 40 indivíduos.

A vantagem da utilização de co-evolução para a resolução deste problema sobre o AG simples é nítida. Podemos ver que para 4 agentes temos algo em torno de 6 gerações e $0.8s$ para uma população total de 80 indivíduos. Para esse tamanho populacional o desempenho de um AG simples fica em 20 gerações e um pouco mais de $1.2s$.

Convém lembrar que esta aplicação co-evolutiva foi realizada inteiramente na mesma máquina onde reside o processo servidor.

É evidente que uma função polinomial como a apresentada é simples e que não seria necessário um esquema desse porte para calcular suas raízes. A intenção aqui foi a de ilustrar o ganho de desempenho de uma aplicação co-evolutiva sobre um AG simples.

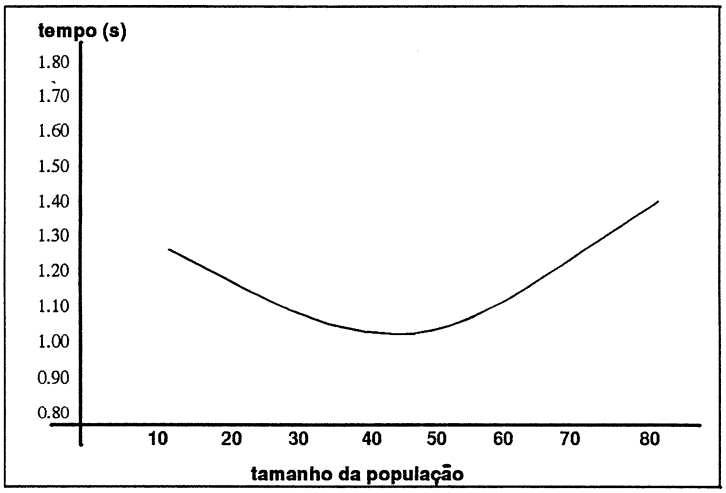


Figura 5.9: Gráfico do tempo gasto em função do tamanho populacional para um AG simples.

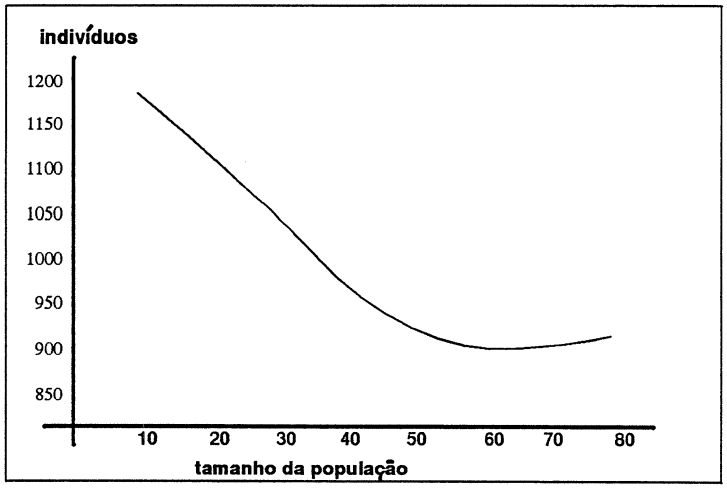


Figura 5.10: Gráfico do número de indivíduos processados em função do tamanho populacional para um AG simples.

5.4 Classificação de Aplicações Co-Evolutivas

Em [Hillis 91] é feita a apresentação de uma aplicação co-evolutiva para o problema de Redes de Ordenação Mínimas, que consiste em se determinar o menor número de comparações e trocas entre elementos de um conjunto, em determinada ordem, de forma que resulte em um conjunto ordenado.

Para este problema foram projetados dois processos co-evolutivos. Um processo que evolui esquemas de comparações e trocas, e outro que evolui casos de testes para esses esquemas. A evolução no primeiro processo é motivada pelo sucesso em resolver os casos de testes gerados pelo segundo processo, ao passo que a evolução do segundo processo se dá pela capacidade de gerar casos de teste que invalidem o maior número de esquemas gerados no primeiro processo.

Para se ter uma medida do resultado atingido por essa aplicação, basta dizer que sob métodos

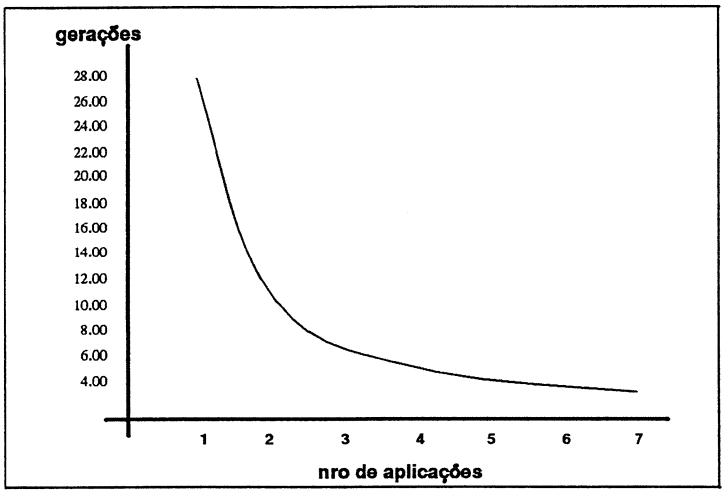


Figura 5.11: Gráfico do número de gerações em função do número de agentes em uma aplicação co-evolutiva.

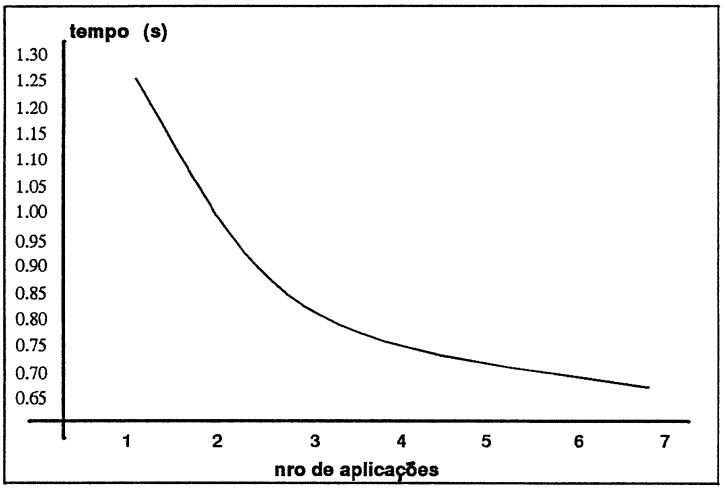


Figura 5.12: Gráfico do tempo gasto em função do número de agentes em uma aplicação co-evolutiva.

convencionais, para um conjunto de desesseis elementos, segundo o autor, em 1962, Bose e Nelson resolveram com 65 trocas; em 1964, Knuth atingiu 63; e, em 1969, Green obteve 60 trocas. A aplicação apresentada obteve, sob esse esquema, 61 comparações, uma a mais que o número obtido por Green.

O tipo de relacionamento entre esses dois processos evolutivos foi batizado pelo autor como "parasitismo". Pensando dessa forma, e fazendo também uma analogia com as relações entre seres vivos na Natureza, poderíamos classificar o relacionamento entre processos co-evolutivos da seguinte forma:

- Mutualista: quando os processos evolutivos contribuem um para a evolução dos outros e vice-versa. O relacionamento entre os processos na aplicação apresentada para resolução da equação pode ser enquadrado nesse tipo.
- Comensalista: quando um processo se beneficia do processo evolutivos de outro, sem no entanto

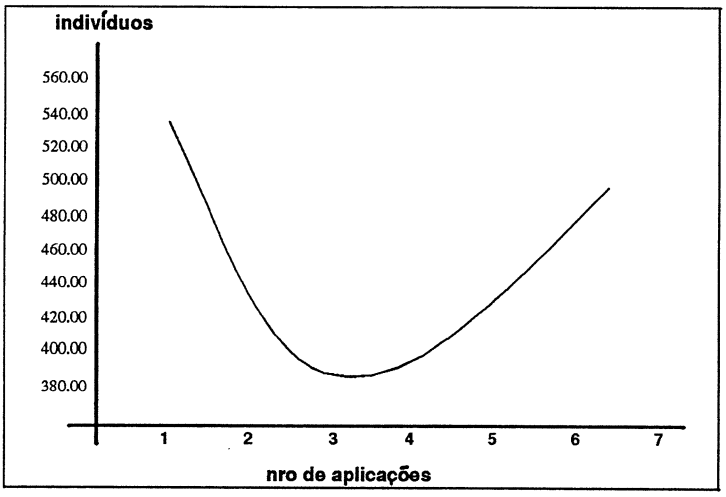


Figura 5.13: Gráfico do número de indivíduos processados em função do número de agentes em uma aplicação co-evolutiva.

prejudicá-lo ou beneficiá-lo. Na aplicação para a equação, se tivéssemos um processo para evolução do valor de k, o relacionamento entre esse processo e os outros seria desse tipo. Apesar dos processos normais se valerem do valor de k gerado evolutivamente por esse processo especial, esse mesmo processo não é afetado, pois usa informação própria.

- Competitivo: quando um processo evolutivo existe para prejudicar o processo evolutivo de outros processos, como em [Hillis 91]. É evidente que esse prejuízo é em benefício de um processo evolutivo global principal, onde essa competição é um fator a mais de estímulo à evolução.

5.5 Comentários

A decisão de se utilizar uma aplicação co-evolutiva em rede ou na mesma máquina, ou um AG simples, ou paralelo, ou até mesmo outro método para a resolução de um problema deve depender de aspectos tais como:

- Complexidade de codificação e avaliação: há muitos problemas simples cuja codificação do espaço de busca pode vir a ser intrincada demais e que não compense o uso de AGs;
- Complexidade do problema a ser resolvido: podem existir casos onde esse esforço seja compensado pelo desempenho do AG, co-evolutivo ou não.
- Desempenho de rede e distância entre máquinas: podem ser um fator importante na escolha entre resolver um problema pelo modelo apresentado ou não.

Capítulo 6

Conclusão

6.1 Plataformas de Operação

Por suas características, o modelo proposto está voltado para plataformas multi-usuário. O processo servidor deve residir, preferencialmente, em máquina de grande poder computacional. Entretanto, aplicações remotas, conforme a necessidade, não precisam ser necessariamente implementadas em ambientes multi-usuário. O modelo de servidor e de aplicações ilustrado neste trabalho foi todo implementado em linguagem C, sob ambiente Unix em estações de trabalho. A comunicação entre os processos clientes e o servidor foi implementada por "sockets" em protocolo TCP/IP. Esse ambiente, pela forma que foi implementado, permite a utilização da Internet para comunicação entre um servidor e clientes que se situem em cidades - ou até mesmo países - diferentes. Entretanto, alguns problemas operacionais de configuração no TCP/IP impediram testes confiáveis entre máquinas diferentes.

6.2 Comparações com Outras Implementações Paralelas

Há várias outras formas de implementação de AGs paralelos. Uma característica comum entre elas é a divisão da população em subpopulações, uma para cada processo paralelo. [?] faz o cálculo de várias estimativas de complexidade para o mapeamento de AGs para uma máquina paralela. Sua conclusão foi que o cálculo das estatísticas populacionais, para seleção e escalonamento, seria o principal gargalo em implementações desse tipo. [Grefenstette 81] examinou algumas implementações paralelas de AGs e distiguiu quatro classes:

- Mestre-Escravo síncrono
- Mestre-Escravo semi-síncrono
- Concorrente, assíncrono e distribuído
- Rede

Numa implementação Mestre-Escravo (Figura 6.1), um único processo mestre coordena k processos escravos. O mestre controla a seleção, reprodução e operadores genéticos. Os escravos fazem simplesmente a avaliação. Essa aplicação, segundo Goldberg, apresenta dois problemas principais. Primeiro, por ser síncrono, pode-se perder muito tempo se existir muita variação no tempo gasto para avaliação em cada processo. Segundo, é a dependência do estado do processo mestre. Se este for interrompido o sistema fica bloqueado.

A implementação concorrente assíncrona utiliza k processos idênticos realizando operações genéticas e avaliação independentemente uns dos outros e acessando uma área de memória compartilhada (Figura 6.2).

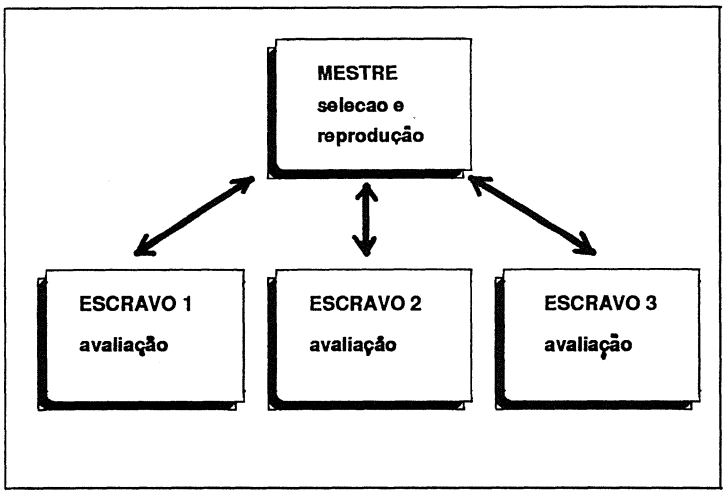


Figura 6.1: Implementação de um AG paralelo síncrono sob filosofia mestre-escravo.

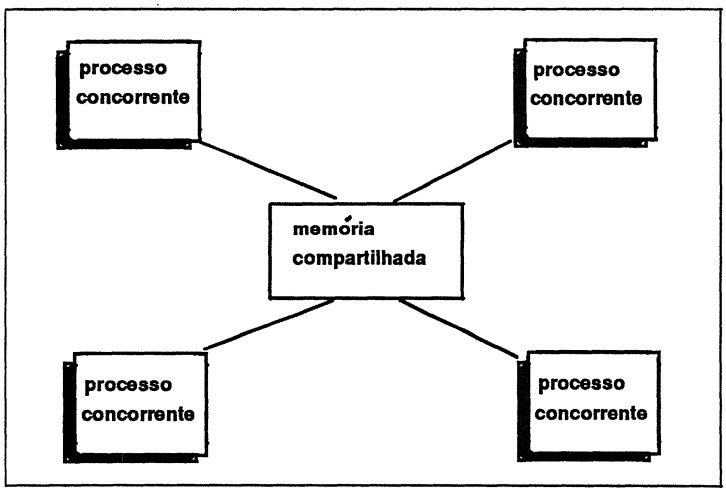


Figura 6.2: Esquema de um AG paralelo concorrente e assíncrono.

Na implementação em rede (Figura 6.3) os processos possuem cada um sua memória própria e são executados normalmente, com a exceção de que os melhores indivíduos encontrados são disseminados pela rede para as outras subpopulações. Não ocorre nesse tipo de implementação o problema de dependência de um único processo.

[Maruyama et al. 93] propõem uma implementação paralela, assíncrona para AGs da classe celular, onde cada processo controla apenas um indivíduo corrente e há um "buffer" para armazenar os outros indivíduos. Como se pode ver, as implementações paralelas relacionadas propõem uma divisão da população em subpopulações, cada uma a ser controlada por um processo paralelo [Pettey and Leuze 89] e [Shonkwiler 89]. As restrições e as tentativas de aperfeiçoamento desses modelos estão centralizadas na reunificação das estatísticas das subpopulações para levantamento do estado corrente da busca por soluções do problema. Daí a incoveniência de modelos síncronos.

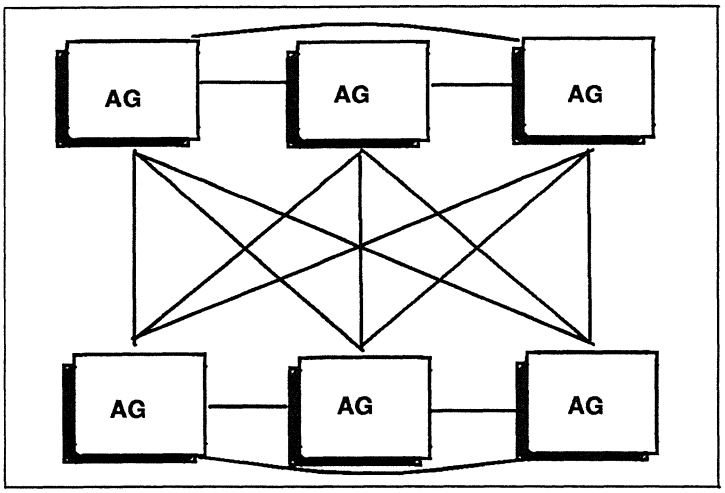


Figura 6.3: Esquema de um AG paralelo em rede.

O que distingue o modelo proposto dos modelos apresentados acima é o enfoque totalmente inverso da prestação de serviços. Nos modelos relacionados, notadamente nos esquemas Mestre-Escravo, o mestre manipula as subpopulações e os escravos as avaliam. Assim, a avaliação é o serviço prestado e não a manipulação populacional. O resultado da execução de um modelo como estes para um determinado problema é fornecido pelo mestre após avaliação unificada das subpopulações. Como podemos observar, nessas implementações o paralelismo é usado para aprimorar a busca de soluções de um único problema.

No modelo proposto ocorre exatamente o contrário. O servidor atua como se fosse um único "super-escravo", prestando seus serviços para um ou mais clientes que seriam seus mestres. Como o servidor não precisa conhecer os detalhes de codificação e avaliação das populações que manipula, criamos condições de implementar algo não levantado nos modelos analisados, que é a possibilidade de aplicações distintas fazerem uso do servidor de forma compartilhada e independente, isto é, usar o paralelismo para possibilitar o compartilhamento das operações populacionais por diversas aplicações possivelmente distintas. O modelo também não impossibilita as implementações nas quais se basearam os modelos analisados, com possível exceção dos modelos celulares, embora a recíproca não seja verdadeira (Figura 6.4).

A dependência ao processo servidor, citada por [Goldberg 89], é um problema operacional que não invalida o modelo, pois pode ser resolvido com a utilização de servidores "back-up", por exemplo.

Em relação a aplicações co-evolutivas a principal diferença com os modelos paralelos apresentados é que em co-evolução os AGs são agentes autônomos, não dependendo necessariamente uns dos outros para que se chegue a um resultado. Obviamente esse resultado chegará mais cedo tanto quanto melhor for a forma de comunicação entre eles. Nos esquemas paralelos citados acima, um AG depende do outro, e o resultado final surge apenas a partir de uma avaliação global dos estados dos AGs. Isso implica que a forma de implementação e o desempenho dos AGs - a cada geração - em um esquema paralelo convencional, não afetam o número de gerações para resolução de um problema; afetam apenas o tempo gasto. Para aplicacões co-evolutivas, esses aspectos afetam o número de gerações, pois o comprometimento do tempo de processamento das comunicações faz com que o processo evolutivo de cada agente tenha uma convergência mais lenta.

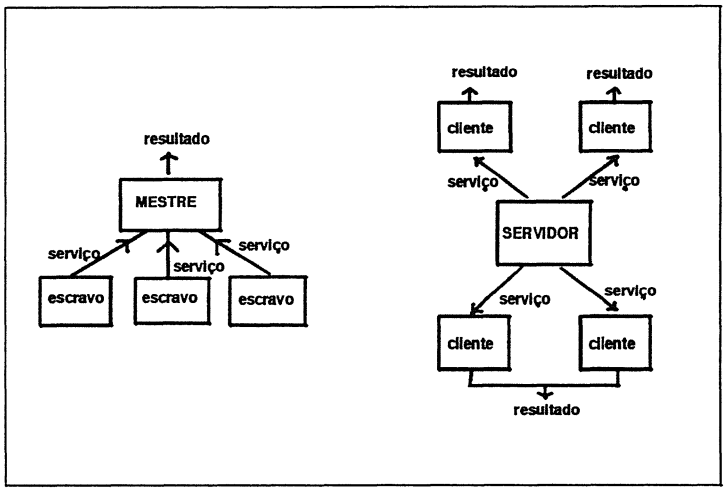


Figura 6.4: Comparação entre a arquitetura mestre-escravo e a arquitetura cliente-servidor.

6.3 O Futuro do Modelo

6.3.1 Facilidades a Serem Incluídas

O modelo proposto, tal como está implementado, é apenas um protótipo. Entre as facilidades que podem ser incluídas no processo servidor, sem comprometer a sua filosofia de independência, podemos citar as seguintes:

- Escolha do tipo de crossover a ser utilizado.
- Escolha do tipo de seleção a ser utilizado.
- Alteração dos parâmetros da aplicação em tempo de execução.
- Codificação em mais de um cromossomo.
- Uso de cromossomos diplóides e noção de dominância e recessividade, ou seja, mais de um cromossomo para uma mesma característica.

A escolha do tipo de crossover utilizado é uma opção que permite à aplicação determinar o poder exploratório do AG. A forma de se realizar o processo seletivo também pode ser selecionável. Algumas formas de seleção, conforme o caso, podem ser mais adequadas do que outras. Essas duas facilidades podem ser facilmente incluídas pois dependem apenas da inclusão de um campo a mais para cada uma no protocolo de pedido de cadastramento mostrado na Figura 4.6.

A alteração de parâmetros da aplicação durante o tempo de execução dá à aplicação o poder e a flexibilidade de, conforme uma análise da população, intervir e alterar o contexto de busca de soluções se assim lhe for conveniente. Por exemplo, se em algum instante a população apresentar muito pouca diversidade em relação a seu tamanho, a aplicação pode requisitar uma mudança no tipo de crossover, ou elevar a probabilidade de ocorrência de mutação, a propósito de atingir outros pontos dentro do espaço de busca. A implementação dessa facilidade necessita a criação de mais um protocolo de comunicação cliente-servidor nos moldes do pedido de cadastro de aplicação. O servidor, ao receber esse pedido, altera na área reservada aos parâmetros da aplicação, os ítens requisitados.

Frequentemente nos deparamos com problemas cuja solução (ou soluções) constitui um conjunto de atributos. Normalmente, para codificar um conjunto de atributos em um cromossomo, dividimos este em seções, cada uma referente a um atributo. Porém, dependendo do tipo de crossover utilizado, pode haver, a cada cruzamento, recombinação em apenas algumas dessas seções. Para que o crossover possa ser realizado - independente do seu tipo - sobre todas as codificações de atributos, e também para dar uma estrutura mais adequada ao conjunto de atributos codificados, podemos criar indivíduos com mais de um cromossomo, um para cada atributo codificado. Quando ocorrer a reprodução, cada cromossomo de um indivíduo será cruzado com o respectivo cromossomo do outro indivíduo.

Para o AG, o conceito de dominância e recessividade pode ser útil exatamente nos casos em que as características do problema a ser resolvido não se mantenham constantes durante o tempo de execução. Assim, uma codificação já explorada e descartada anteriormente poderia permanecer latente e ser posteriormente reaproveitada positivamente sob uma nova condição imposta por mudanças nas características do problema. Assim o AG trabalharia com dois cromossomos para cada característica, definindo a dominância ou recessividade de suas posições em função de algum critério estabelecido. Esse conceito pode ser expandido para mais de dois cromossomos por característica se for conveniente. Embora mais complexo, esse esquema também não interfere nas operações de seleção, crossover e mutação.

6.3.2 Comentário Final

A versatilidade do modelo proposto permite que ele seja um infra-estrutura adequada para o estudo e desenvolvimento de novas implementações paralelas com a possível existência de dois ou mais processos servidores dividindo entre si os clientes.

Evidentemente, os problemas-exemplo mostrados neste trabalho podem ser mais eficientemente resolvidos por métodos tradicionais. A intenção foi ilustrar os esquemas e critérios envolvidos em AGs e Co-Evolução em relação a problemas de fácil assimilação.

A eficácia das técnicas apresentadas depende de vários fatores relacionados com a natureza do problema e também com a infra-estrutura disponível. A pesquisa em AGs e aplicações co-evolutivas vem crescendo cada vez mais, motivada principalmente pelo interesse de aplicabilidade a um leque cada vez mais amplo de problemas.

Bibliografia

[Bethke 76] . D. Bethke, "Comparison of Genetic Algorithms and Gradient Based Optimizers on Parallel Processors: Efficiency of Use of Processing Capacity", Technical Report No. 197, Ann Arbor, University of Michingan, Logic of Computer Group, 1976, Apud [Goldberg 89].

[Bukatova and Gulyaev 93] I. L. Bukatova and Y. V. Gulyaev. "From Genetic Algorithms to Evolutionary Computer". In: Forrest, S. editor. *Proceedings of the 5th International Conference on Genetic-Algorithms, ICGA-93*, Morgan Kaufmann, San Mateo CA, 1993.

[Davis and Steenstrup 87] L. Davis and M. Steenstrup. *Genetic Algorithms and Simulated Annealing: an Overview.* Londom: Pitmasn, 1987.

[Deb and Goldberg 89] K. Deb and D. E. Goldberg. "An Investigation of Niche and Species Formation in Genetic Function Optimization". In: Schaffer, J. D. editors. *Proceedings of the 3rd International Conference on Genetic-Algorithms, ICGA 89*, Morgan Kaufmann, San Mateo CA, 1989.

[De Jong 75] K. De Jong. *An Analysis of The Behavior of a Class of Genetic Adaptive Systems.* Doctoral Dissertation, University of Michigan, 1975

[De Jong and Spears 89] K. De Jong and W. Spears. "Using Genetic Algorithms to Solve NP-Complete Problems". In: Schaffer, J. D. editor. "*Proceedings of the 3rd International Conference on Genetic-Algorithms, ICGA-89*, Morgan Kaufmann, San Mateo CA, 1989.

[De Jong and Spears 93] K. De Jong and W. Spears. "On the State of Evolutionary Computation". In: Forrest, S. editor. *Proceedings of the 5th International Conference on Genetic-Algorithms, ICGA-93*, Morgan Kaufmann, San Mateo CA, 1993.

[Fogel et al. 66] L. J. Fogel, A. J. Owens and M. J. Walsh. *Artificial Intelligence Through Simulated Evolution*, New York: Wiley Publish.,1966

[Foley et al. 90] J. D. Foley, A. van Dam, S. Feiner and J. F. Huges. *Computer Graphics: Principle and Practice*, 2nd ed., Addison-Wesley, 1990.

[Goldberg and Richardson 87] D. E. Goldberg and J. Richardson. Genetic Algorithms and their applications: Proceedings of the Second International Conference on Genetic Algorithms, 1-8.

[Goldberg 89] D. E. Goldberg, *Genetic Algorithms in search, optimization, and machine learning*, Addison-Wesley, 1st ed., 1989.

[Gordon et al. 92] V. Gordon, D. Whitley and A. Bohm, "Dataflow Parallelism in Genetic Algorithms", *Parallel Problem Solution from Nature 2*, North Holland, 1989.

[Grefenstette 81] J. J. Grefenstette, *Parallel Adaptive Algorithms for Function Optimization*, Technical Report No. CS-81-19, Nashville, Vanderbilt University, Computer Science Department, 1981.

[Grobman 62] A. Grobman, *BSCS - Biological Sciences Curriculum Study*, American Institute of Biological Sciences. 8th ed., 1980.

[Hamaifar et al. 93] A. Hamaifar, S. Guan and G. Liepins. "A New Approach on the Traveling Salesman Problem by Genetic Algorithms". In: Forrest, S. editor. "*Proceedings of the 5th International Conference on Genetic-Algorithms, ICGA-93*, Morgan Kaufmann, San Mateo CA, 1993.

[Harvey 92] Inman Harvey, "Species Adaptation Genetic Algorithms: The Basis for a Continuing SAGA". In: F. J. Varela and P. Bourgine, editors. *Towards a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life*, MIT Press/Bradford Books, Cambridge, MA, 346–354, 1992.

[Hillis 91] W. D. Hillis, "Co-Evolving Parasites Improve Simulated Evolution as an Optimization Parameter". In: C. G. Langton, J. D. Farmer, S. Rasmussen and C. Taylor, editors. *Artificial Life II*, Addison-Wesley, 311–324, 1992.

[Holland 75] J. Holland, *Adaptation in natural and artificial systems*, Ann Arbor: The University of Michigan Press, 1975.

[Holland 92] J. Holland, "Genetic Algorithms", *Scientific American*, pg 44–50, July 1992.

[Husbands et al. 91] Philip Husbands and Frank Mill, "Simulated Co-Evolution as the Mechanism for Emergent Planning and Scheduling". In: R. K. Belew and L. B. Booker, editors. *Proceedings of the Fourth Intl. Conf. on Genetic Algorithms, ICGA-91*, Morgan Kaufmann, San Mateo CA, 264–270, 1991.

[Jarman 70] C. Jarman. *Evolution of Life*, Hamlyn Publish. Gr. Lim., 1970.

[Kauffman 89] Stuart Kauffman, "Adaptation on Rugged Fitness Landscapes", In: Daniel L. Stein, editor. *Lectures in the Sciences of Complexity*, Addison Wesley: Santa Fe Institute Studies in the Sciences of Complexity, 527–618, 1989.

[Maruyama et al. 93] T. Maruyama, T. Hirose and A. Konagaya, "A Fine-Grained Parallel Genetic Algorithm for Distributed Parallel Systems". In: Forrest, S. editor. *Proceedings of the 5th International Conference on Genetic-Algorithms, ICGA-93*, San Mateo CA, 1993.

[Pettey and Leuze 89] C. C. Pettey and M. R. Leuze. "A theoretical Investigation of a Parallel Genetic Algorithm". In: Schaffer, J. D. editor. *Proceedings of the 3rd International Conference on Genetic-Algorithms, ICGA-89*, San Mateo CA, 1989.

[Schaffer et al. 89] J. D. Schaffer, L. J. Eshelman and R. A. Caruana, "Biases in the Crossover Landscape". In: Schaffer, J. D. editor. *Proceedings of the 3rd International Conference on Genetic-Algorithms, ICGA-89*, Morgan Kaufmann, San Mateo CA, 1989.

[Shonkwiler 89] R. Shonkwiler. "Parallel Genetic Algorithms". In: Schaffer, J. D. editor. *Proceedings of the 3rd International Conference on Genetic-Algorithms, ICGA-89*, Morgan Kaufmann, San Mateo CA, 1989.

[Storer and Usinger 78] T. I. Storer; R. L. Usinger *General Zoology*, McGraw-Hill, 4th ed., 1978.

[Syswerda 89] G. Syswerda. "Uniform Crossover in Genetic Algorithms". In: Schaffer, J. D. editor. *Proceedings of the 3rd International Conference on Genetic-Algorithms, ICGA-89*, San Mateo CA, 1989.

[Weisskopf 63] V. F. Weisskopf. *Knowledge and Wonder*, Anchor Books, 1st ed., 1963.

[Whitley 89] D. Whitley, T. Starkweather and D. Fuquay. "Scheduling Problems and Traveling Salesman: The Genetic Edge Recombination Operator". In: Schaffer, J. D. editor. *Proceedings of the 3rd International Conference on Genetic-Algorithms, ICGA-89*, Morgan Kaufmann, San Mateo CA, 1989.

[Whitley and Gordon 93] D. Whitley and V. S. Gordom, "Serial and Parallel Genetic Algorithms as Function Optimizers". In: Forrest, S. editor. *Proceedings of the 5th International Conference on Genetic-Algorithms, ICGA-93*, Morgan Kaufmann, San Mateo CA, 1993.

Apêndice A

Relatório da execução do Algoritmo Genético para o problema do dispositivo "caixa-preta".