Patrícia Akemi Ikeda

Introdução aos Algoritmos Genéticos

Introdução

Métodos de busca tradicionais são otimizados para encontrar informações em meio a dados armazenados. Estes dados possuem características bem definidas, fator que restringe o espaço de busca e oferece uma direção precisa para o algoritmo. Na busca de um produto no catálogo de uma loja, por exemplo, podemos utilizar uma rotina de busca binária ou outro tipo adequado à esse problema.

Por outro lado, resolver quebra-cabeças como Torre de Hanoi ou o problema da mochila, não há como buscar a solução em um banco de dados, já que os caminhos para atingir o objetivo dependem da sua configuração inicial. Os métodos tradicionais são ineficientes, ou até mesmo inviáveis, pois seria necessário uma busca exaustiva, passando por cada combinação possível até encontrarmos a solução ótima, e ainda assim, sem garantias de sucesso. Nestes casos em que o espaço de busca é extenso (ou até mesmo infinito), uma boa alternativa são os algortimos genéticos.

Algoritmos Genéticos

Algoritmos Genéticos (AG) são uma técnica computacional utilizada para encontrar soluções aproximadas em problemas de otimização e busca. É inspirada na forma como os seres vivos sobrevivem e passam seu material genético para as próximas gerações, utilizando princípios de seleção natural e evolução propostos por Charles Darwin.

Na teoria de Darwin, as espécies são selecionadas segundo suas capacidades, expressas por meio de seus comportamentos (fenótipo) e genoma (genótipo), determinando as mais aptas a sobreviver e, consequentemente, se reproduzir. Quanto melhor um indivíduo se adaptar ao seu meio ambiente, maior será sua chance de sobreviver e gerar descendentes.

Os seres vivos tem no núcleo de suas células arranjos de DNA, chamados genes, que determinam suas características. A evolução genética ocorre nos cromossomos, uma sequência de genes responsáveis pela codificação dos indivíduos, onde estruturas bem sucedidas se reproduzem mais vezes do que aqueles que codificaram estruturas mal sucedidas. Além da reprodução, na qual os filhos possuem uma combinação dos cromossomos dos pais, podem ocorrer eventuais mutações, conferindo vantagens, desvantagens ou até mesmo não produzir efeito algum.

Os AGs se baseiam principalmente na evolução das espécies como forma de alcançar a solução para um problema. A abordagem evolutiva é utilizada para compreender como a natureza competitiva em que se encontram ajuda a buscar a ser "o mais forte", que no caso da heurística não significa apenas força, mas sim um conjunto de informações que, quando processadas no meio em que se encontram, trazem melhores resultados que outras combinações. O processo de evolução é aleatório porém, guiado por um mecanismo de seleção baseado na adaptação de estruturas individuais.

Uma das grandes vantagens está no fato de simplificarem a formulação e solução de

problemas complexos que trabalham com um grande número de variáveis (consequentemente, espaços de soluções de dimensões elevadas), além de não impor muitas das limitações encontradas nos métodos de busca tradicionais. São capazes de identificar e explorar fatores ambientais e se aproximar de soluções ótimas, ou aproximadamente ótimas, em níveis globais. Em muitos casos onde outras estratégias de otimização falham na busca de uma solução, os AGs convergem. Possuem um paralelismo implícito decorrente da avaliação independente de cada indivíduo e não são sensíveis a erros de arredondamento no que se refere a resultados finais.

Os AGs foram propostos por John Holland em 1975, com a publicação de seu livro *Adaption in Natural and Artificial Systems*. O trabalho foi resultado do estudo, em parceria com seus alunos na Universidade de Michigan, de processos naturais adaptáveis e no desenvolvimento de modelos em que os mecanismos da evolução pudessem ser importados para os sistemas computacionais. Em 1989, David Goldberg edita *Genetic Algorithms in Search, Optimization and Machine Learning* que, juntamente com a obra de Holland, é considerado como um dos principais livros sobre o assunto. Desde então, vêm sendo aplicados com sucesso na resolução de vários problemas práticos de otimização combinatória.

Existe uma clara distinção entre um processo de otimização, que pode ser visto como um processo de aperfeiçoamento, e o seu objetivo máximo, o ótimo. Em geral, as técnicas de otimização são avaliadas primeiramente pela sua capacidade de convergir para o ótimo. Questões ligadas à velocidade de convergência e ao esforço demandado para atingí-lo tendem a ser consideradas como secundárias. Esta tendência deriva da natureza matemática dos problemas de otimização. Quando não há tempo e recursos suficientes para encontrar a melhor solução dentre todas as possíveis, para os problemas do dia-a-dia, é melhor desenvolver a capacidade de criar soluções intuitivas e aperfeiçoá-las ao máximo, respeitando as limitações de tempo e recursos que o mundo real impõe.

Segundo Goldberg (1989), o AG difere dos métodos tradicionais de busca e otimização, principalmente em quatro aspectos:

- Manipulação de código - AG opera num espaço de soluções codificadas, enquanto que os outros métodos controlam as variáveis diretamente.
- Procura pelo ótimo é feita a partir de uma população de pontos e não de um ponto isolado.
- A procura é cega e feita por amostragem. A única informação necessária é o valor da função objetivo, não exigindo o uso de derivadas ou qualquer outro tipo de conhecimento.
- O AG usa transições probabilísticas e não regras determinísticas.

Termos chave

- indivíduo - qualquer solução possível
- cromossomo - codificação das características de um indivíduo
- população - grupo de todos os indivíduos
- espaço de busca - todas as soluções possíveis para o problema
- genoma - coleção de todos os cromossomos de um indivíduo

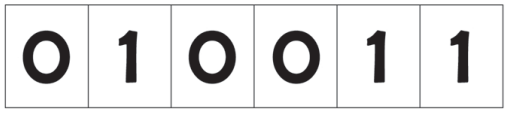
Estrutura básica de um AG

Um AG é estruturado de forma que as informações referentes a determinado sistema possam ser codificadas de maneira análoga aos cromossomos biológicos. Desta forma, o algoritmo assimila-se muito ao processo evolutivo natural.

Cromossomo

Em um AG, o cromossomo é uma estrutura de dados que representa uma possível solução para o problema. Existem várias formas de codificá-lo. A mais comum é utilizar uma série (string) binária, onde cada bit da série representa alguma característica da solução.

Exemplo de representação de um cromossomo.



Exemplos de outras codificações possíveis:

- vetor de números reais (2.946, 5.78, 1, 4.29)
- vetor de números inteiros (3, 8, 5, 1, 7)

A forma como o cromossomo é representado dependerá das particularidades do problema a ser solucionado. A essa representação, dá-se o nome de alfabeto do AG.

Criação da população inicial e pontuação

Inicialmente, é gerada uma população formada por um conjunto aleatório de indivíduos, ou seja, possíveis soluções do problema. Estas são codificadas em uma forma inspirada no genoma dos seres vivos: um vetor de características que possa ser desmembrado e recombinado com partes dos vetores de outras soluções.

Através da função objetivo, a população é avaliada: para cada indivíduo é dada uma nota, ou índice, refletindo sua habilidade de adaptação a determinado ambiente. A aptidão é definida com relação à população corrente. A função objetivo é construída a partir dos parâmetros envolvidos no problema. Ela fornece uma medida da proximidade da solução em relação a um conjunto de parâmetros, possibilitando calcular sua probabilidade de ser selecionado. Os parâmetros podem ser conflitantes, ou seja, quando um aumenta o outro diminui. O objetivo é encontrar o ponto de equilíbrio, o ótimo.

Seleção

Para o processo de recombinação é necessário selecionar os pais dos indivíduos que formarão a nova geração. A escolha deve ser feita de tal forma que os membros mais adaptados ao ambiente tenham maior chance de se reproduzir, isto é, apresentam um valor da função fitness mais elevado. Apesar disso, não apenas os melhores são escolhidos, a fim de evitar a convergência no máximo local (valor que, quando comparado à sua vizinhança, parece ser o melhor, mas não é efetivamente a melhor solução para o problema).

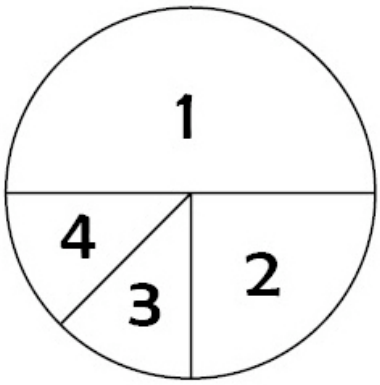
Dessa forma, um dos métodos utilizados é o da roleta (Roulette Wheel), onde se atribui a cada cromossomo uma probabilidade de ser selecionado, proporcional ao seu índice de aptidão. Os que possuem um índice mais alto, ocuparão uma porção maior do que aqueles com valor menor. É importante notar que este procedimento permite a perda do melhor indivíduo, assim como alguém seja selecionado mais de uma vez.

O número de vezes que a roleta é girada varia conforme o tamanho da população. A cada giro, é selecionado um indivíduo que participará do processo de geração da nova linhagem. O objetivo é direcionar o AG para as melhores regiões do espaço de busca.

No exemplo abaixo é possível visualizar como os cromossomos são alocados na roleta, de acordo com sua aptidão. O número 1, devido ao seu alto valor de fitness, possui 50% de chance de ser escolhido para dar origem à próxima geração, o que não impede de 3 e 4 também terem direito de participar do processo, embora com probabilidades menores.

Cromossomo	Fitness	%
1	10	50
2	5	25
3	2,5	12,5
4	2,5	12,5

Exemplo de cromossomos alocados na roleta de acordo com sua aptidão



Crossover

Após os indivíduos serem selecionados, inicia-se o processo de cruzamento, principal mecanismo na geração de novos pontos no espaço de otimização. Nesta fase há troca de genes entre dois membros, denominados pais, dando origem a outros membros, os filhos (offspring).

Existem várias de formas de fazer o crossover, sendo as mais comuns apresentadas abaixo.

Crossover em 1 ou mais pontos

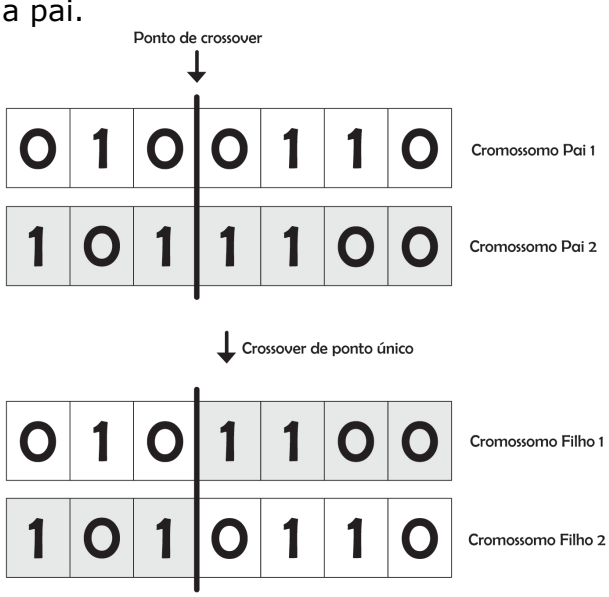
Para que tal cruzamento seja possível, 2 fatores devem ser analisados:

- saber se a combinação será efetivamente realizada através da taxa de crossover
- determinar em qual(is) ponto(s) o crossover será feito

Por exemplo, se a taxa for de 0.7 (70%), considerando uma escala entre 0 e 1, e o valor sorteado for maior, então o crossover não é realizado, ou seja, os filhos serão idênticos aos pais. Caso contrário, é feito o corte de cruzamento em ponto(s) determinado(s) pelo algoritmo, criando indivíduos com a combinação dos genes dos pais.

No exemplo abaixo, o crossover ocorre em apenas 1 ponto, resultando em filhos

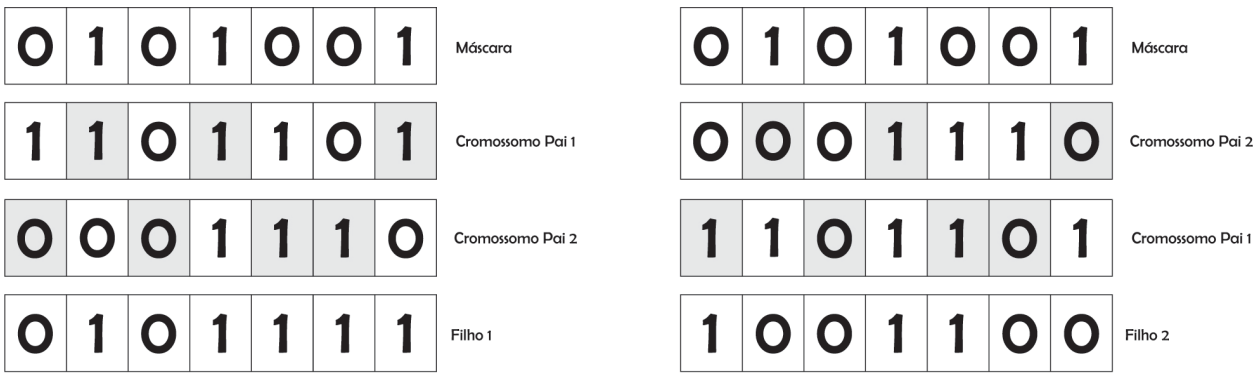
com um pedaço de cada pai.



Exemplo de crossover em um ponto

Crossover uniforme

Cada bit do descendente é escolhido de acordo com uma máscara de cruzamento gerada aleatoriamente. Onde houver 1 na máscara, o bit virá do primeiro pai, caso contrário, virá do segundo. O processo é repetido com os pais trocados para produzir o segundo filho. Para cada cruzamento, uma nova máscara é criada.



Exemplo de crossover uniforme

Crossover aritmético

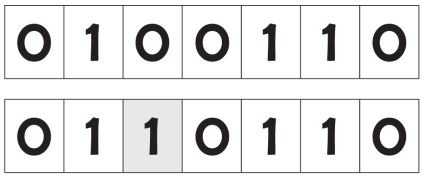
O cromossomo filho é resultado de uma combinação aritmetica dos pais.

Mutação

A mutação é a alteração de uma informação genética a fim de garantir a diversidade e explorar áreas desconhecidas do espaço de busca, prevenindo que todas as soluções caiam em um ponto ótimo local. Enquanto o crossover cuida da troca de informações

entre dois cromossomos, as mutações alteram sem relacionar com outro indivíduo. Ela muda aleatoriamente a descendência criada pelo cruzamento.

Há diversas formas de realizar o processo, que depende da forma como o cromossomo é codficado. Respeitando uma taxa (probabilidade) de mutação, que decide se o gene será modificado ou não, assim como ocorre no crossover, pode-se trocar um bit de 0 para 1, no caso de cadeias binárias, ou sortear um novo número, caso a cadeia seja composta por números inteiros ou reais. A taxa não deve ser nem alta nem baixa, mas o suficiente para assegurar a diversidade de cromossomos na população.



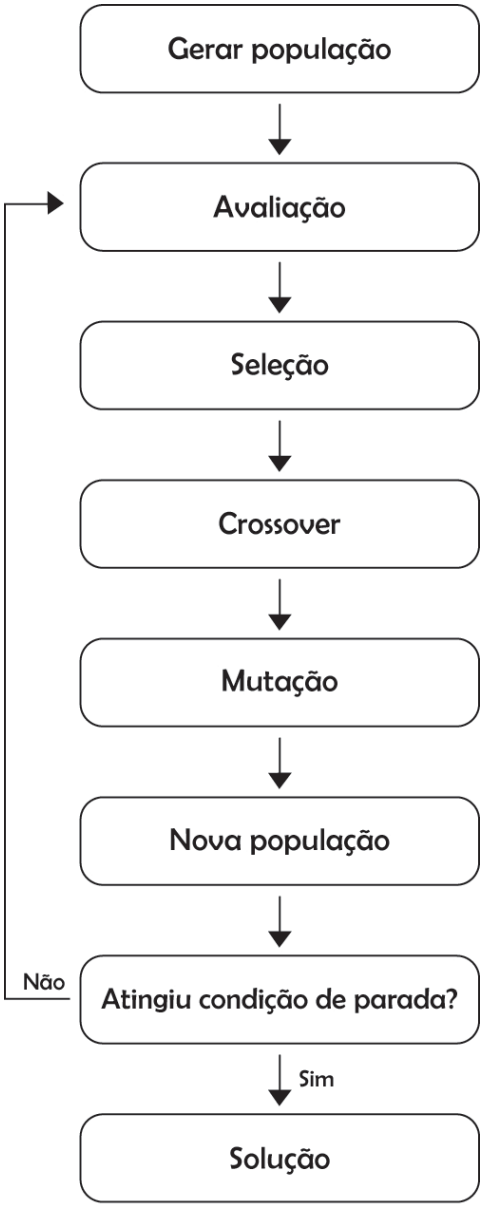
Exemplo de mutação em 1 bit

Nova geração

Depois que todos os processos anteriores forem executados, surge uma nova população. Esta, por sua vez, servirá de base para a próxima iteração do algoritmo, e assim sucessivamente, criando um ciclo de busca progressiva na busca por seres ideais, até que a condição de parada seja atingida. Os algoritmos genéticos baseiam-se na combinação da estratégia "gerar-e-testar" juntamente com a teoria darwiniana da evolução e com a genética.

Existem ainda outros conceitos, como o elitismo, em que indíviduos com melhor fitness de cada geração são preservados para gerações futuras, além de diversas estratégias para se selecionar adequadamente os indivíduos que serão cruzados durante o processo de crossover. De acordo com as particularidades de cada problema, estratégias adequadas são adotadas no intuito de atingir a melhor solução.

A figura abaixo ilustra o funcionamento de um AG:



Parâmetros do AG

É importante também analisar a forma como alguns parâmetros influem no comportamento dos AGs, para que se possa estabelecê-los de acordo com as necessidades do problema e dos recursos disponíveis.

Tamanho da população

O tamanho da população (que mede o número de cromossomos em uma geração) afeta o desempenho global e a eficiência dos AGs. Com uma população pequena, o algoritmo terá poucas possibilidades de realizar cruzamentos, além da cobertura do espaço de soluções ser reduzido. Uma grande população geralmente fornece uma cobertura representativa do domínio do problema, além de prevenir convergências prematuras para soluções locais ao invés de globais. No entanto, se houver muitos cromossomos, serão necessários mais recursos computacionais ou que o AG trabalhe

por um período de tempo maior. Pesquisas mostram que após determinado limite (que depende principalmente da codificação e do problema), não é conveniente aumentar a população porque isso não resolve o problema mais rapidamente do que com tamanhos moderados de população.

Taxa de crossover

Taxa de crossover é a frequência com que o cruzamento é realizado. Quanto maior for a taxa, novas estruturas serão introduzidas na população mais rapidamente, na esperança de que os novos cromossomos contenham partes boas dos ancestrais, se tornando indivíduos melhores.

Por outro lado, se a taxa for muito alta,pode provocar a perda de estruturas de alta aptidão pois grande parte da população será substituída, dificultando chegar na resposta de forma clara, rápida e precisa. Se a taxa for muito baixa, o algoritmo pode ficar muito lento, demorando para atingir a solução.

Taxa de mutação

Taxa de mutação é a frequência com que as partes dos cromossomos sofrerão mutação. Se a taxa for nula, a descendência é gerada logo após o cruzamento sem nenhuma alteração. Uma baixa taxa de mutação previne que uma dada posição fique estagnada em um valor, não convergindo no máximo local, além de possibilitar que se chegue em qualquer ponto do espaço de busca. Já com valores muito altos a busca se torna essencialmente aleatória.

Número de gerações

O número de gerações constitui o número de vezes que os passos do AG serão repetidos até chegar em uma solução, e varia conforme a complexidade do problema. O ideal seria que o processo fosse feito até chegar no ponto ótimo, mas como nem sempre isso é possível, normalmente é determinado um número máximo de gerações ou um tempo limite de processamento.

Intervalo de geração

O intervalo de geração indica a porcentagem da população que será substituída na próxima geração. Valores muito altos implicam que grande parte dos indivíduos serão substituídos, podendo ocorrer perda de estruturas de alta aptidão. Já valores muito baixos, com pequena parte da população sendo alterada, pode-se demorar muito para chegar na solução.

Elitismo

Elitismo trata-se da repetição dos cromossomos com melhores resultados na próxima geração a fim de evitar que todos os bons cromossomos sejam alterados pelo crossover e pela mutação, permitindo que o algoritmo convirja mais rapidamente para uma solução. Por outro lado, deve-se tomar cuidado com seu uso, pois quanto maior for o número de cromossomos a serem mantidos, maior a probabilidade de convergir para um máximo local.

Recomendações

Uma escolha mal feita dos parâmetros do AG pode prejudicar o desempenho do algoritmo na obtenção dos resultados esperados. Reconhecer o efeito individual e combinado desses parâmetros favorece a obtenção de melhores resultados. Embora não exista uma teoria geral que possa ser aplicada em qualquer problema, há recomendações a respeito das implementações de AG, apesar de não haver um consenso. Eis algumas delas:

1. Tamanho da população

Pode ser um pouco surpreendente que populações de tamanho muito grande, normalmente não aumentam o desempenho do AG (no sentido de aumentar a velocidade com que são encontradas as soluções). Alguns autores também mostram que o melhor tamanho da população depende do tamanho da série codificada (cromossomos), ou seja, para cromossomos com 32 bits, a população deve ser maior do que se tivesse cromossomos com 16 bits.

Tanomaru (1995) afirma que, na prática, valores entre 50 e 200 cromossomos resolvem a maior parte dos problemas, mas populações maiores podem ser necessárias para problemas complexos. Tem-se também o trabalho de Mitchell (1996), que diz que o melhor tamanho para a população é entre 50 e 100 indivíduos. Laura Núñez-Letamendia (2007) utilizou em seus experimentos um tamanho de população igual 100. Já Cheng e Chang (2007) utilizaram valores entre 20 e 40 para o tamanho da população.

2. Taxa de crossover

A taxa de cruzamento deve em geral ser alta, cerca de 80% a 95%. Entretanto, alguns resultados mostram que para alguns tipos de problemas, uma taxa de cruzamento de cerca de 60% é o melhor.

Tanomaru (1995) relata que bons resultados geralmente são obtidos com alto valor da taxa de recombinação (maior que 70%). Já Mitchell (1996) sugere uma taxa de 60%, enquanto que Núñez-Letamendia (2007) definiram esse valor em 95%.

3. Taxa de mutação

Por outro lado, a taxa de mutação deve ser muito baixa. As melhores taxas parecem estar na faixa de 0.1% a 1%.

Tanomaru (1995) sugere um baixo valor para a taxa de mutação (menor que 1%). Já Mitchell (1996) sugere uma taxa de 0,1%, enquanto que Núñez-Letamendia (2007) definiram esse valor em 1%.

Thomas Bäck (1996), constatou em suas últimas pesquisas que o desempenho do AG tende a cair em populações relativamente grandes (maior que 200) com grande taxa de mutação (maior que 5%) e em populações pequenas (menos que 200) com pequena probabilidade de mutação (menor que 2%).

Aplicações

Os algoritmos genéticos podem ser utilizados em uma ampla variedade de problemas complexos. Eis alguns exemplos:
• Otimização de funções numéricas em geral
• Determinação de rotas e sequenciamento de tarefas
• Programação genética
• Balanceamento de carga em sistemas multiprocessados
• Escalonamento de tarefas
• Processamento de imagens
• Computação evolutiva - programas que se adaptam a mudanças no sistema ao longo do tempo
• Problemas de otimização complexos (problema do caixeiro viajante, gerenciamento de carteiras de investimento etc)
• Fabricação de roupas - cortar um tecido com perda mínima de material

Exemplos de aplicações retirados de [22]

1. Petróleo e Gás – Inversão Sísmica

O problema da inversão sísmica, muito importante no campo da geologia, consiste na determinação da estrutura dos dados de subsolo a partir da prospecção geológica, tendo como objetivo primário obter uma seção geológica ou modelo 3D. Tal problema é extremamente suscetível a aplicação de AGs pois sua função objetivo é extremamente irregular, não linear, possui muitos mínimos e máximos locais, podendo apresentar descontinuidades.

2. Música

Foi apresentado em 1999, na *CEC99 – IEEE – International Conference on Evolutionary Computation*, um ambiente interativo utilizando AGs para a avaliação de músicas (seqüências de acordes) tocadas em arquivos MIDI. No caso, os indivíduos da população foram definidos em grupos de quatro vozes (soprano, contralto, tenor e baixo) ou coros. Cada um é avaliado segundo três critérios: melodia, harmonia e oitavas. A composição destes três critérios definia a aptidão (fitness) definida pela função de seleção, que retorna o melhor indivíduo ou melhor coro. Um ciclo genético cria novos indivíduos a partir dos anteriores e procura sempre pelo melhor. Quando um novo grupo é selecionado, ele é tocado em MIDI. A duração do ciclo genético determina o ritmo da evolução. O sistema criado foi denominado Vox Populi.

3. Telecomunicações

Blanchard (1994), no *WCCI'94 – World Congress on Computational Intelligence* – ocorrido em Orlando, na Flórida, mostrou uma série de soluções promissoras a situações reais utilizando AGs, dentre as quais o caso da *US West*, uma companhia regional de telecomunicações do estado do Colorado, que vem usando um sistema baseado em AGs que possibilita projetar, em duas horas, redes óticas especializadas, trabalho que levaria seis meses utilizando especialistas humanos. O sistema produz resultados ainda 10%

melhores que os realizados pelo homem. A companhia estimava, naquele momento, que o sistema possibilitaria uma economia de 100 milhões de dólares até o final do século.

4. Médica

No Hospital universitário da UFSC (1999), em Florianópolis, os AGs foram utilizados para auxiliar na elaboração de uma escala de trabalho dos médicos plantonistas neonatal da maternidade. O objetivo pretendido foi o de auxiliar na solução da escala de trabalho dos médicos, bem como diminuir o esforço e o desgaste humanos para a confecção do plantão. O problema resumia-se na disponibilidade de 12 médicos e na necessidade de atendimento 24 horas por dia, tendo-se como variáveis envolvidas o número de médicos contratados e o turno com número adaptável de horas. O questionamento apresentava ainda todo o conjunto de restrições de trabalho, como cargas horárias, turnos de trabalho, plantões noturnos e diurnos, finais de semana e feriados, número máximo de horas de trabalho consecutivas, períodos específicos de possibilidade de trabalho, horários fixos para determinados médicos e cargas horárias variáveis entre os médicos, podendo inclusive haver mudança nas variáveis todos os meses.

O grupo de pesquisa observou em um primeiro momento que a função aptidão (*fitness*) precisava ser refinada, pois os resultados obtidos não eram satisfatórios. Na análise seguinte, após o refinamento dos dados obteve-se grande melhoria, com 11 médicos satisfeitos e 1 insatisfeito, com 20 horas a mais, resultando num melhor aproveitamento no que diz respeito a função funcionário/horas trabalhadas.

Outros tipos de AG

Os conceitos apresentados anteriormente se referem aos AGs simples. Existem outros que foram desenvolvidos para problemas mais específicos [10].

Genitor

Genitor (Whitley 1988; 1989) é um algoritmo cujos melhores pontos encontrados são preservados na população. Este procedimento é denominado de *elitismo.* Na prática isto resulta numa busca mais agressiva, que na prática é geralmente bastante efetiva. No entanto existe o perigo de uma convergência prematura para mínimos locais. Cada indivíduo selecionado e cruzado com seu parceiro é colocado no lugar do pior indivíduo da população anterior. A aptidão é atribuída de acordo com um "ranking", ou seja, a aptidão de cada indivíduo assume valores discretos.

CHC

Outro AG que coleciona os melhores indivíduos da população atual é o CHC (*Cross generational elitist selection, heterogeneous recombination and Cataclysmic mutation*).

Após o cruzamento, feito aleatoriamente, os N melhores indivíduos são coletados levando-se em consideração a população atual e a população gerada após o cruzamento. Remove-se os indivíduos duplicados. Este método impõe uma busca mais agressiva, assim como no Genitor. Repare que a seleção está implícita no algoritmo, a partir do momento que escolhe os melhores indivíduos de cada população (anterior e

atual). Normalmente usa-se populações pequenas, com 50 indivíduos, por exemplo. O ponto de cross-over é sempre a metade do cromossomo. Para se solucionar o problema de convergência prematura para mínimos locais é utilizada uma alta taxa de mutação, sempre preservando o melhor indivíduo da população. A partir da primeira seleção aleatória, utiliza-se o cross-over diretamente nas populações subsequentes.

Algoritmos Híbridos

Muitos autores consideram que os AGs nem sempre são a melhor solução para problemas de otimização específicos. Os algoritmos genéticos na sua forma original utilizam normalmente uma representação binária para codificar uma solução de um problema na estrutura de um cromossomo, que nem sempre é a mais eficiente. Além disso, o processo de seleção de cromossomos reprodutores, o cruzamento e as mutações são muitas vezes aleatórios. O critério de sobrevivência de cromossomos é definido a partir de uma função que mede o nível de aptidão de cada cromossomo.

Os mecanismos assim definidos não são eficientes na solução de diversos problemas complexos de otimização combinatória, em parte pelas próprias deficiências no processo de busca causadas pela escolha aleatória de reprodutores e pela formação prematura de uma população homogênea causada pelo critério de escolha dos cromossomos sobreviventes. Com isso, apesar do seu caráter genérico, o desempenho dos AGs 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 de, nos últimos anos, diversos pesquisadores terem proposto modificações nos AGs convencionais, incorporando técnicas de busca local, ou ferramentas de outras metaheurísticas tais como *Simulated Annealing*, *Tabu Search*, *Scatter Search*, entre outros, ou desenvolverem versões paralelas para melhorar o desempenho dos AGs.

Com isso procura-se um certo equilíbrio entre o caráter genérico e simplista dos AGs convencionais com AGs mais especializados e sofisticados encontrados nas versões híbridas (algoritmos genéticos não convencionais). Dentre as técnicas híbridas destaca-se a reunião de AGs com *Tabu Search* (TS) e *Scatter Search* (SS).

O primeiro tem se mostrado como uma das melhores opções para a solução aproximada de problemas de elevado nível de complexidade, principalmente na área de Otimização. *Tabu Search* utiliza basicamente: listas de informações para proibir a análise de soluções geradas anteriormente durante um certo número de iterações, além de trabalhar com ferramentas de intensificação de buscas em regiões mais promissoras e diversificação.

Já *Scatter Search* (SS) pode ser visto como uma versão determinística dos AGs, incorporando além disso conceitos de TS. No SS, também são geradas populações de soluções como nos AGs, mas as etapas de seleção de cromossomos pais e de reprodução são efetuadas através de critérios determinísticos. A geração de novas soluções a partir de informações de um conjunto de soluções atuais é feita através de combinações lineares das soluções atuais. Diversos trabalhos publicados recentemente têm mostrado a superioridade desta técnica quando comparada com outras metaheurísticas, incluindo AGs convencionais.

Conclusão

Os algoritmos genéticos buscam soluções de forma análoga ao processo de evolução dos seres vivos e podem ser utilizados para resolver um grande número de problemas de otimização complexos. Eles são apropriados quando há muitas variáveis envolvidas e um espaço de soluções de dimensão elevada. Embora sejam mais lentos, o tempo de execução torna-se um problema cada vez mais secundário devido à rápida evolução dos sistemas computacionais.

O controle sobre os parâmetros do algoritmo é de fundamental importância para uma convergência rápida. Para problemas específicos é aconselhável a utilização de algoritmos híbridos, que misturam as técnicas dos AGs com métodos de otimização tradicionais e/ou outras metaheurísticas.

Referências

[1] Uma introdução aos algoritmos genéticos (parte 1)
http://www.computacao.gigamundo.com/2009/04/15/uma-introducao-aos-algoritmos-geneticos-parte-1

[2] Uma introdução aos algoritmos genéticos (parte 2)
http://www.computacao.gigamundo.com/2009/04/15/uma-introducao-aos-algoritmos-geneticos-parte-2

[3] Uma introdução aos algoritmos genéticos (parte 3)
http://www.computacao.gigamundo.com/2009/04/22/uma-introducao-aos-algoritmos-geneticos-parte-3

[4] Obitko, Marek – Genetic Algorithms
http://www.obitko.com/tutorials/genetic-algorithms/

[5] Ochi, Luiz Satoru – Algoritmos Genéticos: Origem e Evolução
http://www.sbmac.org.br/bol/bol-2/artigos/satoru/satoru.html

[6] Algoritmos Genéticos
http://www.deti.ufc.br/~pimentel/disciplinas/ica_files/Documentos/Algorimos_Geneticos.pdf

[7] Marques, Artur – Introdução aos Algoritmos Genéticos
http://arturmarques.com/mcei/mcei_introducao_aos_algoritmos_geneticos.pdf

[8] Filitto, Danilo – Algoritmos Genéticos: Uma visão explanatória
http://www.uniesp.edu.br/revista/revista6/pdf/13.pdf

[9] Algoritmos Genéticos
http://www.icmc.usp.br/~andre/research/genetic/index.htm

[10] Miranda, Marcio Nunes de – Algoritmos Genéticos: Fundamentos e Aplicações
http://www.gta.ufrj.br/~marcio/genetic.html

[11] Genetic Algorithms
http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/tcw2/report.html

[12] Wall, Matthew – Introduction to Genetic Algorithms
http://lancet.mit.edu/~mbwall/presentations/IntroToGAs/

[13] Rennard, Jean-Philippe – Introduction to Genetic Algorithm
http://www.rennard.org/alife/english/gavintrgb.html

[14] Skinner, Michael – Genetic Algorithms Overview
http://geneticalgorithms.ai-depot.com/Tutorial/Overview.html

[15] Gero, John S.; Kazakov, Vladimir – Machine Learning in Design Genetic Engineering-Based Genetic Algorithms
http://www.cad.strath.ac.uk/~alex/AID/AID00/Gero.pdf

[16] Algoritmos Genéticos
http://www.comp.ita.br/~carlos/texts/8-AlgoritmosGeneticos.pdf

[17] Gorni, Henrique – Algoritmos Genéticos – Visão Geral
http://asylum.umfiloqualquer.org/2009/05/20/algoritmos-geneticos-visao-geral/

[18] Yamamoto, Lia; Wilhelm, Volmir Eugenio – Uso de simulated annealing e algoritmo genético no problema da reconfiguração de uma rede de distribuição de energia elétrica
http://dspace.c3sl.ufpr.br/dspace/handle/1884/1106

[19] Pinho, Alexandre Ferreira de; Montevechi, José Arnaldo Barra; Marins, Fernando Augusto Silva – Análise da aplicação de projetos de experimentos nos parâmetros dos algoritmos genéticos
www.latec.uff.br/sg/arevista/Volume2/Numero3/SG091.pdf

[20] Soares, Gustavo Luís - Algoritmos Genéticos: Estudo, Novas Técnicas e Aplicações

[21] Sobrinho, Antonio Carlos; Girardi, Maria Del Rosário – Uma análise das aplicações dos Algoritmos Genéticos em Sistemas de Acesso à Informação Personalizada

[22] Redusino, Augusto Cesar – Aplicações de Algoritmos Genéticos