Ministério da Agricultura e do Abastecimento

ISSN 1518-3602

UM ESTUDO SOBRE ALGORITMOS GENÉTICOS

Jaime Hidehiko Tsuruta
Marcelo Gonçalves Narciso

Embrapa

Informática Agropecuária

Ministério da Agricultura e do Abastecimento

ISSN 1518-3602

UM ESTUDO SOBRE ALGORITMOS GENÉTICOS

Jaime Hidehiko Tsuruta
Marcelo Gonçalves Narciso

Embrapa
Informática Agropecuária

ISSN 1518-3602

Empresa Brasileira de Pesquisa Agropecuária
Embrapa Informática Agropecuária
Ministério da Agricultura e do Abastecimento

UM ESTUDO SOBRE ALGORITMOS GENÉTICOS

Jaime Hidehiko Tsuruta
Marcelo Gonçalves Narciso

Campinas, SP
2000

SUMÁRIO

UM ESTUDO SOBRE ALGORITMOS GENÉTICOS

Jaime Hidehiko Tsuruta[1], Marcelo Gonçalves Narciso[1]

RESUMO - Este trabalho apresenta uma descrição da metaheurística Algoritmos Genéticos. Como, onde e quando aplicar estes algoritmos são o alvo deste trabalho, assim como as circunstâncias em que não se deve aplicá-los.

Termos para indexação: Algoritmo genético; Otimização.

A STUDY ON GENETIC ALGORITHMS

ABSTRACT - This paper presents a description of the metaheuristic called Genetic Algorithms. How, where and when the user should apply these algorithms are the aim of this work, as well as the circumstances when the user should not apply them.

Index terms: Genetic algorithm; Optimization.

[1] Pesquisadores da Embrapa Informática Agropecuária, Caixa Postal 6041, Barão Geraldo - 13083-970 - Campinas, SP.

UM ESTUDO SOBRE ALGORITMOS GENÉTICOS

1. Introdução

Em Otimização Combinatória estudam-se problemas que se caracterizam pelo número finito de soluções possíveis. Embora, em princípio, a solução ótima possa ser obtida através de uma simples enumeração, na prática, freqüentemente isto se torna inviável devido ao número extremamente alto de soluções viáveis. Assim, estudando as propriedades estruturais do problema, métodos heurísticos têm sido apresentados pela comunidade científica para obter soluções exatas ou aproximadas.

Os métodos heurísticos têm a característica de obterem boas soluções (e em muitos casos soluções ótimas) em intervalos de tempo reduzidos e compatíveis com as exigências de situações reais. Isto tem contribuído para o grande interesse pelo assunto, sendo que para diferentes problemas combinatoriais, os melhores resultados da literatura foram obtidos através destes métodos.

Algumas heurísticas, denominadas metaheurísticas, podem ser usadas na resolução de diversos problemas de otimização combinatória através de uma representação adequada e a adaptação de alguns parâmetros para cada problema específico.

Metaheurísticas são heurísticas de busca no espaço de soluções, e podem ser divididas em duas classes. A primeira compreende os métodos que exploram uma vizinhança a cada iteração, alterando tanto a vizinhança quanto a forma de explorá-la de acordo com suas estratégias e escolhendo apenas um elemento dessa vizinhança nessa iteração. Esse tipo de varredura do espaço de soluções gera um caminho ou trajetória de soluções obtida pela transição de uma solução para outra de acordo com os movimentos permitidos pela metaheurística. Dentro dessa classe de métodos podem ser citados a *Tabu Search* (Glover, 1989; Glover, 1990) e *Simulated Annealing* (Kirkpatrick et al., 1983; Cerny, 1985).

Em contraposição a esses métodos, existem aquelas metaheurísticas que exploram uma população de soluções a cada iteração. Esses métodos constituem a segunda classe de metaheurísticas e suas estratégias de busca são capazes de explorar várias regiões do espaço de soluções de cada vez. Dessa forma, ao longo das iterações não se constrói uma trajetória única de busca, pois novas soluções são obtidas através de combinação de soluções anteriores. Nessa classe estão os *Algoritmos Genéticos* (Holland,1995; Goldberg, 1989).

Os algoritmos genéticos são um dos representantes dos algoritmos evolutivos, que têm inspiração baseada na evolução natural dos seres vivos.

Os algoritmos genéticos[2] (GAs) foram inicialmente propostos pelo Prof. John Holland (1995) da Universidade de Michigan, mas somente a partir dos anos 80 é que realmente

[2] Do inglês Genetic Algorithms (GAs), doravante estará abreviado conforme é conhecido e muito difundido nos textos científicos publicados.

começaram a se popularizar. A idéia inicial foi tentar imitar algumas etapas do processo de evolução natural das espécies, incorporando-as a um algoritmo computacional.

Basicamente, o ponto de referência é gerar, a partir de uma população inicial, novas populações de cromossomos com propriedades genéticas superiores às de seus antecedentes. Esta idéia foi então associada a soluções de um problema onde, a partir de um conjunto de soluções atuais, são geradas novas soluções superiores às antecedentes, sob algum critério preestabelecido.

Encontra-se na literatura vasto material sobre as vantagens dos algoritmos genéticos em relação a outros métodos de otimização. Porém, quando se lê os artigos de pesquisadores usando outros métodos de otimização, as vantagens não parecem ser tão grandes. O tema central deste trabalho é mostrar quando usar os algoritmos genéticos para um problema de otimização de tal forma que seja vantajoso em termos de tempo e qualidade da solução. Assim, este trabalho apresenta uma proposta de caracterização dos tipos de aplicações em que os GAs podem ser eficientes, e aqueles em que tal escolha pode não ser tão boa. É importante notar que mesmo para aplicações onde outros métodos de otimização possam ser melhores, com bastantes experimentos com os parâmetros de otimização, um bom desempenho de procura de solução viável pode também ser obtido usando GA. Naturalmente, para se ter a execução de qualquer método de otimização de uma maneira ótima, algumas experiências com os parâmetros de otimização naquela técnica são requeridas, uma vez que escolhas pobres destes parâmetros podem resultar em fraco desempenho de busca.

2. *Simulated Annealing* e *Tabu Search*

Este trabalho não tem como enfoque as metaheurísticas *Tabu Search* (ou Busca Tabu) (Glover, 1989; Glover, 1990) e *Simulated Annealing* (Kirkpatrick et al., 1983; Cerny, 1985). Visto que *Simulated Annealing* e *Tabu Search* são citados ao longo deste trabalho, aqui são descritas suas características principais.

O *Simulated Annealing* (recozimento simulado) é um algoritmo de otimização similar aos GAs, que se utiliza do mesmo princípio de evolução da solução ao longo do tempo. O termo recozimento refere-se à forma em que os metais são aquecidos e esfriados vagarosamente de modo a garantir uma baixa energia e uma forma estrutural altamente sólida. O recozimento simulado procura varrer todo espaço de busca, com alto nível de movimentação, de forma a encontrar uma solução global e, na seqüência do processo, o resfriamento permitirá apenas pequenos movimentos no espaço de soluções, convergindo para uma solução final. A natureza de movimentos através deste processo significa que uma vez "resfriado", a solução é movida para uma área com menor valor objetivo possível.

O *Simulated Annealing* tem as mesmas vantagens e desvantagens dos algoritmos genéticos. A abordagem pode resolver problemas com funções de alto grau de complexidade, porém a otimização tem um caráter local, e não existe maneira de saber quando uma solução ótima foi alcançada. Novamente, quanto mais se permite que o algoritmo processe a solução, melhor a solução será. Entretanto, não existe maneira de se certificar quando finalizar o processo e obter a melhor solução possível.

Tabu Search (Busca Tabu) é um procedimento adaptativo que guia um algoritmo de busca local na exploração contínua do espaço de busca, sem:

- ser confundido pela ausência de vizinhos aprimorantes;
- retornar a um ótimo local previamente visitado (condição desejada, mas não necessária).

Ainda sobre *Tabu Search*, pode-se dizer que esta metaheurística se utiliza de estruturas flexíveis de memória para armazenar conhecimento sobre o espaço de busca, contrariamente a algoritmos que:

- não utilizam memória (e.g. *simulated annealing*);
- utilizam estruturas rígidas de memória (e.g. *branch-and-bound*, um tipo de método exato, isto é, obtém a solução ótima mas com um elevado custo computacional).

O *Tabu Search* parte de uma solução inicial, e a cada iteração move-se para a melhor solução na vizinhança ("corrente"). Movimentos que levam a soluções na lista tabu (lista de informações para proibir a análise de soluções geradas anteriormente durante um certo número de iterações) são proibidos ("em princípio"). Se o melhor movimento na vizinhança é não-aprimorante, isto levará a uma solução que deteriora o valor da função de custo. Mais informações sobre *Tabu Search*, ver os trabalhos de (Glover, 1989; Glover, 1990).

3. Funcionamento dos Algoritmos Genéticos

O GA simples (Fig. 1) consiste dos seguintes importantes procedimentos (Back, 1996; Schaffer, 1999) para a sua execução:

- **Inicialização da população** - a população inicial de cromossomos é normalmente inicializada aleatoriamente, mas pode ser inicializada conforme algum critério proposto pelo usuário. Pode-se ter um outro tipo de inicialização, mas isto não é crucial uma vez que se tem uma diversidade na população. Se o sistema a ser otimizado for explicitamente conhecido, aquela informação pode ser incluída na população inicial.

- **A avaliação da adaptação dos cromossomos** - o objetivo da função de adaptação é achar um valor numérico para a performance dos cromossomos. Nas aplicações do mundo real, a escolha da função de adaptação é a parte mais crucial dos métodos de otimização de GAs.

- **Seleção** - este operador seleciona os cromossomos da população para reprodução. Os de maiores escores de adaptação são escolhidos uma ou mais vezes para compor uma nova população, de um modo semi-aleatório. Cromossomos de baixo escore são removidos da população. Existem vários métodos de se realizar a seleção. Um dos métodos mais comuns é o método de seleção por competição binária, onde cada cromossomo da população compete por uma posição na nova população. Dois cromossomos são sorteados aleatoriamente da população atual, e

o cromossomo de escore de adaptação maior é colocado na nova população. Ambos os cromossomos são recolocados na população e uma nova competição se inicia, até que a nova população seja completada. A característica deste esquema é que o pior cromossomo da população nunca será selecionado para ser incluído na nova população.

- **Recombinação** - aqui são utilizados os operadores de combinação e mutação. O operador combinação escolhe uma posição do cromossomo (*locus*) e muda as seqüências antes e depois desta posição entre dois cromossomos para criar dois descendentes. A probabilidade (p) destes cromossomos serem combinados é controlada pelo usuário e geralmente é estabelecido um alto valor (p.ex., *pcross* = 0.95). Se a união é permitida, o operador de combinação (cruzamento) é empregado para fazer trocas de genes entre os genitores (*parents*), para a produção de dois descendentes. Se a união não é permitida, os genitores são colocados na próxima geração sem modificações. O operador mutação troca o valor do *locus* (bit) de um cromossomo. Este operador pode ocorrer em cada posição do locus de um cromossomo com uma probabilidade geralmente bem pequena (ex., *pmutation* = 0.001). O processo de geração de novas populações se repete até que se chegue a um número fixo de gerações.

A noção de avaliação e adaptação é usada igualmente. Porém, algumas vezes se distinguem funções de avaliação de funções de adaptação em GAs. Funções de avaliação, ou funções objetivas, fornecem medidas de performance com respeito a um conjunto particular de parâmetros. A função de adaptação transforma aquela medida de performance em uma alocação de oportunidades de reprodução. A avaliação de uma representação de *strings* de um conjunto de parâmetros é independente da avaliação de qualquer outro *string*. A adaptação daquela *string*, contudo, é sempre definida com respeito aos outros membros da população atual (Ochi, 2000).

4. Como implementar um GA

A representação, ou a codificação, das variáveis de otimização tem um grande impacto na performance da procura, uma vez que a otimização é realizada sobre a representação destas variáveis. A escolha mais conveniente de representação é baseada no tipo de aplicação, sendo que as duas representações mais comuns utilizam codificação em binários ou em números reais, diferindo principalmente em como os operadores de recombinação e mutação são executados.

Na prática, a implementação do modelo genético de computação é historicamente feita usando-se uma cadeia de bits que representam os cromossomos, analogamente às quatro bases de cromossomos que são conhecidas do DNA (Adenina, Citosina, Timina, e Guanina). O código binário de cada cromossomo é um vetor que contém zeros e uns representando os genes. Nos problemas de otimização envolvendo variáveis de valores reais, códigos binários são implementados pela conversão de valores numéricos com ponto flutuante em *string* de bits de tamanho fixo, e então retornando a conversão de *string* de bits para número em ponto flutuante, de modo que possa ser utilizado na adaptação. Isto resulta no efeito de mapeamento de um número em ponto flutuante em um conjunto finito de valores discretos determinados pelo número

de bits usados na representação binária. Para formular o cromossomo para otimização, os *strings* de bits são concatenados com os *strings* de bits de outras variáveis para formar um único longo *string* de bits. A Fig. 1, a seguir, ilustra como se deve implementar um algoritmo genético (GA).

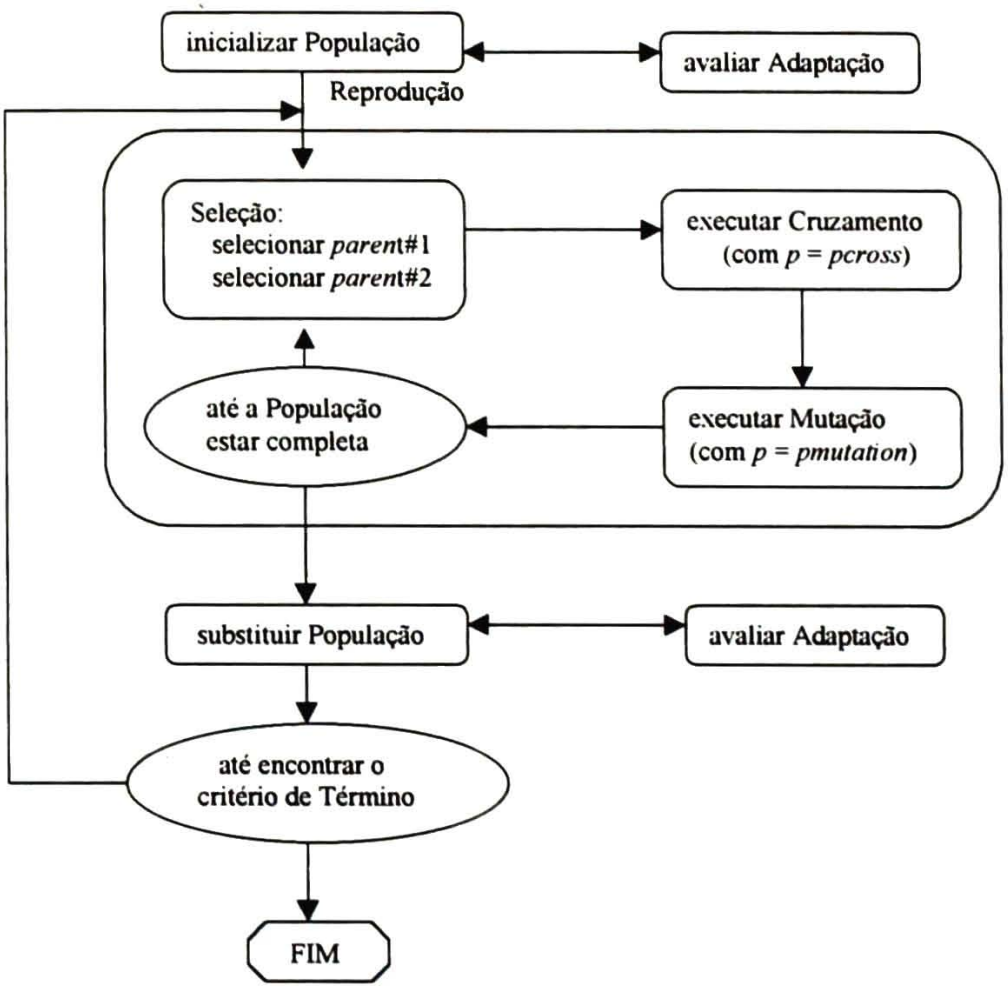

FIG. 1. Fluxograma para implementação de um Algoritmo Genético.

Operações simples de manipulação de bits permitem a implementação de cruzamentos, mutações, e outras operações. Embora alguns pesquisadores tenham executado com *strings* de tamanho variado, e outras estruturas, a maioria trabalha com algoritmos genéticos usando tamanho fixo de caracteres.

5. Por que os GAs funcionam

As pesquisas heurísticas de um GA são baseadas na teoria de esquema de Holland (Holland, 1995). A matemática deste teorema foi desenvolvida usando representação binária, embora em trabalhos recentes tenha sido estendida para incluir representações de números inteiros e reais (Eshelmann & Schaffer, 1993).

Um esquema (H) é definido como um *template* para descrever um subconjunto de cromossomos com seções similares. O *template* consiste de múltiplos de *zeros* e *uns*, e um metacaracter de símbolo '#'. O metacaracter é simplesmente uma notação usada para significar que ou *zero* ou *um* será usado naquele local. Por exemplo, considerando-se o esquema tal como #0000, isto significa dois cromossomos: 10000 e 00000. O *template* é um meio poderoso de descrever similaridades entre os exemplares de cromossomos. O número total de esquemas presentes em um cromossomo de comprimento L é igual a 3^L, visto que se tem 3 possibilidades para cada posição do cromossomo e o número de posições é L. De acordo com Holland (1995), a ordem de um esquema (o(H)) é igual ao número de posições fixas (não-metacaracteres) e a definição de comprimento de um esquema (L(H)) é o número total de caracteres. Assim, o esquema #00#0 é de ordem 3 (o(H)=3) e tem um comprimento 5 (L(H)=5). Holland (1995) deduziu uma expressão para estimar o número de cópias de um esquema particular H que existirá na próxima geração depois da seleção, recombinação e mutação. Esta expressão é dada como:

$$m(H,t+1) >= m(H,t) - f(H)/f[1-(Pr(L(H)/(t-1))-o(H)Pm]$$

onde H é um particular esquema, t é a geração, m(H,t+1) é o número de vezes que um particular esquema é esperado na próxima geração, m(H,t) é o número de vezes em que o esquema existe na atual geração, f(H) é a média da adaptação (*fitness*) de todos os cromossomos que contém o esquema H, f é a média da adaptação de todos os cromossomos, P_r é a probabilidade de recombinação, e P_m é a probabilidade de mutação. (o(H)) é a ordem de H e (L(H)) é o comprimento de H. A conclusão primária que se pode observar desta equação é que enquanto a razão de f(H) e f se torna enorme, o número de vezes que H é esperado na próxima geração aumenta. Assim, um particular bom esquema se propagará para futuras gerações.

Embora tanto a mutação como a recombinação destruam esquemas existentes, elas são necessárias para se criar outros esquemas melhores. O grau pelo qual eles são destruídos é dependente da ordem (o(H)) e do comprimento (L(H)). Assim, esquemas que são de baixa ordem, bem definidos, e tem uma média de adaptação boa são preferidos e são chamados de *building blocks*. Esta definição nos leva para o princípio de *building blocks* de GAs que afirma que existe uma alta probabilidade de que esquemas de adaptação média, de baixa ordem e bem definidos irão se combinar através de recombinação para formar esquemas de adaptação acima da média, de alta-ordem. Recombinação é crucial porque é o único procedimento pela qual os *building blocks* localizados em diferentes seções de um cromossomo podem se combinar num mesmo cromossomo. Pelo emprego deste conceito de *building blocks*, a complexidade de um problema é reduzida. Em vez de se tentar achar alguns esquemas de alta-ordem casualmente, pequenos pedaços do cromossomo que são importantes (isto é, *building blocks*) são combinados com outros importantes pequenos pedaços para produzir, durante muitas gerações, um cromossomo otimizado. GAs são ditos terem a propriedade de um paralelismo implícito porque têm a capacidade de processar muitos esquemas em uma dada geração, e por isso tornam-se algoritmos de otimização eficientes.

6. Quando usar GAs

Muitos cientistas preferem usar métodos de otimização como GAs, *Simulated Annealing, e Simplex* (Ramalhete et al., 1984), dos quais não requerem informações de gradiente, quando não se conhece muito sobre a superfície de resposta e o cálculo do gradiente é computacionalmente pesado ou numericamente instável. Uma das razões de se preferir o uso de GAs é a sua versatilidade. Usando os conhecimentos sobre o sistema, pode-se moldar o algoritmo para uma aplicação particular. Se a aplicação pede um método de otimização com características de *hill-climbing* [3] o algoritmo pode ser modificado pelo uso de estratégia elitista, método introduzido por De Jong que força o GA a manter um certo número de melhores cromossomos para a próxima geração (Mitchell, 1996). Se o problema for o de cair em ótimos locais, a mutação pode ser aumentada. Assim, enquanto não existem garantias de que GAs executam o melhor para uma aplicação particular, pode-se geralmente modificar alguns aspectos da configuração genética, ou usar diferentes operadores genéticos para obter uma performance de procura adequada.

Outra característica interessante e vantajosa é que os GAs não otimizam diretamente as variáveis, mas as suas representações. Aplicações de escolhas de comprimentos de onda tendo os cromossomos codificados tais que cada gene represente um comprimento de onda particular, ou grupo de comprimentos de onda, são particularmente úteis e a experiência mostra que o uso de GAs é uma excelente escolha.

O que acontece se aplicações onde as variáveis sendo otimizadas são muito diferentes umas das outras (p.ex., uma mistura de números inteiros, valores binários, e pontos flutuantes)? GAs são ainda uma excelente escolha para estes tipos de aplicações. A configuração de GA pode ser modificada para incluir diferentes operadores de mutação para seções diferentes do cromossomo. Em recentes trabalhos, o cromossomo incluía seções representando valores binários e outras seções com variáveis inteiras. Tentou-se executar a seleção de comprimento de onda, assim como otimizar o número de variáveis latentes empregado num modelo de calibração multivariada usando um medidor de espectroscopia infravermelho. Foi feita a hipótese de que a inclusão de ambos os tipos de variáveis no cromossomo permitiria que o GA encontrasse as combinações de comprimentos de onda e os números de variáveis latentes que produziria uma melhor previsão de performance do que otimizar os comprimentos de onda separadamente. Os resultados da representação mista foram muito promissores (Schaffer, 1999).

7. Quando não usar GAs

Para aplicações onde o cálculo do vetor gradiente é numericamente preciso e rápido, é recomendado o uso de certas formas do gradiente descendente, por exemplo, o método de Powell (Ochi, 2000). GAs funcionariam para este tipo de aplicações, porém seriam obtidas as regiões ótimas muito mais vagarosamente que os métodos de *hill-climbing*.

Aplicações que requerem um ótimo global exato têm sido um desafio para um GA. GAs são melhores para se achar a região ótima global mas algumas vezes encontram dificuldades

[3] A tradução literal para *hill-climbing* é "subida de morro", e este método de otimização local só considera os movimentos que melhoram a solução. Se não melhorar a solução, o algoritmo termina.

em achar o local ótimo exato. Muitos pesquisadores usam GAs para chegar perto da região ótima e mudam para outro método para uma exploração final.

Uma das dificuldades mais comuns com GAs é que, comparado a técnicas de *hill-climbing*, eles requerem mais respostas (*fitness*) de funções de avaliação. Se a superfície de resposta é muito suave, então os métodos *hill-climbing*, tais como otimização Simplex, superam um GA num dado número de avaliações.

Infelizmente existem certos problemas de otimização, chamados de *GA-hard*, que apresentam um difícil desafio para GAs. Uma das principais áreas de pesquisa de GAs é estudar estes tipos de aplicações e desenvolver métodos para se determinar a priori se o problema de otimização é *GA-hard*. Só recentemente os teóricos de GA têm entendido algumas das causas mais comuns do *GA-hard* (Schaffer, 1999)

Uma das causas do *GA-hard* ocorre quando os genes do cromossomo não são ordenados de forma apropriada e o comprimento do esquema L(H) é muito grande para o GA processá-los corretamente. De acordo com a teoria do esquema, esquemas de comprimento longo têm alta probabilidade de serem destruídos pela mutação e recombinação. Um esquema de reordem onde zeros e uns estão próximos num cromossomo pode ser evitado pela redução de L(H). David Goldbert, citado por Ochi (2000), desenvolveu um método automático de reordem de cromossomos chamado *messy-GA*.

Para um GA otimizar efetivamente, o GA deve seguir o teorema de *building blocks* (esquemas). Experiências mostram que para um GA funcionar deve-se exprimir alguma coisa sobre o todo somente conhecendo-se suas partes. No contexto de um GA, as partes são os *building blocks*, e o todo é o escore da função de adaptação para o cromossomo. O trabalho de Davidor (1990) define *epistasis* como uma interdependência entre as variáveis. Ele elaborou a hipótese de que *epistasis* era importante para determinar se um particular espaço de otimização seguiria os princípios dos *building blocks*. Se o *epistasis* fosse pequeno, nenhuma variável (gene) seria afetada pelos valores das outras variáveis. Quando um espaço de otimização tem um *epistasis* alto, muitas variáveis são dependentes umas das outras e os *building blocks* se tornam de ordem muito alta. Davidor elaborou a hipótese de que os GAs executam quase otimamente entre os dois extremos.

Os resultados de pesquisa de classificação dos *GA-hard* mostram que, para problemas de otimização onde um bom ponto inicial está disponível, as variáveis são altamente correlacionadas (isto é, a ordem dos *building blocks* é grande) e um limitado número de avaliações de resposta (*fitness*) está disponível. Nesse caso, a otimização Simplex parece executar melhor que o *Simulated Anneling* e os GAs. Portanto, isto justifica o ponto que a escolha do melhor método de otimização para uma aplicação requer alguns conhecimentos do sistema.

8. Exemplos de aplicações de Algoritmos Genéticos

Entre as aplicações práticas dos Algoritmos Genéticos, conforme citado no Grupo de Sistemas Inteligentes (2000), da Universidade de Maringa, podem ser destacadas:

- *Multiprocessor Scheduling* - a aplicação de GAs no problema de associação ótima de processos e processadores. O objetivo é diminuir o custo que deriva da

comunicação entre processos em um ordenador paralelo de memória distribuída. *Multiprocessor Scheduling* podem ser relacionados com problemas de robótica.

- Biologia Molecular e Físico-Química - existe uma hipótese de trabalho que sustentaria a utilização de técnicas baseadas em populações devido à estrutura da função objetivo. Em problemas relacionados com estrutura molecular existem experimentos que dão indícios claros sobre a validade dessa hipótese. Assim é fácil observar que são cada vez mais numerosas as aplicações de GAs neste campo.

- Engenharia em Construções - os GAs têm ganhado aceitação em um grande número de problemas de engenharia. Uma aplicação é a otimização discreta de estruturas.

- Geofísica - para o problema chamado de *Seismic Waveform Inversion*, geralmente se utiliza uma estatística Bayesiana na qual se combina a informação que se tem "a priori", sobre o modelo com os dados obtidos "a posteriori". Surgem assim funções que requerem métodos de otimização global. Outro aplicação nessa área é no problema de *Inversion for seismic anisotropy*.

- Redes Neurais - alguns estudiosos têm buscado encontrar uma relação entre Redes Neurais e GAs, tentando correlacionar a Aprendizagem da Redes Neurais com problemas de otimização relacionados com a busca de funções lineares discriminantes em problemas de classificação.

- Compressão de Dados - a compressão de dados em geral, e a compressão de imagens sólidas em particular. Esta aplicação consiste em encontrar um método que utiliza os GAs para encontrar um sistema de funções locais iteradas (LIFS) para a codificação de imagens, produzindo como resultado final uma imagem com qualidade similar àquela utilizada pelo método convencional de compressão fractal, com um tempo 30% menor.

Em termos de pesquisa tem-se procurado melhorar os GAs, visto que estes não são eficientes para todos os casos. Tem-se procurado melhorar os métodos de seleção, mutação e recombinação, bem como unir os GAs a outras técnicas, tais como *Simulated Annealing*, *Tabu Search*, etc. Como exemplo destas novas pesquisas pode ser citado o Algoritmo Genético Construtivo (AGC).

O AGC, uma nova versão de algoritmos genéticos, usa um método construtivo para a formação de uma população com o máximo de informações possíveis sobre o problema em questão. Na tese de doutorado de Furtado (1998) e em Furtado & Lorena (1998) foram conseguidos excelentes resultados para os seguintes problemas de *clustering*: p-medianas, p-medianas capacitado e particionamento de grafos.

A aplicação do AGC ao problema da coloração de grafos foi realizada com sucesso e alcançou excelentes resultados (Ribeiro Filho & Lorena, 1999). Trata-se de um problema clássico de grande importância

Aplicação do AGC ao problema da formação de celulas de manufatura em ambientes de produção também teve bons resultados. Máquinas que processam tarefas são agrupadas em células com o objetivo de tornar a produção mais eficiente. Trata-se também de um problema clássico com grande aplicação prática. Na formulação do AGC usa-se a relação com o problema das p-medianas, com uma medida de distância apropriada. Mais detalhes ver Ribeiro Filho & Lorena (1998)

Em Narciso & Lorena (1999), tem-se uma aplicação de algoritmos genéticos para o Problema Generalizado de Atribuição. Este problema trata da distribuição de n tarefas a m agentes, $n > m$. Como exemplo, pode ser citada a atribuição de n tarefas computacionais distintas a m processadores, visando minimizar o tempo de execução de todas elas, levando-se em conta os diferentes tipos de processadores e tarefas. O AGC foi aplicado a este tipo de problema e os resultados obtidos foram muito próximos da solução ótima. Mais detalhes, ver Narciso & Lorena (1999).

9. Configurando um GA

Um dos aspectos mais desafiantes ao se usar GAs é escolher os valores iniciais dos parâmetros de configuração. Discussões dos teóricos de GA fornecem poucas direções de seleção apropriada de inicializações. Diversos trabalhos de pesquisa têm sido publicados para preencher este vazio. A experiência mostra que o tamanho da população, a taxa da mutação e o tipo de recombinação têm grande efeito na performance da procura. Algumas direções são apresentadas abaixo.

9.1 Tamanho da população

O melhor tamanho para a população é dependente tanto da aplicação quanto do comprimento do cromossomo. Uma boa população de cromossomos contém uma diversidade de seleção de *building blocks*, resultando numa melhor seleção. Se a população perde diversidade, a população tem uma convergência prematura, e pouca exploração é feita. Para cromossomos longos e problemas de otimização desafiantes, tamanhos de população grandes são necessários para manter a diversidade (grande diversidade pode também ser obtida através da alta razão de mutação e de cruzamento uniforme) e assim se obter uma melhor exploração. Muitos pesquisadores sugerem o tamanho da população entre 25 e 100.

9.2 Taxa de mutação

A mutação é usada para dar uma nova informação para a população e também prevenir que a população se torne saturada de cromossomos similares (convergência prematura). Uma elevada taxa de mutação aumenta a probabilidade de que bons esquemas sejam destruídos,

mas aumenta a diversidade da população. A melhor taxa de mutação é dependente da aplicação, mas em muitas aplicações ela está entre 0.001 e 0.1.

Alguns pesquisadores têm publicado a "regra prática" para se escolher a melhor taxa de mutação baseado no comprimento do cromossomo e no tamanho da população. DeJong (1975) sugeriu que a taxa da mutação seja inversamente proporcional ao tamanho da população. Hessner & Manner, citados por Ochi (2000) sugerem que a taxa de mutação seja aproximadamente $(M*L^{1/2})^{-1}$, onde M é o tamanho da população, e L é o comprimento do cromossomo.

9.3 Tipo de recombinação

Embora o tipo básico de recombinação seja o descrito no item 3, existem diversas outras escolhas disponíveis para os usuários de GA. O texto de (Davis, 1996), intitulado "*Handbook of Genetic Algorithms*", tem um tratamento prático de métodos básicos de cruzamentos, assim como alguns métodos de cruzamentos para aplicações específicas. Recomenda-se testar cada um destes métodos, e selecionar um que executa o melhor em conjunto com os outros parâmetros de configuração estabelecidos.

10. Como melhorar a performance de procura

A primeira abordagem para melhorar a performance de procura é simplesmente usar diferentes valores para a taxa de mutação, tamanho da população, etc. Muitas vezes, a performance de procura pode ser melhorada tornando a otimização mais estocástica ou tornando-a mais *hill-climbing*. Esta abordagem de tentativa e erro, embora seja demorada, geralmente resulta em melhoria de performance de procura. Se a modificação dos parâmetros de configuração não produz efeito, um problema mais fundamental pode ser a causa.

Um método simples para melhorar a performance é simplesmente modificar a representação genética. Muitos pesquisadores publicam resultados em que a codificação binária funciona melhor em suas aplicações, enquanto outros mostram resultados diferentes (Schaffer, 1999). A codificação deve ser selecionada tal que os esquemas curtos e de baixa ordem sejam relevantes para o problema de otimização básico. Se a codificação não obedece aos princípios do *building block*, então uma boa performance de procura será difícil de ser alcançada.

Problemas de ligações ocorrem quando os esquemas são localizados bem distantes um do outro no cromossomo. Para superar os problemas de ligação, o comprimento do esquema L(H) deve ser reduzido tal que os esquemas sejam menos destruídos pela recombinação. Isto pode ser executado pela mudança da posição dos genes num cromossomo. Numa aplicação de seleção de comprimentos de onda, a ordem dos comprimentos de onda pode ser mudada tal que altos sinais para comprimentos de onda de ruídos possam ser agrupados juntos no cromossomo. A hipótese é que estes comprimentos de onda sejam mais importantes para a otimização do que os comprimentos de onda de ruídos. Para aplicações de seleção de características ou reconhecimento de padrões, os genes de características semelhantes podem ser codificados de forma adjacentes. Reordenação em cromossomo é uma área onde um conhecimento detalhado da aplicação é necessário. Um método automático de reordenar o cromossomo é usar o *messy GA*. Uma solução potencial para problemas de encadeamento é empregar um operador de cruzamento

uniforme. Cruzamento uniforme não troca seções contiguas do cromossomo entre pais, eliminando o problema de encadeamento

Quando o espaço de otimização contém somente *building blocks* de alta ordem (p.ex., problemas do tipo *GA-hard*), os meios primários para processa-los não são por recombinação, mas por mutação. Assim, aumentando as propriedades estocásticas do algoritmo por cruzamento uniforme ou alta taxa de mutação, pode-se superar este problema parcialmente.

11. Comentários e conclusões sobre GAs

Os Algoritmos Genéticos formam a parte da área de sistemas inspirados na natureza, simulando os processos naturais e aplicando-os à solução de problemas reais. São métodos generalizados de busca e otimização que simulam os processos naturais de evolução, aplicando a idéia darwiniana de seleção. De acordo com a aptidão e a combinação com outros operadores genéticos, são produzidos métodos de grande robustez e aplicabilidade. Estes algoritmos estão baseados nos processos genéticos dos organismos biológicos, codificando uma possível solução a um problema de "cromossomo" composto por cadeia de bits e caracteres. Estes cromossomos representam indivíduos que são levados ao longo de várias gerações, na forma similar aos problemas naturais, evoluindo de acordo com os princípios de seleção natural e sobrevivência dos mais aptos, descritos pela primeira vez por Charles Darwin em seu livro "Origem das Espécies". Emulando estes processos, os Algoritmos Genéticos são capazes de "evoluir" soluções de problemas do mundo real.

O princípio básico do funcionamento dos GAs é que um critério de seleção vai fazer com que, depois de muitas gerações, o conjunto inicial de indivíduos gere indivíduos mais aptos. A seleção natural garante que os cromossomos melhores adaptados se propaguem para as futuras populações. Além do critério de seleção, os cromossomos melhores adaptados são procurados também através de operadores genéticos: mutação e recombinação.

Os Algoritmos Genéticos podem convergir em uma busca aleatória, porém sua utilização assegura que nenhum ponto do espaço de busca tem probabilidade zero de ser examinado. Toda tarefa de busca e otimização possui vários componentes, entre eles: o espaço de busca onde são consideradas todas as possibilidades de solução de um determinado problema, e a função de avaliação (ou função de custo), uma maneira de avaliar os membros do espaço de busca. As técnicas de busca e otimização tradicionais iniciam-se com um único candidato que, iterativamente, é manipulado utilizando algumas heurísticas (estáticas) diretamente associadas ao problema a ser solucionado. Por outro lado, as técnicas de computação evolucionária operam sobre uma população de candidatos em paralelo. Assim, elas podem fazer a busca em diferentes áreas do espaço de solução, alocando um número de membros apropriado para a busca em várias regiões. Os Algoritmos Genéticos diferem dos métodos tradicionais de busca e otimização, principalmente em quatro aspectos:

a) GAs trabalham com uma codificação do conjunto de parâmetros e não com os próprios parâmetros;
b) GAs trabalham com uma população e não com um único ponto;

c) GAs utilizam informações de custo ou recompensa e não derivadas ou outro conhecimento auxiliar;
d) GAs utilizam regras de transição probabilísticas e não determinísticas.

Algoritmos Genéticos são muito eficientes para busca de soluções ótimas, ou aproximadamente ótimas, em uma grande variedade de problemas, pois não impõem muitas das limitações encontradas nos métodos de busca tradicionais. Os pesquisadores referem-se a "algoritmos genéticos" ou a "um algoritmo genético" e não "ao algoritmo genético", pois GAs são uma classe de procedimentos com muitos passos separados, e cada um destes passos possui muitas variações possíveis. Os GAs não são a única técnica baseada em uma analogia da natureza. Por exemplo, as Redes Neurais estão baseadas no comportamento dos neurônios do cérebro, podendo ser utilizadas em uma grande variedade de problemas de classificação, como reconhecimento de padrões no processamento de imagens.

Dentre as vantagens que se pode citar sobre o uso de GAs, podem-se destacar: não requer nenhum conhecimento ou informações de gradiente de uma superfície de resposta; a presença de descontinuidades na superfície de resposta tem pouco efeito na performance geral da otimização; resistem em não cair nas armadilhas de ótimos locais; realizam muito bem os problemas de otimização de larga escala; e podem ser empregados em uma grande variedade de problemas de otimização.

Apesar do seu caráter genérico, o desempenho dos GAs na sua forma convencional não se mostra satisfatório para muitos problemas que já possuem algoritmos eficientes para a sua solução aproximada. Esta é a motivação pela qual, nos últimos anos, diversos pesquisadores terem proposto modificações nos GAs convencionais, incorporando técnicas de busca local, ou ferramentas de outras metaheurísticas tais como: *Simulated Annealing*, *Tabu Search*, entre outros, ou desenvolverem versões paralelas para melhorar o desempenho dos GAs (Ochi, 2000). Como exemplo de modificação em GAs tradicionais, podem ser citados os trabalhos de Furtado (1998); Furtado & Lorena (1998); Narciso & Lorena (1999); Ribeiro Filho & Lorena (1999).

Existem atualmente conferências internacionais dedicadas a tratar de algoritmos genéticos. O número de artigos sobre GAs cresce rapidamente. Alander (2000) catalogou uma extensa bibliografia de mais de 2500 livros, artigos, teses dedicados GAs. Alander mostrou que existe um rápido crescimento, com uma taxa anual de aproximadamente 40%. Existem alguns laboratórios de pesquisa famosos trabalhando na computação evolutiva. Nos EUA, temos como exemplo o grupo de Holland na Universidade de Michigan (Michigan State University, 2000), o de Goldberg em Illinois (Illinois Genetic Algorithms Laboratory, 2000) o de Goldman e Punch no Estado de Michigan (Michigan State University, 2000), o de De Jong na Universidade George Mason (Genetic Algorithms Group, 2000). Também existem trabalhos no Centro Naval de Pesquisa Aplicada em Inteligência Artificial (Naval Research Laboratory, 2000) e no Instituto Santa Fe (Santa Fe Institute, 2000). No Reino Unido existem o grupo de Flemings em Sheffield (University of Sheffield, 2000), o grupo de computação evolutiva da Universidade do Oeste da Inglaterra em Bristol (Intelligent Computer Systems Centre, 2000) e o da Universidade de Edinburgh (University of Edinburgh, 2000). No Brasil, existem diversos pesquisadores e instituições (INPE, Unicamp, USP, ITA, etc.) com trabalhos publicados na área.

Os GAs são mais que apenas um outro método de otimização. Muitos cientistas estudam GAs para entender como a evolução acontece, e também alguns conceitos mais

esotéricos como a música genética e arte genética. Alguns cientistas até discutem que GAs não pretendiam ser um método de otimização numérica, e que comparações com outros métodos não são justas.

Assim, usar os métodos de GAs para um problema de otimização é uma questão difícil de responder. Muitos cientistas dizem não existir um método ótimo de otimização (Schaffer, 1999). Existem estas discussões no grupo da Usenet comp.ai.genetic – Encore (Evolutionary Computation Repository, 2000). Usados cegamente, qualquer técnica de otimização terá a mesma performance. A chave da escolha para a melhor abordagem está no uso conhecimento do sistema a ser otimizado.

12. Referências bibliográficas

ALANDER, J. T. (Ed.). **Proceedings of the second Nordic Workshop on Genetic Algorithms and their Applications (2NWGA)**. Disponível em: <http://www.uwasa.fi/cs/publications/2NWGA/2NWGA.html>. Acesso em: 20 dez. 2000

BACK, T. **Evolutionary algorithms in theory and practice:** evolution strategies, evolutionary programming, genetic algorithms. New York: Oxford University Press, 1996. 314p.

CERNY, V. A thermodynamical approach to the travelling salesman problem: an efficient simulated annealing algorithm. **Journal of Optimization Theory and Application,** v. 45, p. 41-51, 1985.

DAVIDOR, Y. Epistasis variance –A Viewpoint on GA-Hardness. In: WORKSHOP ON FOUNDATIONS OF GENETIC ALGORITHMS - FOGA, 1990, 1, Bloomington Campus, USA, 1990. **Proceedings**. [Indiana, USA, s. n.,1990]. P 23-35. Disponível em http://www.mpi-sb.mpg.de/services/library/proceedings/contents/foga90.html. Acesso em: 08 de janeiro de 2001.

DAVIS, L. D. **Handbook of genetic algorithms.** London: ITP, c1996. 385p.

DE JONG, K. **An analysis of the behaviour of a class of genetic adaptive systems**. 1975. Ph.D. Thesis (PhD) - University of Michigan, Ann Arbor. 1975.

ESHELMANN, L. D.; SCHAFFER, J. D. A study of control parameters affecting online performance of genetic algorithms for function optimization. In: WORKSHOP ON FOUNDATIONS OF GENETIC ALGORITHMS 2 (FOGA 93), 4., 1993, Bloomington. **Proceedings...** [Bloomington: s.n., 1993]. p. 5-17.

EVOLUTIONARY COMPUTATION REPOSITORY. **The Hitch-Hicker's guide to evolutionary computation**. Disponível em: <http://alife.santafe.edu/~joke/encore>. Acesso em: 27 jan. 2000.

EVOLUTIONARY COMPUTING GROUP. [**Evolutionary Computing Group**: home page]. Disponivel em: <http://www.soi.city.ac.uk/~peters/Evolutionary.html>. Acesso em: 20 dez. 2000.

FURTADO, J. C. **Algoritmo genético construtivo na otimização de problemas combinatoriais de agrupamentos**. 1998. Tese (Doutorado em Computação Aplicada) - Instituto Nacional de Pesquisas Espaciais, São José dos Campos.

FURTADO, J. C.; LORENA, L. A. N. Algoritmo genético construtivo na otimização de problemas combinatoriais de agrupamentos. In: OFICINA DE CORTES E EMPACOTAMENTO, 3., 1998, Curitiba. **Anais da 3. Oficina de Cortes e Empacotamento.** [Curitiba: s.n., 1998]. p. 10-26. Também disponível em: <http://www.lac.inpe.br/~lorena/sbpo98/agc-clust.pdf >. Acesso em: 20 dez. 2000.

GENETIC ALGORITHMS GROUP. **The Genetic Algorithms Group, George Mason University.** Disponível em: <http://www.cs.gmu.edu/research/gag>. Acesso em: 27 jan. 2000.

GLOVER, F. Tabu search – Part I. **ORSA Journal on Computing,** v. 1, p. 190-206, 1989.

GLOVER, F. Tabu search – Part II. **ORSA Journal on Computing,** v. 2, p. 4-32, 1990

GOLDBERG, D. E. **Genetic algorithms in search, optimization and machine learning.** Reading: Addison Wesley, 1989. p. 11-172.

GRUPO DE SISTEMAS INTELIGENTES. **Algoritmos genéticos**.
Disponível em: <http://www.din.uem.br/ia/geneticos>. Acesso em: 20 jan. 2000.

HOLLAND, J. H. **Adaptation in natural and artificial systems**: an introductory analysis with applications to biology, control and artificial intelligence. Cambridge: MIT, 1995. 211 p.

ILLINOIS GENETIC ALGORITHMS LABORATORY. **IlliGAL home page**. Disponível em: <http://gal4.ge.uiuc.edu>. Acesso em: 27 jan. 2000.

INTELLIGENT COMPUTER SYSTEMS CENTRE. **ICSC home**. Disponível em: <http://www.csm.uwe.ac.uk/icsc/index.html>. Acesso em: 20 dez. 2000.

KIRKPATRICK, S.; GELAT, C. D.; VECCHI, M. P. Optimization by simulated annealing. **Science**, v. 220, p. 671-680, 1983.

MICHIGAN STATE UNIVERSITY. **MSU GARAGe home page.** Disponível em: <http://garage.cse.msu.edu>. Acesso em: 20 dez. 2000

MITCHELL, M. **An introduction to genetic algorithms**. Cambridge: The MIT Press, 1996. 168p. 205 p.

NARCISO, M. G.; LORENA, L. A. N. Algoritmo genético construtivo aplicado ao problema generalizado de atribuição. In: SOBRAPO, 31., 1999, Juiz de Fora. **Anais da 31. SOBRAPO**. [Juiz de Fora: s.n., 1999]. p. 91-106.

NAVAL RESEARCH LABORATORY. Information Technology Division. **Navy Center for Applied Research in Artificial Intelligence – NRL.** Disponível em: <http://www.aic.nrl.navy.mil>. Acesso em: 27 jan. 2000.

OCHI, L. S. **Algoritmos genéticos**: origem e evolução. Disponível em: <http://www.info.lncc.br/sbmac/sem-fig/public/bol/BOL-2/artigos/satoru/satoru.html>. Acesso em: 27 jan. 2000.

RAMALHETE, M.; GUERREIRO, J.; MAGALHÃES, A. **Programação linear**. Lisboa: McGraw-Hill, 1984. p. 166-280.

RIBEIRO FILHO, G.; LORENA, L. A. N. Algoritmo genético construtivo aplicado ao projeto de células de manufatura. In: OFICINA DE CORTES E EMPACOTAMENTO, 3.; SBPO, 30., 1998, Curitiba. **Anais da III Oficina de Cortes e Empacotamento**. [Curitiba: s.n., 1998]. p. 131-142.

RIBEIRO FILHO, G.; LORENA, L. A. N. **Improvements on constructive genetic approaches to graph coloring.** Versão resumida apresentada na 1ª Oficina do projeto temático FAPESP - Planejamento e Controle da Produção em Sistemas de manufatura. UNICAMP, abril, 1999. Não publicado.

SANTA FE INSTITUTE. **SFI home page**. Disponível em: <http://www.santafe.edu>. Acesso em: 27 jan. 2000.

SCHAFFER, J. D. **A practical guide to genetic algorithms**. Naval Research Laboratory. Disponível em: <http://chemdiv-www.nrl.navy.mil/6110/sensors/chemometrics/practga.html>. Acesso em: 20 dez. 2000.

UNIVERSITY OF EDINBURGH. Division of Informatics. Department of Artificial Intelligence. **Edinburgh University – Artificial Intelligence**. Disponível em: <http://www.dai.ed.ac.uk>. Acesso em: 27 jan. 2000.

UNIVERSITY OF SHEFFIELD. Department of Automatic Control & Systems Engineering Department. Disponível em: <http://www.shef.ac.uk/uni/projects/gaipp/index.html>. Acesso em: 27 jan. 2000.

Empresa Brasileira de Pesquisa Agropecuária
Centro Nacional de Pesquisa Tecnológica em Informática para a Agricultura
Ministério da Agricultura e do Abastecimento
Av. Dr. André Tosello s/nº. - Cidade Universitária "Zeferino Vaz"
Barão Geraldo - Caixa Postal 6041
13083-970 - Campinas, SP
Telefone (19) 3289-9800 - Fax (19) 3289-9495
sac@cnptia.embrapa.br
http://www.cnptia.embrapa.br

MINISTÉRIO DA AGRICULTURA E DO ABASTECIMENTO

GOVERNO FEDERAL
Trabalhando em todo o Brasil