UNIVERSIDADE VEIGA DE ALMEIDA – UVA

BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

INTELIGÊNCIA COMPUTACIONAL: APLICAÇÕES DO ALGORITMO GENÉTICO

HEYDER PESTANA DIAS

RIO DE JANEIRO

2015

UNIVERSIDADE VEIGA DE ALMEIDA - UVA

HEYDER PESTANA DIAS

Monografia apresentada ao curso de Ciência da Computação da Universidade Veiga de Almeida, como requisito parcial para obtenção do título de bacharel em Ciência da Computação.

Orientador: Prof. Jobson Luiz Massollar da Silva

INTELIGÊNCIA COMPUTACIONAL: APLICAÇÕES DO ALGORITMO GENÉTICO

RIO DE JANEIRO

2015

UNIVERSIDADE VEIGA DE ALMEIDA - UVA

BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

HEYDER PESTANA DIAS

INTELIGÊNCIA COMPUTACIONAL: APLICAÇÕES DO ALGORITMO GENÉTICO

Monografia apresentada como requisito final à conclusão do curso em Bacharel em Ciência da Computação.

APROVADA EM: 22/06/2015

CONCEITO: _____

BANCA EXAMINADORA

_____

PROF. JOBSON LUIZ MASSOLLAR DA SILVA

ORIENTADOR

_____

PROF. CARLOS ALBERTO ALVES LEMOS

_____

PROF. MIGUEL ANGELO ZACCUR DE FIGUEIREDO

Coordenação de Ciência da Computação

Rio de Janeiro

AGRADECIMENTOS

Agradeço a minha família e a todos que me auxiliaram neste trabalho.

Também a todos que participaram e auxiliaram em todos estes anos de estudo e trabalho, para atingir a minha formação.

A aqueles que simplesmente me influenciaram com suas ideias, assim auxiliando a tomar as decisões até hoje.

RESUMO

O crescimento da necessidade de sistemas computacionais em diversas áreas do desenvolvimento humano aumenta a demanda por computadores e redes mais robustos e mais inteligentes. Entretanto, além do hardware também é necessária a evolução do software, como por exemplo, sistemas para reconhecimento de imagem, para distribuição de trafego de rede ou para pesquisa na internet. Uma das formas de aumentar a inteligência dos sistemas está na utilização de técnicas de inteligência computacional. Assim, se faz necessário um estudo mais aprofundado dessa área e do uso de algoritmos que se adaptam com o objetivo de resolver um problema específico. Esta monografia tem como objetivo analisar o uso do algoritmo de programação genética em diferentes casos de uso como uma das soluções viáveis para aplicações que necessitam adotar técnicas de inteligência computacional.

Palavras-chave: algoritmo genético, inteligência computacional, computação evolutiva.

ABSTRACT

The growing need for computer systems in several areas of human development increases the demand for computers and more robust and smarter networks. However, in addition to hardware it is also required the development of software, such as systems for image recognition, for network traffic distribution or search the Internet. One way to increase the intelligence of the systems is exploring computational intelligence techniques. Thus, it is necessary a further investigation of this area and the use of algorithms that adapt themselves aiming at solving a particular problem. This paper aims to analyze the use of genetic programming algorithm in different use cases as a viable solutions for applications that need to adopt computational intelligence techniques.

Keywords: genetic algorithm, computational intelligence, evaluative computing.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AG – Algoritmo Genético

ABL - Algoritmo de Busca Linear

SUMÁRIO

1 INTRODUÇÃO

A computação já está integrada ao desenvolvimento humano, sendo utilizada tanto no cotidiano como em pesquisas avançadas. Não é mais possível ignorar a influência destas tecnologias em nossas vidas, pois ela está em toda parte, criando a cada dia uma forma mais eficiente de solucionar nossos problemas.

Alguém poderia dizer que os problemas intratáveis são raríssimos e de pouco interesse prático. Entretanto, isso infelizmente não é verdade. Os problemas de complexidade não polinomial (intratáveis) permeiam a nossa vida (LINDEN, 2008).

Quando se discute algoritmos é importante avaliar se a solução proposta é viável. Entretanto, comparar algoritmos levando em consideração CPU, SO, memória, compilador, etc., tornaria essa comparação impraticável. Ao invés disso, é verificado o tempo de execução conforme o aumento do volume de informação na entrada. Desta forma, calculamos o tempo aproximado de execução de um dado algoritmo, levando em conta que todas as instruções fundamentais levam o mesmo tempo de execução. Considerando o pior caso de execução de cada algoritmo, chegamos a uma função que representa seu desempenho em relação à entrada de dados, definindo um limite superior para a mesma.

Essa é a notação O, que define que para uma função f(n) existe uma O(g(n)), de forma que o crescimento de f(n) não é maior que g(n). Esta métrica é importante para tornar a comparação entre algoritmos uniforme e permitir a comparação de dois algoritmos distintos que solucionem o mesmo caso (LINDEN, 2008).

Contudo, é importante verificar que existem problemas intratáveis, cujo tempo necessário para resolvê-lo é considerado inaceitável. Como pode ser verificado na Tabela 1, à medida que temos crescimento da entrada de dados na função, podemos verificar que algoritmos com complexidade n! tornam-se rapidamente intratáveis.

Tabela 1 – Comparativa de complexidade com o crescimento de entrada de dados.

n	**n^2**	**n^3**	**2^n**	**n!**
10	10^2	10^3	$\cong 10^3$	$\cong 10^6$
100	10^4	10^6	$\cong 10^{30}$	$\cong 10^{158}$
1000	10^6	10^9	$\cong 10^{300}$	$>>10^{1500}$

Como exemplos de problemas intratáveis é possível citar:

- Encontrar o melhor roteamento de pacote de dados em uma rede.
- Traçar rota de entrega de mercadorias que demandem menor tempo e/ou distância.
- Escolha de turmas e horários que use o máximo de salas de aula de forma eficiente.

Em geral, estes problemas podem ser representados como grafos ou combinacionais de tipos diversos. Para esses problemas especiais foram desenvolvidos vários algoritmos inteligentes, muitos destes aplicando técnicas de resolução de problemas inspiradas na natureza. Como exemplo temos redes neurais, lógicas nebulosas e algoritmos genéticos, tentando deste modo imitar a natureza e conseguir formas mais inteligentes e adaptáveis de solução de problemas. Isto não significa que essas técnicas solucionaram o problema, mas que criaram artificios que tornaram aceitável o tempo necessário para chegar a uma solução.

Portanto, não se trata de inserir os dados em uma máquina que simplesmente irá retornar os resultados. Existe uma variedade de conhecimentos específicos necessários para tentar desenvolver soluções viáveis para os problemas intratáveis, dificultando muito o seu processo de definição e otimização. Com a aplicação de algoritmos convencionais, ou seja, sem a aplicação de técnicas de inteligência computacional, demoraria um tempo demasiadamente longo para tingir uma solução viável. Este trabalho apresenta o Algoritmo Genético como uma solução viável para substituir o uso das técnicas convencionais nos casos considerados intratáveis.

1.1 OBJETIVO

Esse trabalho apresenta os fundamentos da aplicação do Algoritmo Genético e seus principais componentes, detalhando a funcionalidade do mesmo. Também apresenta alguns casos de especialização do uso desta técnica, onde será discutido como devem ser tratados os problemas mais comuns enfrentados no uso deste algoritmo.

Também serão apresentados dois casos reais de uso deste algoritmo e seus resultados, exemplificando assim a diversidade de problemas nos quais o AG consegue atuar. Por fim, será apresentado um estudo de caso de comparação do desempenho desta técnica com um algoritmo convencional.

Assim, o principal objetivo deste trabalho é mostrar o uso de algoritmos genéticos, apresentando sua capacidade de tratamento de problemas diversos com alta qualidade de

resposta em tempo aceitável, tentando, assim, mostrar o quanto a aplicação desse algoritmo, se utilizado da forma adequada, pode superar outros tipos de algoritmos desenvolvidos com técnicas convencionais.

1.2 ORGANIZAÇÃO DO TRABALHO

Esta monografia é composta de cinco capítulos, organizados da seguinte forma:

- Capítulo 2: apresenta as definições do AG e sua estrutura.
- Capítulo 3: apresenta aplicações do AG.
- Capítulo 4: apresenta o estudo de caso utilizando AG.
- Capítulo 5: apresenta a conclusão.

2 ALGORITMOS GENÉTICOS

2.1 UMA BREVE HISTÓRIA DA GENÉTICA

Os principais conceitos aplicados no algoritmo genético (AG) são os fundamentos Teoria da Evolução e Seleção Natural (DARWIN, 1859), ou seja, este algoritmo simula de forma simplificada a evolução.

> Quando um mesmo órgão se encontra em muitos indivíduos da mesma classe, sobretudo nos indivíduos tendo hábitos de vida muito diferentes, podemos ordinariamente atribuir este órgão a um antepassado comum que o transmitisse por hereditariedade aos descendentes; podemos, além disso, atribuir a sua falta, em alguns indivíduos da mesma classe, a um desaparecimento provindo do não uso ou da ação da seleção natural. (DARWIN, 1859)

Segundo a Teoria da Evolução, as espécies atuais são descendem de outras espécies que sofreram modificações. Seus ancestrais são descendentes de predecessores que também diferem deles, e por um processo continuo, que origina em organismos precursores, desconhecidos e extremamente primitivos. Darwin relacionava as variações das gerações com a reprodução das especiais sob as condições dispostas pelo ambiente. Desde modo propiciando o que chamou de seleção natural. (Darwin, 1859)

> Numerosos experimentos têm demonstrado que se duas plantas que diferem constantemente em um ou diversos caracteres são cruzadas, os caracteres comuns a ambas são transmitidos sem mudanças para os híbridos e sua descendência, mas cada par de caracteres diferentes se une no híbrido para formar um novo caráter, o qual é geralmente variável na descendência deste híbrido. (MENDEL, 1865)

Foi Gregor Mendel que descreveu que o processo de transmissão de características estava ligado a uma unidade básica de informação. Utilizando experimentos com cruzamentos de plantas, como a hibridação de plantas, propôs a existência de características a partir da existência de unidades elementares de hereditariedade, atualmente conhecido como gene.

Entretanto, foram Francis Crick e James Watson que desvendaram a verdadeira forma do DNA e sua estrutura, detalhando os seus processos de funcionamento a nível molecular. Senso assim, todo individuo animal, vegetal ou até mesmo vírus, possui uma cópia completa que descreve o organismo chamado de genoma.

2.2 ORIGEM DO ALGORITMO GENÉTICO

Alan Turing, propôs “genetical and evolutionary search” como uma solução de automatização para problemas. (TURING, 1950). Foram usados conceitos de mutação, hereditariedade e seleção natural para escolher a melhor solução para um problema. Entretanto, somente em 1962, que Bremermann fez experimentos reais de otimização utilizando evolução e recombinação como princípios. (BREMERMANN, 1962). Posteriormente, vários pesquisadores utilizaram este mesmo princípio implementando algumas modificações. Rechenberg e Schwefel trabalharam em estratégias evolutivas (RECHENBERG, 1965). Fogel, Owens e Walsh definiram a programação evolucionária (FOGEL, 1966), e por fim, John Henry Holland desenvolveram o algoritmo genético (HOLLAND, 1975).

Sendo assim, o AG não foi o primeiro algoritmo a utilizar conceitos evolucionários como base para o desenvolvimento de um processo de busca de otimização de solução. Este tipo de algoritmo faz parte de um ramo de buscas chamado de “Técnicas Aleatórias-Guiadas”, ou seja, apesar de usar componentes aleatórios, usam informações do estado corrente para guiar os passos seguintes. Contudo, sua versatilidade e adaptabilidade a problemas que precisam percorrer vastas áreas de possíveis soluções é o que faz seu diferencial. (LINDEN, 2008)

> Os organismos vivos são solucionadores de problemas consumados. Eles exibem uma versatilidade que fazem dos melhores programas de computador uma vergonha. Esta observação é especialmente irritante para cientistas da computação, que podem passar meses ou anos de esforço intelectual em um algoritmo, enquanto organismos vêm por suas habilidades através do mecanismo aparentemente sem direção de evolução e seleção natural. (HOLLAND, 1998)

Os algoritmos genéticos são técnicas heurísticas de otimização global, ou seja, não necessariamente irão obter a solução ótima para um problema. E dificilmente conseguem repetir resultados obtidos em testes anteriores, mesmo utilizando os mesmos parâmetros de partida. Esse é um algoritmo extremamente eficiente para percorrer vastas áreas de um espaço de soluções para encontrar uma solução próxima a solução ótima. Logo, eles são mais indicados para problemas especialmente difíceis, como os denominados NP difíceis.

"Algoritmos genéticos tornam possível explorar uma gama muito maior de possíveis soluções para um problema do que fazem programas convencionais." (HOLLAND, 1998)

2.3 CARACTERÍSTICAS DO ALGORITMO GENÉTICO

O funcionamento do AG apresenta semelhança a maioria dos algoritmos evolutivos. Que pertencem a classe de métodos probabilísticos de busca e otimização, embora utilize técnicas aleatórias, este utiliza informações do estado corrente para guiar para o seguinte passo.

Características primárias:

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

2.4 ESTRUTURA BÁSICA DO ALGORITMO

A estrutura simples de um algoritmo genético tem algumas partes obrigatórias, e estes itens e suas disposições são necessárias para o funcionamento da técnica. Existem diversas formas de refinamentos em algumas das partes do diagrama da Figura 1, mas essencialmente sempre respeitam esta formação.

O primeiro passo é a criação da primeira geração, a partir da qual é calculada a aptidão dos indivíduos, que pode servir de parâmetro para que a condição de término seja ou não satisfeita, uma vez que podemos utilizar outros parâmetros como condição de término, como por exemplo a quantidades máxima de gerações. Caso não seja, será feita uma seleção dos indivíduos, que participarão da criação de uma nova geração. O processo se repete até que a condição de termino seja satisfeita.

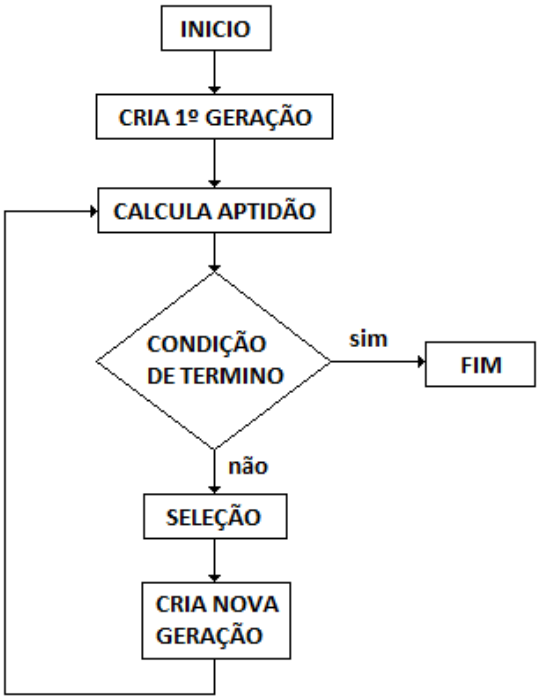

Figura 1 – Diagrama de blocos de AG simples.

2.5 REPRESENTAÇÃO CROMOSSÔMICA

Como sua criação é influenciada também pela genética, este algoritmo utiliza o conceito de cromossomo como forma de expressar a passagem de uma capacidade de um indivíduo pai para o filho, fornecendo à geração seguinte a capacidade de aperfeiçoar o desempenho atingindo pela geração anterior.

Cada indivíduo é constituído por um ou mais cromossomos concatenados, onde cada cromossomo é referente a uma variável do problema proposto. Como é possível ver no exemplo da Figura 2, os cromossomos são como vetores de genes, e por sua vez os genes são seleções de possíveis valores de alelos. Os algoritmos genéticos mais simples costumam utilizar a representação cromossômica como um vetor binário, onde cada bit é um gene e os valores possíveis 0 e 1 são os alelos, e a cadeia completa de bit’s é o cromossomo. Porém, existem também as representações em valores reais, estrutura de dados como árvores e também lógica Fuzzy.

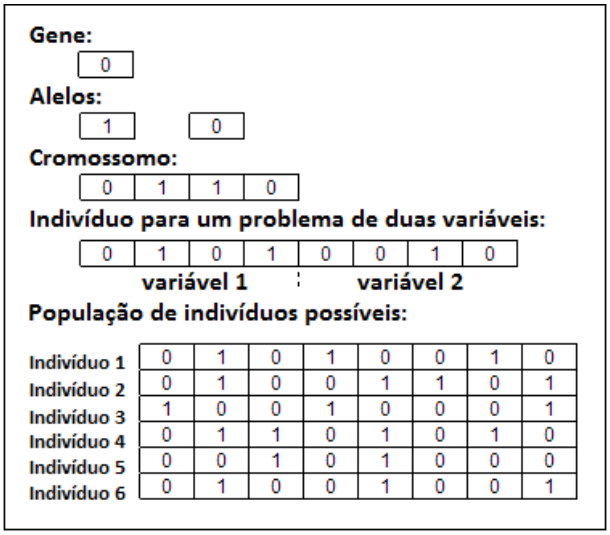

Figura 2 – Exemplo e definição de cromossomo.

> Reformulação na linguagem dos algoritmos genéticos, a busca de uma boa solução para um problema é uma busca para determinadas cadeias binárias. O universo de todas as cadeias possíveis pode ser considerado como uma paisagem imaginária; vales marcar a localização de sequências que codificam soluções pobres, e ponto mais alto da paisagem corresponde à melhor sequência possível. (HOLLAND, 1998)

Portanto, definir como será a representação cromossômica para a área de busca é um fator muito importante, já que nem sempre é viável ou aceitável mapear todas as soluções possíveis. Para isso é necessário um estudo detalhado do problema antes de propor o cromossomo, de modo a viabilizar e otimizar a solução final. Caso isso seja mal planejado, o universo de soluções, como dito por Holland, pode acabar não representando adequadamente o espaço real de busca do problema.

Nos AG's assume-se que cada indivíduo terá um único cromossomo e a sequência do cromossomo estará ligada diretamente com o desempenho do indivíduo. Geralmente é utilizada uma população de número fixo de indivíduos, também com cromossomos de tamanho constante. Contudo existem casos em que o cromossomo é representado por uma àrvore de valores, mudando assim as limitações e formas de atuação dos mesmos, mas não a lógica empregada pelo algoritmo.

Portanto, o primeiro passo para utilização do AG, é a definição das variáveis necessárias para executar a tarefa proposta para solucionar o problema, definindo, assim, a quantidade de variáveis e a representação adequada a ser utilizada. Esses valores podem ser reais, bits, números inteiros ou outras estruturas de dados, e também deve ser definido o valor máximo e mínimo permitido para estas variáveis, delimitando o algoritmo a uma área de atuação e uma granularidade.

Assim, é gerado um alfabeto de possíveis valores para o gene, onde cada elemento é equivalente a um alelo e cada formação de cromossomo é um possível indivíduo que representa uma possível solução no espaço de busca da solução do problema.

2.6 PRIMEIRA GERAÇÃO

Neste passo é gerado um conjunto de indivíduos que constituíram a primeira geração, e a cada iteração é desenvolvida uma nova geração, embora nem todos os indivíduos de uma população sejam respectivamente filhos da geração anterior. Isso ocorre porque a reprodução depende da técnica escolhida, que pode ser desde substituição total dos indivíduos a cada geração ou simplesmente parcial.

Em um AG simples, a população inicial terá uma quantidade predefinida de indivíduos que serão gerados aleatoriamente por um processo heurístico. Como não há evolução sem variedade, necessitamos de indivíduos diferentes para assim termos adaptações diferentes ao problema. Deste modo, é importante que a população inicial cubra a maior área possível do espaço de busca, permitindo uma maior probabilidade de reduzir o número de iterações necessárias para encontrar a solução. Da mesma forma a variedade irá proporcionar diferentes direções para a solução do problema, permitindo uma atuação mais eficiente do algoritmo.

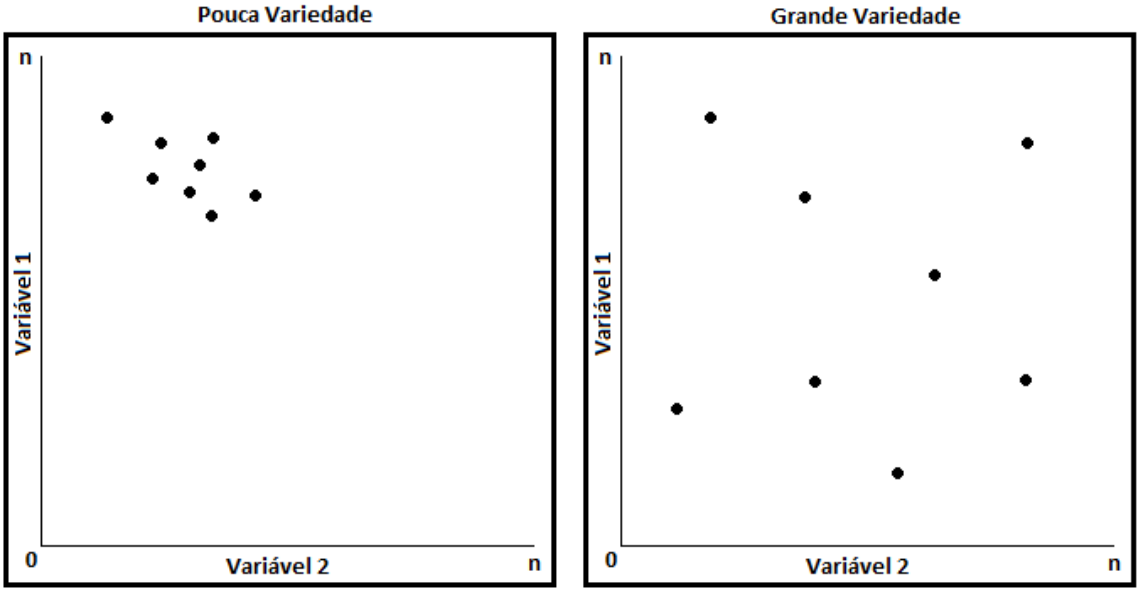

Figura 3 – Comparação de variedade na distribuição da 1º população.

A Figura 3 apresenta um exemplo de uma área de busca que é representada por um cromossomo de duas variáveis de N valores possíveis. Se todos os indivíduos da primeira geração estiverem muito próximos, ou seja, pouca variedade genética, isso não

necessariamente irá impedir que o AG consiga executar a busca pelo valor ótimo, mas poderá provocar uma maior necessidade de iterações para encontrar esse valor, uma vez que terá uma baixa variedade genética no cruzamento dos indivíduos. Dessa fora, fica muito dependente que a taxa de mutação leve as gerações seguintes para a partes mais distantes do ponto de partida, o que não deve ser o único fator levado em consideração para busca do valor ótimo.

2.7 AVALIAÇÃO DO INDIVÍDUO

Para avaliar a capacidade de um indivíduo ser uma boa solução para o problema é necessário a criação da função de avaliação da população, ou seja, essa é uma a forma de verificar o quão bem adaptado é determinado indivíduo. Essa será a peça chave para definir as chances do cruzamento de indivíduos e para criação das próximas gerações. Esta função irá utilizar os parâmetros do cromossomo do indivíduo, aplicados na função de avaliação do problema, em que seu resultado é chamado de aptidão, nomeado por Holland como *fitness*. Portanto a função ira retornar a qualidade de cada individua para resolver o problema proposto.

Todos os indivíduos criados serão testados por esta função, que determinará o *fitness* de cada indivíduo. Visto que esta informação será utilizada pelo AG para dirigir a evolução dos indivíduos até a melhor solução possível, então ela deve ser definida com cautela, pois pode direcionar as iterações para longe da otimização do problema ou para um resultado inferior ao melhor possível no espaço de busca.

Um exemplo de função *fitness* pode ser visualizado no caso de um AG que precisa encontrar todos os pontos zeros da função $f(x) = x^3+x^2$. Como a distância do zero para um valor negativo ou positivo tem a mesma relevância para o AG, então a melhor função para servir de *fitness* seria o modulo desta função, como apresentado na Figura 4.

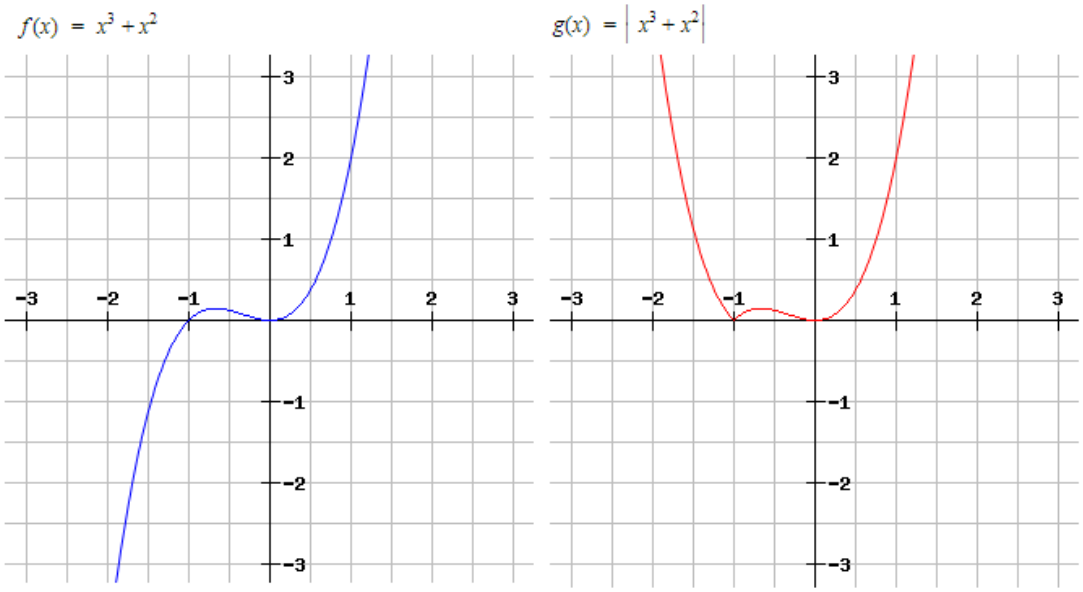

Figura 4 – Gráfico de função fitness X equação do problema.

Portanto, todos os indivíduos mais próximos de zero na função *fitness* seriam os mais bem adaptados. Também é possível inverter a função de *fitness*, tornando negativo o resultado do módulo da função, para permanecer com a condição de maior *fitness* é igual ao melhor resultado ao tornar valores mais distantes de zero inferiores a zero, mas isto não é uma regra. Verifica-se, então, que não existe um padrão para esta função, ou seja, cada problema terá uma forma diferente de tratar a melhor solução de um determinado indivíduo.

2.8 SELEÇÃO NATURAL NO ALGORITMO

A seleção é o mecanismo que simula o processo de seleção natural e é feita de acordo com o valor de *fitness*. Nesse caso, indivíduos não selecionados serão extintos dependendo da técnica de reprodução escolhida. Dentre os escolhidos, serão feitas recombinações cromossômicas para a geração de novos indivíduos, que podem substituir totalmente ou parcialmente a antiga geração. Deste modo, a cada geração teremos em média um grupo mais bem adaptado à otimização do problema em comparação com as gerações anteriores.

> “Os filhos não substituem os pais; em vez disso, substituem aqueles de baixo "fitness", que são descartados a cada geração, para que a população total continue a ser o mesmo tamanho” (HOLLAND, 1998)

Portanto, indivíduos com baixo *fitness* terão alta probabilidade de desaparecerem, e indivíduos com alto *fitness* terão alta probabilidade de permanência nas gerações futuras, como é possível observar na Figura 5 onde na 3º geração um individuo criado é extinto na mesma geração.

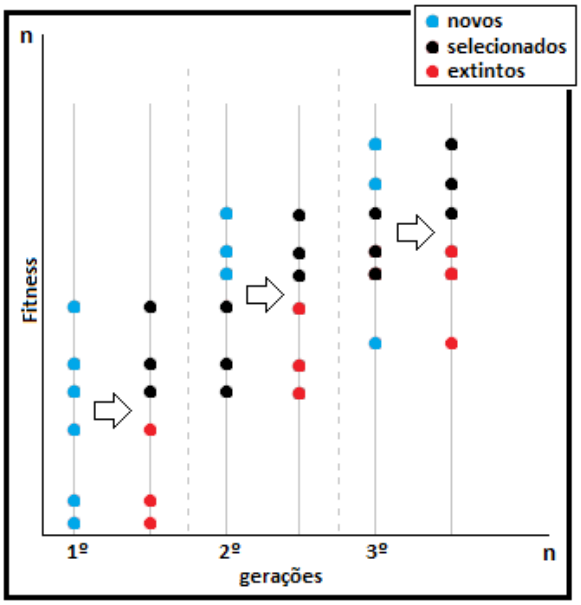

Figura 5 – Comparação seleção de indivíduos a cada geração.

2.8.1 Roleta Russa

> "Nesta técnica é atribuída a cada indivíduo uma probabilidade de passar para a próxima geração. Essa probabilidade é proporcional a sua adaptação ao ambiente, em relação à somatória da adaptação de todos os indivíduos, sendo maior a probabilidade dos indivíduos mais adaptados serem sorteados" (ZUBEN, 2000).

Esse é um método de seleção de um grupo de indivíduos a partir de seu grau de adequação com o sistema, como exemplificado na Tabela 1. O fitness de cada indivíduo da geração atual é transformado em um número de graus de um círculo usando a fórmula:

$$f(x) = 360 \times (b_x / \sum_{i=1}^{n} b_i)$$

onde x é o indivíduo para o qual se deseja calcular a participação na roleta, bx é o seu valor de *fitness*, que é dividido pela soma de todos os *fitness* da geração atual. Então, a chance de sorteio para cada integrante é relativa a quantidade de espaço que existe na roleta, como mostra a Figura 6 que representa a roleta da Tabela 2. Deste modo, quanto maior *fitness* em relação ao grupo, maior é a chance de ser sorteado na roleta e, assim, participar do grupo que irá influenciar na criação da próxima geração.

Tabela 2 – Distribuição de chance por adequação.

N	Cromossomo	Fitness	Graus
1	010101110111010	8.0	137,143
2	011110111010111	6.0	102,857

3	010101111101010	3.0	51,429
4	010101110111010	2.0	34,286
5	011101110101010	1.0	17,143
6	111101110111110	1.0	17,143

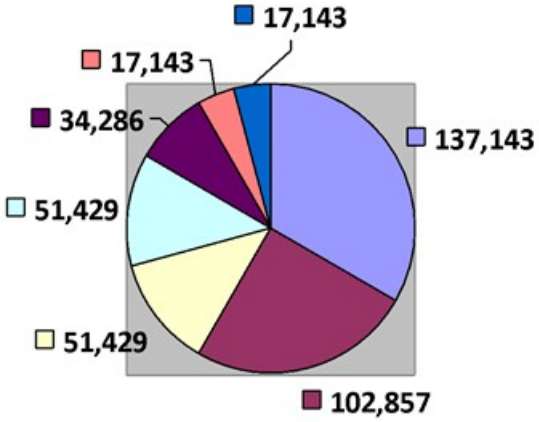

Figura 6 – Roleta Russa para Seleção de Grupo

2.8.2 Torneio

Este método consiste em selecionar aleatoriamente um conjunto de K indivíduos da população, com K sempre menor que o tamanho da população, onde dentre os selecionados será escolhido o indivíduo de maior *fitness*. O processo se repete até atingir a quantidade de pais necessária para o operador genético.

Com o uso deste método de seleção, temos um aumento na chance de seleção de pais com *fitness* inferiores, porém estes são importantes para uma boa variedade genética, uma vez que sem eles a população poderia convergir genericamente. Este aumento não significa diminuição significativa da chance da seleção dos indivíduos de melhor *fitness*, mas que, caso haja em uma seleção aleatória um grupo de baixo *fitness*, será selecionado o melhor individuo dentre eles, mesmo que este seja um inferior em comparação a população corrente.

2.8.3 Seleção Truncada

Esse método define um número de x% dos primeiros melhores indivíduos, ou seja, a população é ordenada de forma decrescente em relação ao *fitness* e são selecionados arbitrariamente os primeiros x% indivíduos encontrados.

Este é um dos métodos menos custosos em relação a tempo de processamento, porém existe o risco de menor variedade genética, que pode levar a uma rápida convergência genética. Logo, esse método depende muito da diversidade do grupo criado na primeira geração.Entretanto, esse efeito negativo pode ser evitado se não for utilizada uma faixa de

corte muito pequena, permitindo assim a entrada de pais piores no grupo dos selecionados, onde será aplicado o operador genético para gerar a população seguinte.

2.9 OPERADORES GENÉTICOS

O ponto chave do AG são os operadores genéticos, que criam novos indivíduos a partir dos pais selecionados. Existem vários métodos para a produção dos filhos e todos eles levam em consideração teorias genéticas. Os operadores podem utilizar dois ou mais indivíduos pais para a produção dos novos indivíduos filhos. Os operadores são definidos para os tipos de dados relacionados ao cromossomo, os seguintes citados são operadores binários, mas existem operadores para números reais e estrutura de dados como arvores.

2.9.1 *Crossover* ou Cruzamento de Um Ponto

> "O cruzamento consiste na manipulação do material genético existente na população e permitirá a criação de um ou mais indivíduos (filhos) dos indivíduos selecionados (pais) pela seleção. No final deste processo à população geralmente permanece do mesmo tamanho da população anterior" (GROSKO, 2006)

A recombinação é um processo sexuado, ou seja, envolve dois indivíduos, e emula o fenômeno *crossover*, que é a troca de fragmentos entre pares de cromossomos. De uma forma simples, trata-se de um processo que ocorre com probabilidade determinada no início da aplicação pelo usuário. Dentro dos indivíduos selecionados serão produzidos novos indivíduos, a partir do cruzamento dos cromossomos dos melhores indivíduos da geração atual, ou seja, os que tiveram melhor *fitness*.

A escolha do ponto de cruzamento do cromossomo pode ser aleatória, este ponto determina o corte igual a ambos os indivíduos selecionados, como apresentado na Figura 7. Que posteriormente serão recombinadas, produzindo no final duas novas versões dos cromossomos em comparação aos selecionados, a preservação ou não dos pais na geração seguinte é critério da população.

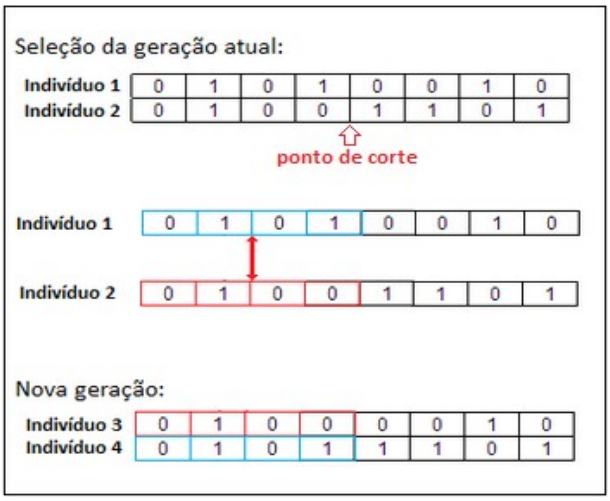

Figura 7 – Demonstração de cruzamento simples

2.9.2 *Crossover* de Dois Pontos

É semelhante ao operador anterior, mas se diferencia pelo fato de utilizar dois pontos de cortes aleatórios (ou não aleatórios) no cromossomo. Passa a ser necessário para cromossomos mais longos, uma vez que o resultado do cruzamento de um único ponto pode se gerar baixa produtividade na criação de indivíduos novos. Porém, mesmo o *crossover* de dois pontos tem seus limites, sendo assim em alguns casos especiais deve se utilizar um *crossover* mais complexo, deste modo assegurando um bom resultado na criação de novos indivíduos como mostra a Figura 8.

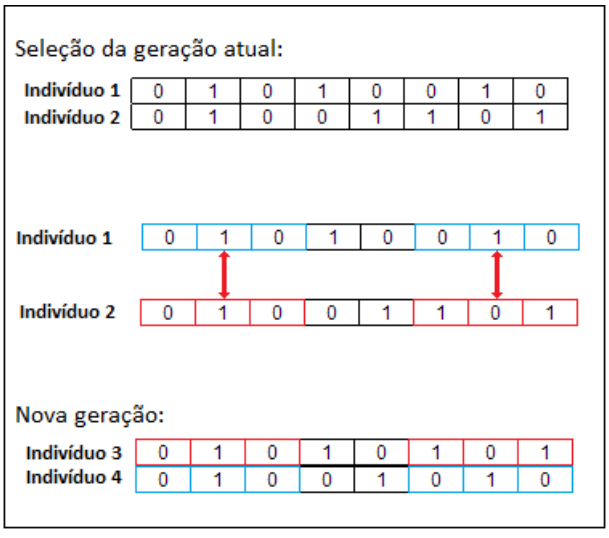

Figura 8 – Demonstração de cruzamento de dois pontos

2.9.3 *Crossover* Uniforme

Este é o operador genético de aplicação mais ampla, pois pode ser utilizado em qualquer esquema existente, lembrando que estes são operadores binários. A cada gene é

sorteado um número zero ou um, criando ao final uma máscara de que será aplicada como parâmetro para troca de genes dos pais, como é visto na Figura 9.

Fica claro que a máscara não pode ser feita somente por número um ou zero, pois perderia a propriedade necessária para o cruzamento. Então, é importante que o algoritmo que sorteia esta máscara trate esta condição como fora dos resultados possíveis.

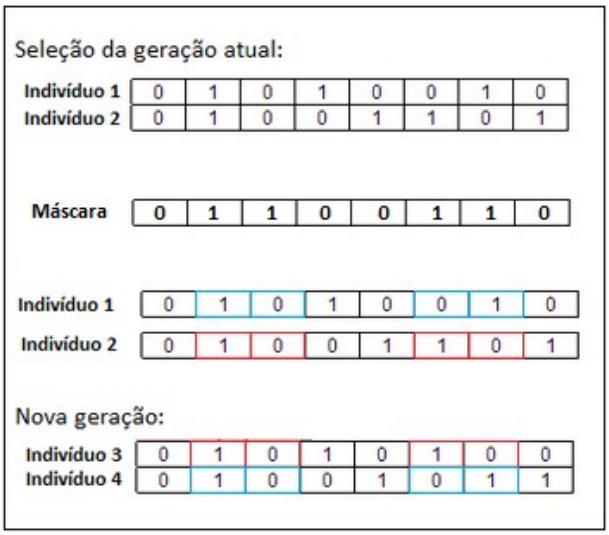

Figura 9 – Demonstração de cruzamento utilizando máscara

2.9.4 *Crossover* Baseado na Maioria

O principal resultado do uso deste operador genético é criar indivíduos que tenham semelhança com o conjunto de pais selecionados, deste modo a tendência é acelerar a convergência genética, por isso deve ser usado em momentos específicos. Por este motivo deve ser evitada nas primeiras gerações e utilizado próximo as ultimas gerações, permitindo especializar a busca da solução ao se aproximar do termino das iterações do algoritmo.

Como mostra a Figura 10, neste *crossover* são sorteados N pais, para N maior que dois e é testado cada gene de todos os pais. O valor que mais se repete será o valor do filho. Este operador gera somente um filho por operação.

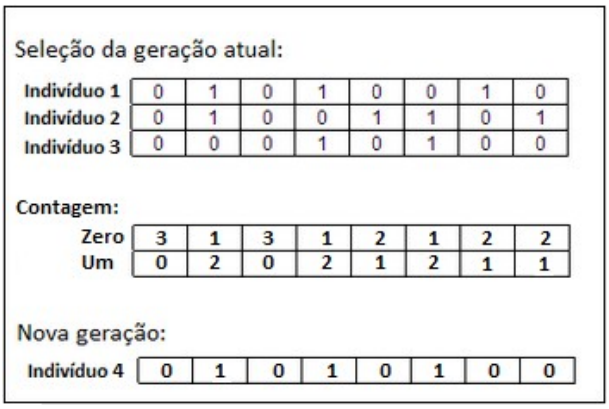

Figura 10 – Demonstração de cruzamento na maioria.

2.9.5 Mutação

> "Este mecanismo ajuda a solucionar o problema do confinamento a mínimos locais na otimização, pois promove alterações que direcionam a pesquisa para outros locais da superfície de resposta. [...] A mutação na representação binária é realizada pela troca de um por zero e vice-versa […]" (FILHO, 1998)

A mutação foi introduzida para aumentar a área de cobertura da população, impedindo, por exemplo, que as gerações futuras ficassem aprisionadas a um limitado grupo de melhores soluções. Este processo ocorre basicamente, selecionando uma posição aleatória de um cromossomo e mudando o seu valor para outro possível alelo.

A mutação pode ocorrer em um ou mais genes como visto na Figura 11, dependendo da configuração escolhida. Este operador evita a convergência genética, mas se utilizado indevidamente pode retardar a busca por gerar indivíduos nem sempre úteis à pesquisa, porque existe um risco de criar um novo individuo em qualquer área do campo de busca, tanto nas áreas de soluções desejadas como nas áreas de muito baixo *fitness*.

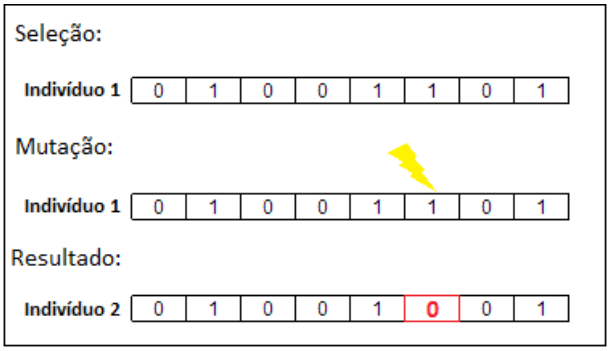

Figura 11 – Demonstração de mutação

2.9.6 Operadores com Probabilidade Variáveis

Até o momento foi apresentado o uso de operador genético com probabilidade fixa, porém esta pode ser alterada ao longo das gerações do AG, se desejado. Apesar das aplicações mais simples deste algoritmo levar em consideração uma chance fixa, determinada previamente a sua inicialização, ao utilizar operadores genéticos com chances diferentes a cada iteração das gerações é muito útil, uma vez que nas primeiras gerações geralmente se obtem uma grande diversidade genética e se procura fazer o maior número de *crossover* possíveis para refinar a busca no campo das soluções. Ao final, com a proximidade da convergência genética, se utiliza mais o operador de mutação para retardar o processo e aumentar novamente a variedade entre os cromossomos da população.

> "Depois de um grande número de gerações, ocorre a convergência genética, o que implica na pouca diversidade na população, tornando extremamente interessante que o operador de mutação fosse escolhido mais frequentemente do que o operador de crossover, para que possamos reinserir a diversidade genética dentro da nossa população" (LINDEN, 2008)

2.10 POPULAÇÃO

Este é a quantidade limite de indivíduos presentes em cada geração, podemos a cada geração substituir todos ou seguir regras mais refinadas que escolhem determinados pais para permanecerem nas gerações futuras. Este parâmetro é muito importante e deve ser definido com cautela, já que uma população muito pequena reduz a variedade genética e uma população muito grande ira tornar o algoritmo muito lento além de se aproximar de uma técnica de busca exaustiva, o que sairia do objetivo do AG que é criar uma busca mais refinada e inteligente pela solução mais próxima da ótima.

A forma mais básica de se tratar uma população, é substituir ela totalmente a cada geração. Por exemplo, para uma população de tamanho N se utilizarmos um operador binário simples, vamos gerar dois filhos para cada dois pais selecionados. Portanto para permitir um tamanho constante da população seria necessário selecionar N/2 vezes os pais necessários para gerar uma nova população de N filhos.

2.10.1 Controle da População por Elitismo

Elitismos é uma modificação no modulo mais comum de população, que assegura que X melhores indivíduos de uma população devem permanecer vivos na geração seguinte. Permitindo assim que os melhores genes de uma geração passada estejam presentes em uma

atual, que estes pais somente iram morrer nesta nova geração se os novos filhos superarem eles na nova seleção dos novos X melhores indivíduos.

É implícito que a quantidade X de indivíduos selecionados deve ser menor que o tamanho da geração e que também seu número não pode ser tão pequeno que não proporcione grande influência nos resultados. Porém o elitismo aumenta o número máximo de gerações, já que ele retarda as alterações geradas a cada geração.

2.10.2 Steady State

Nesta técnica iremos mais uma vez modificar o modulo comum de população, onde ao invés de substituir total uma população iremos substituir uma pequena parcela desta considerada com *fitness* mais inferior. É semelhante ao elitismo, mas neste caso reduzimos o crescimento da nova geração a quantidade de indivíduos eliminados da geração anterior. Gera a necessidade de mais gerações para atingir o objetivo do AG, mas reduz a geração de filhos e por usa vez o processamento envolvido. Será definido um número X de indivíduos por geração, e serão eliminados, não são selecionados todos os indivíduos de baixo *fitness* e sim uma parcela de *fitness* inferior ao valor selecionado como corte.

É importante lembrar que a variedade genética é essencial para o AG, portanto em alguns momentos pode se desejar preservar os pais de baixo *fitness*, não todos eles mas pelo menos uma parte, pois nem sempre preservar somente os melhores indivíduos é a solução. Para isso podemos selecionar aleatoriamente o grupo que será eliminado ao invés de definir um ponto de corte baseado no *fitness*, o importante desta técnica é definir uma quantidade de indivíduos a serem eliminados, segundo os critérios desejados.

2.11 CRITÉRIO DE PARADA

É importante que o AG tenha um critério de parada determinado, se não iremos buscar infinitamente pela melhor solução. Além da chance da pesquisa andar em círculos, nunca obteremos a solução otimizada para o problema. Podemos utilizar diversos critérios, o mais simples e mais utilizado é o limite de gerações.

2.11.1 Limite de Gerações

Este critério tem como base contar quantas gerações foram feitas ao longo da execução de um AG, ao atingir o número pré-definido o algoritmo irá parar a busca e apresentar o melhor resultado. É muito importante escolher um valor adequado ao problema, já que este pode terminar a pesquisa muito rápido ou de forma muito tardia.

Como todos os parâmetros predefinidos antes da execução do algoritmo, tem grande influência no resultado porem sozinho não garante a solução do mesmo. Sendo assim esta é a forma mais simples de determinar a parada, não significa que é a menos importante.

2.11.2 Convergência Genética

Determinar a convergência genética é um dos fatores mais importantes a serem avaliados no AG. Esta irá determinar quando uma população tem baixa variedade genética, portanto quando há aglomeração dos indivíduos em um determinado ponto do mapa de soluções possíveis. Isto não é um caso sempre negativo, porem deve se impedir que ocorra prematuramente as iterações do AG, pois é possível que posteriormente a isto a otimização ocorra somente localmente, a menos que o operador de mutação force a população para fora deste ponto de aglomeração.

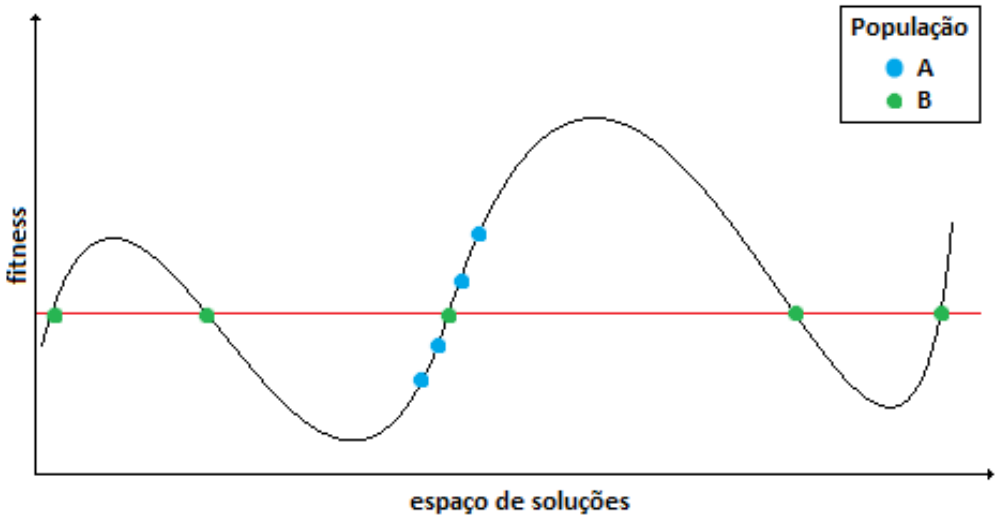

Figura 12 – Demonstração de convergência

Conforme apresentado na Figura 12, todos os elementos denotados pelos círculos verdes têm o mesmo valor de *fitness*, porém uma grande diversidade genética. Entretanto, os elementos denotados pelos círculos azuis têm variação de *fitness* muito distintas, mas estão concentrados em uma pequena área das soluções, neste caso existe convergência genética. Por este motivo não é possível levar em consideração a avaliação dos indivíduos como convergência, e sim uma proximidade genética de seus cromossomos em relação a uma espécie de grupo.

Não seria uma boa solução testar todos os indivíduos contra cada um de sua geração. Porque isto seria uma abordagem muito custosa em termos de tempo, onde teríamos para uma população de tamanho N um desempenho igual a de $O(n^2)$. Isto ocorrendo a cada geração

seria inaceitável, devendo utilizar esta abordagem somente quando for apropriado, como nas gerações mais próximas ao termino. Para verificar a ocorrência de convergência podemos usar o seguinte algoritmo que ameniza este problema.

```
Crie um conjunto com o primeiro individuo da população e com o número de indivíduos
igual a 1
i = 1;
Numeros_Conjuntos = 1;
Enquanto ((i=<tamanho_populacao) && (Numeri_Conjuntos < k)) faça
        j = 1;
        Enquanto (j =< Numero_Conjuntos) faça
                Se distancia (individuo^i e individuo do conjunto j) < &¨
                        Descarte o individuo[i]
                        Incremente o número de indivíduos do grupo j
                        Se o número de indivíduos do grupo j > m
                                Há convergência
                        Fim se
                        Interrompa o loop
                Fim se
        Fim enquanto
        Se j > Numero_Conjuntos
                Numero_Contjunos = Numero_Conjunto = 1;
                Crie um conjunto com i individuo[i] e com o numero de indivíduos igual a 1
        Fim se
        i = i + 1;
Fim enquanto
Se Numero_Conjuntos < k Então
        Há Convergência
Senão
        Não há convergência
Fim se
```

FONTE: Algoritmos Genéticos 2º ED, Brasport. p151

Neste algoritmo o que ocorre é que a cada indivíduo é testado se o mesmo está dentro do valor de distância máxima do indivíduo quanto ao conjunto testado, este será incluso no grupo. Se este não for selecionado para nenhum grupo então o mesmo irá compor um novo grupo, que fara parte dos testes seguintes. Deste modo ainda existe a chance de obter o pior caso em que cada indivíduo participa do próprio grupo sozinho. Mas se for escolhido com cuidado os parâmetros de distância do grupo e limite do grupo, é possível reduzir essa ocorrência.

2.12 RESULTADO DA BUSCA

Ao término da execução do algoritmo, após várias iterações e aplicações de operadores genéticos, obteremos o melhor indivíduo encontrado ao longo da busca, lembrando que este pode não se tratar da melhor solução, pois este algoritmo otimiza resultados a partir da geração inicial. Além disso, temos que reverter a conversão dos valores do problema para o cromossomo e, por fim, obter os valores otimizados para o problema.

2.13 CARACTERÍSTICAS AVANÇADAS DOS ALGORITMOS GENÉTICOS

2.13.1 Outras Representações do Cromossomo

> "A pergunta que se impõe é: quando a representação binaria não é a mais adequada? Existem alguns problemas associados à mesma que podem ajudar a responder esta pergunta" (HERRERA, 1998)

A representação binária não é a única forma de abstrair um problema para o conceito de AG. A maioria das estruturas de dados podem ser utilizadas neste processo. O contratempo para utilização de outras estruturas é que cada uma tem seu próprio conceito de operadores genéticos e *fitness*, tornando a escolha do formato do cromossomo, essencial para o bom desempenho do algoritmo.

> "A representação binaria tem dificuldade ao lidar com múltiplas dimensões de variáveis continuas, especialmente quando uma grande precisão é requerida" (LINDEN, 2008)

Já foram utilizadas várias estruturas diferentes, e gerado no processo seus operadores genéticos. Uma destas é a substituição do cromossomo binário por uma sequência de numerais, onde cada valor pode representar o valor de uma variável do problema, sendo ela inteira ou não.

Por exemplo números inteiros permaneceram representados por seus respectivos valores, entretendo isso obriga a utilização de novos operadores. Uma vez que os operadores binários só podem ser obviamente aplicados a números binários.

2.13.1.1 Árvore Sintática e seu Operador

Na computação e linguística a análise sintática significa analisar uma sequência de dados que respeitam uma estrutura gramatica segundo uma gramatica formal. Esta análise está

integrada ao compilador, juntamente com a análise léxica e semântica. Sendo assim trabalha com um grupo de “tokens” previamente definidos, que definem um conjunto de regras para se construir uma arvore sintática.

Uma arvore é escrita a partir de uma gramatica, um programa é geralmente criado para dar suporte ao AG, para representar a partir uma sequência de “tokens” a criação de uma arvore de funções. Como por exemplo, a conversão de uma função em uma arvore de operações matemáticas, representada pela Figura 13.

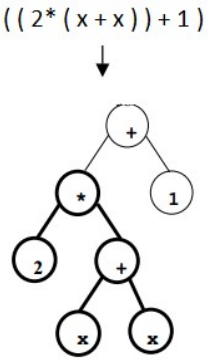

Figura 13 – Exemplo de conversão de tokens em uma estrutura de árvore
FONTE: Computação Evolutiva, POZO UFP 2004

Neste processo o cruzamento entre os indivíduos é um pouco mais complexo. Será escolhido um determinado ponto nos ramos de ambas as arvores, onde será feita a troca de seus ramos, criando ao termino dois novos indivíduos, como visto na Figura 14.

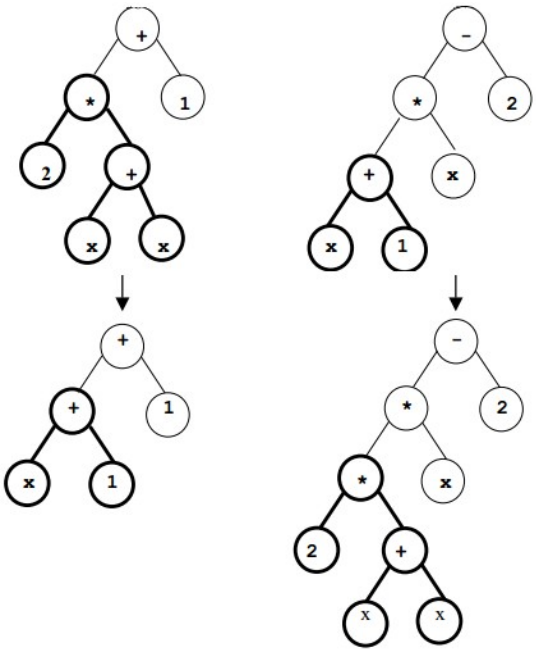

Figura 14 – Exemplo de cruzamento de árvores
FONTE: Computação Evolutiva, POZO UFP 2004

Importante ressaltar que a escolha de tais estruturas tem relação direta com a natureza do problema tratado. Podendo ser útil para alguns casos e como para outros não.

2.13.2 Máximo Local e Global

> "Os algoritmos genéticos são técnicas heurísticas de otimização global. A questão da otimização global opõe os algoritmos genéticos aos métodos como gradiente (hill climbing), que seguem a derivada de uma função de forma a encontrar o máximo de uma função, ficando facilmente retidos em máximos locais" (LINDEN, 2008)

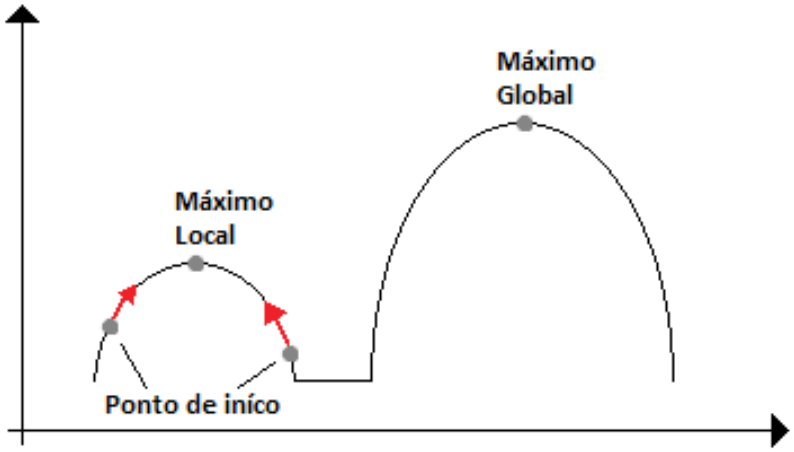

Figura 15 – Máximo local x global
FONTE: Algoritmos Genéticos 2º ED, Brasport. P43

O problema do máximo local ocorre quando, por exemplo, no algoritmo escalada de encosta, a solução fica presa a um máximo local como visto na Figura 15. Que dentro de sua limitada capacidade de avaliação e busca do máximo global, o algoritmo ficaria preso a uma local. Isto impede o encontro de solução que é realmente desejada na busca.

Entretanto, como dito anteriormente, o AG não sofreria deste mal. Uma vez que é capas de driblar este problema, utilizando por exemplo o operador de mutação, caso tenha uma convergência genética muito prematura em relação ao termino da busca. Entre outros recursos que podem ser utilizados, isso não significa que não se deve ter o cuidado para evitar esse problema. Uma vez mal configurado, o AG pode levar mais gerações do que o desejado para sair do máximo local. Semelhante a uma migração de indivíduos em um terreno com partes ruins de se locomover até outras mais ricas.

2.13.3 *Windowing*

Uma boa população inicial costuma ter grande variedade nos valores dos *fitness*, tornando fácil a percepção dos melhores indivíduos. Porem com o passar das gerações, é possível verificar que há uma maior proximidade no valor de *fitness* entre os indivíduos, ao se aproximar do valor máximo da função *fitness*. Isto é um risco, uma vez que está proximidade não valoriza os indivíduos melhores na seleção, já que seus valores são muito semelhantes por sua vez tem oportunidade de seleção quase iguais.

> "Caso usemos métodos de seleção baseados na roleta, a pressão seletiva seja diminuída, pois os pedaços da roleta associados aos indivíduos são extremamente similares" (LINDEN, 2008)

Para isso foi criado o método *Windowing* que aumenta a diferença entre os piores e os melhores indivíduos. Aplicando ao *fitness* de determinada geração uma função sobre ela, que destacaria a diferença entre os indivíduos.

Esta técnica pode ser resumida da seguinte maneira: ache o valor mínimo dentre as funções de avaliação da nossa população e diminua-o de um valor pequeno arbitrativo. Designe para cada um dos cromossomos uma avaliação que seja igual à quantidade que excede este valor mínimo (LINDEN, 2008)

Portanto como apresentado na Figura 16, deste modo irá alterar as diferenças de *fitness* dos indivíduos aumentando novamente a pressão seletiva entre eles. Fator importante para diferenciar os melhores pais, quando temos *fitness* muito semelhantes.

> "[...] em vez das avaliações irem de 999.001 a 999.999 elas passarão a ir de 0.001 a 0.999. Isto faz com que as áreas pertencentes a cada indivíduo passem a ser significativamente diferentes daquelas associadas à indivíduos que tenham uma avaliação apenas 0,5 menor […]" (LINDEN, 2008)

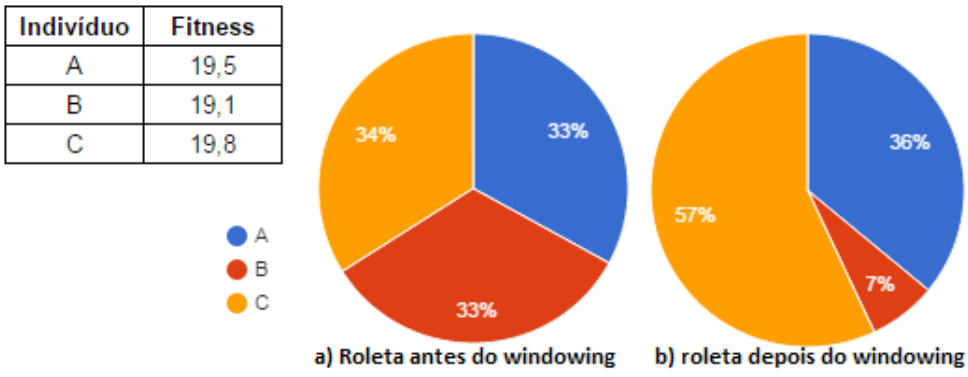

Figura 16 – Comparação de roleta antes e depois do windowing
FONTE: Algoritmos Genéticos 2º ED, Brasport. p162

2.13.4 Preservando a Diversidade

Um dos grandes problemas em um AG é a convergência genética precoce, visto este problema é recorrente. Foram criados inúmeros métodos para tentar retardar este risco, dando oportunidade para o algoritmo fazer o seu trabalho, dentro do tempo determinado pela função de termino.

> Existem alguns métodos de preservação de diversidade que normalmente são aplicados a funções de múltiplos objetivos, mas que podem ser usadas em todas as situações por serem muitos simples e interessantes. Estes métodos são baseados na incorporação de informação sobre a distribuição de densidade dos indivíduos. Assim, quanto maior a densidade de indivíduos na sua vizinhança, menores serão as chances de um indivíduo ser selecionados. (LINDEN, 2008)

Para isto a intenção de preservar a diversidade genética seria uma das principais técnicas para controlar a convergência. Como a convergência é dada pela proximidade genética dos indivíduos da população, é logico pensar que com uma maior diversidade podemos evitar este problema. Dentre os métodos mais usados temos dois *Nearest-Neightbour* (Vizinho mais próximo) e Histogramas.

É importante verificar que todos estes métodos são custosos, tornando o tempo necessário para o processamento do AG muito longo, para evitar isto estes métodos são chamados a partir de um determinado nível de alta convergência genética. Geralmente já num ponto mais avançado da busca pela solução.

2.13.4.1 Nearest-Neightbour

É calculado a distância entre todos os indivíduos e posteriormente ordenado os elementos por esta distância. Onde o resultado é utilizado para reduzir a chance de seleção de indivíduos com mais vizinhos, e aumentar aqueles que se encontram em pontos mais isolados. Não é descartado o *fitness* de cada um, a intenção é aumentar a chance dos melhores indivíduos sozinhos, e reduzir a daqueles mais aglomerados em uma pequena região.

2.13.4.2 Histogramas

Toda a região de solução é subdividida de forma a se tornar uma hiper grade, e se conta a quantidade de indivíduos que estão presentes em cada região. Onde mais uma vez as áreas mais aglomeradas receberam penalidades em sua seleção. Em contrapartida as mais dispersas ganharam vantagem, mas sempre utilizando o *fitness* de cada indivíduo no processo.

3 APLICAÇÕES DO ALGORITMOS GENÉRITOS

Neste capitulo serão apresentadas algumas aplicações conhecidas do AG, com o objetivo de mostrar que este algoritmo é muito versátil, podendo ser aplicado em vários casos. Entretanto, seu desempenho depende integralmente da qualidade da abstração do problema para os métodos utilizados no algoritmo genético. Lembrando que é necessária uma boa escolha na forma de representação genética e da função de *fitness*.

> "É importante lembrar que o uso de algoritmos genéticos deve ser limitado apenas aquelas situações em que não há um algoritmo exato capaz de resolver o problema em um tempo razoável" (LINDEN, 2008)

Se os pontos previamente discutidos forem adequadamente planejados, a possibilidade de utilização de AG é grande, podendo ser utilizado no setor elétrico para o planejamento de expansão, visando tanto o fator econômico como técnico, na alocação de recursos ao escalonar horários em uma instituição de ensino, ou para definir horários de disciplinas e salas.

3.1 MONA LISA

Foi utilizada uma versão do algoritmo genético para recriar a imagem do famoso quadro Mona Lisa de Da Vinci. O sr. Roger Alsing mesclou um programa de geração de imagem com um algoritmo genético que iria buscar a forma mais simples de representar a imagem originalmente apresentada. O resultado de seu trabalho pode ser visto em seu "Weblog", onde o principal objetivo era criar uma nova imagem mais próxima possível do original. A regra usada foi distribuir, em um fundo negro, o menor número de polígonos coloridos semitransparentes, onde o valor *fitness* é o resultado de uma comparação de semelhança da imagem gerada com a original. Cada indivíduo seria composto de uma quantidade máxima de polígonos, onde cada polígono poderia ter posição, formato e cor diferente. Este é, respectivamente, o cromossomo e gene, base da solução do problema.

```
double fitness = 0

for y = 0 to width
  for x = 0 to height

      color c1 = GetPixel(sourceImage, x, y)
      color c2 = GetPixel(generatedImage, x, y)
```

```
        //get delta per color
        double deltaRed = c1.Red - c2.Red
        double deltaGreen = c1.Green - c2.Green
        double deltaBlue = c1.Blue - c2.Blue

        // measure the distance between the colors in 3D space
        double pixelFitness = deltaRed * deltaRed +
                              deltaGreen * deltaGreen +
                              deltaBlue * deltaBlue

        //add the pixel fitness to the total fitness ( lower is better )
        fitness += pixelFitness

    end
end
return fitness
```

FONTE: http://rogeralsing.com/2008/12/09/genetic-programming-mona-lisa-faq/

No quadro anterior encontra-se o código da função *fitness* do algoritmo genético, onde a imagem original é comparada pixel a pixel com a imagem gerada. Por fim, a soma dessas diferenças ao final sempre terá um valor positivo, de forma que a proximidade do zero determina o quão semelhante está a imagem gerada.

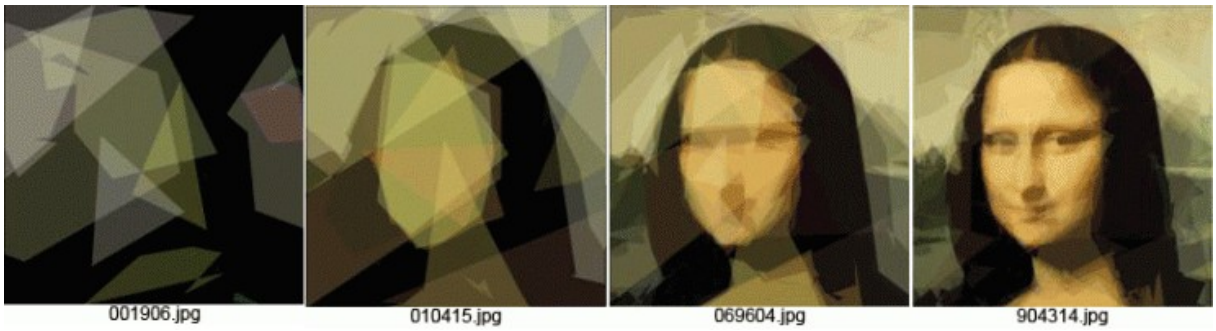

Figura 17 – Seleção de indivíduos ao longo de gerações
FONTE: https://rogeralsing.files.wordpress.com/2008/12/evolutionofmonalisa1.gif

A Figura 17 é uma pequena seleção dentro de todas as gerações, onde cada uma das quatro imagens são indivíduos, estes estão ordenados pelo seu *fitness*, . É possível ver o progresso do algoritmo em encontrar a melhor solução desde o primeiro com uma imagem irreconhecível até o ultimo que é muito semelhante ao original, levando em consideração que a imagem gerada pode ter no máximo 50 polígonos. Isso mostra o quão versátil é a utilização do AG.

3.2 EVOLUÇÃO DE ANTENA

> “A prática atual de projetar e otimizar as antenas a mão é limitada em sua capacidade de desenvolver algo novo e melhor simultaneamente, porque requer significante expertise e muito tempo de trabalho intensivo. Como alternativa, os investigadores têm investido no design da antena evolutiva e otimização. Desde o início de 1990." (HORNBY, 2006)

Este foi um trabalho feito para a NASA com o objetivo de desenvolver um novo projeto de antena, adequado a um conjunto de novas necessidades. Para isso, eles tiveram que programar diversos pontos a serem evitados na evolução da antena, ou seja, regras de exclusão na criação de indivíduos, como a quantidade de elementos envolvidos na fabricação da mesma, dentre outros detalhes técnicos. Por exemplo, a antena deveria ter ganho uniforme independente de sua posição uma vez que o satélite com o qual desejavam manter comunicação encontrava-se girando em órbita. Assim, seria possível garantir uma boa comunicação indiferente da posição da antena em relação a sua rotação. Na figura 18 é possível ver duas antenas geradas no processo.

> “[...] com duas antenas de 38% eficiência conseguimos evoluir uma antena que resultou em uma eficiência de 80%. E por sua vez com estas novas antenas evoluímos a um resultado seguinte de 93% de eficiência. Com requisitos de energia mais baixos, o que resultam alto ganho [...]” (HORNBY, 2006)

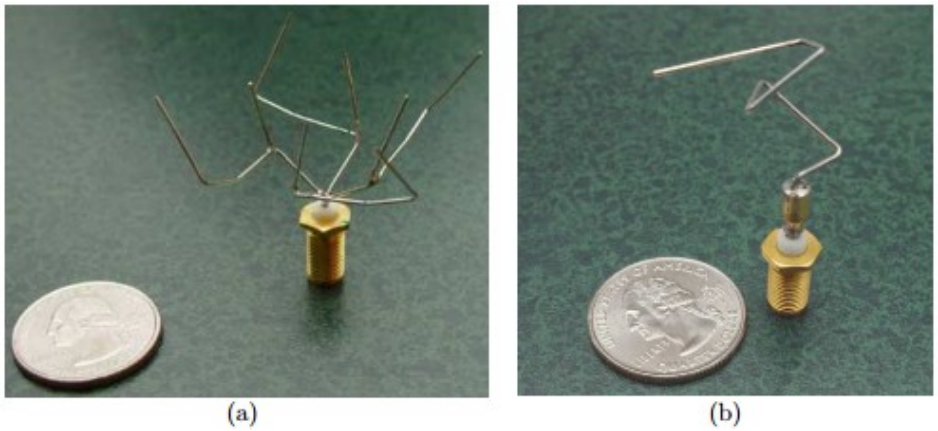

Figura 18 – Fotografias de protótipos de antenas evoluídas: (a) a melhor antena evoluiu para o padrão de ganho inicial requisito, ST5-3-10; (b) a melhor antena evoluiu para as especificações revistas, ST5-33-142-7.
FONTE: HORNBY, 2006

> “[...] Em termos de trabalho, a antena evoluída precisou de cerca de três pessoas por mês para projetar e fabricar. Enquanto a antena convencional são necessárias cerca de cinco pessoas por mês [...]” (HORNBY, 2006)

Assim, foi possível também reduzir o custo no desenvolvimento da antena em diversos componentes, além de obter melhores resultados em seu desempenho final, utilizando padrões de antenas bem inferiores, se comparado ao produto final. Isto mostra que, com estudo adequado da área a ser trabalhada e a codificação de todos os requisitos necessários, é possível utilizar AG em áreas muito especificas.

> "[...] Além de ser o primeiro hardware evoluído no espaço, as antenas evoluídas demonstraram várias vantagens sobre as antenas convencionalmente concebidas. O algoritmo que usamos não se limitou às variações de formas de antena anteriormente desenvolvidos, mas gerou milhares de completamente novos tipos de projetos, muitos dos quais têm estruturas incomuns aos especialistas projetistas [...]" (HORNBY, 2006)

Esses exemplos procuraram mostrar que o AG tem uma grande capacidade de apoio às pesquisas, acelerando os resultados obtidos, e possibilitado descobrir novas soluções que não eram vizualizadas anteriormente, como estas novas antenas ainda incomuns para os especialistas.

4 ESTUDO DE CASO

Neste capítulo será apresentado um estudo comparativo entre o Algoritmo de Busca Linear (ABL), um tipo de algoritmo de busca exaustiva que é inadequado para problemas de complexidade exponencial, com o AG que tem uma natureza peculiar em sua complexidade, favorecendo o seu uso em casos especiais, já que esta diferente do ABL não depende exclusivamente da informação de entrada.

> "Agora levemos em consideração que uma maquina moderna pode realizar cerca de 10^9 operações em um segundo e podemos determinar que, se tivermos que realizar 10^{30} operações [...], levaremos um tempo de ordem de 10^{21} para termina-las. Pensando que um dia tem pouco menos que 10^5 segundos, isto significa um total de 10^{16} dias ou aproximadamente 10^{13} anos" (LINDEN, 2008)

Portanto, quando se trata de problemas com complexidade igual ou superior a n^x, podemos concluir que algoritmos que utilizem uma busca exaustiva como solução é um caminho inviável. Neste ponto, a utilização do AG se torna necessária, uma vez que sua complexidade depende mais dos parâmetros e métodos escolhidos do que o universo de soluções, viabilizando a busca de soluções para casos como, por exemplo, o caixeiro viajante.

4.1 ALGORITMOS E SEU TEMPO DE EXECUÇÃO

Do ponto de vista da complexidade dos dois algoritmos, vale ressaltar que o ABL tem complexidade O(n), ou seja, o tempo aumenta na medida em que aumentamos o número de dados a serem pesquisados. Entretanto, o AG tem complexidade constante, para cada configuração nele aplicada, o que não significa que sempre retornará a resposta desejada, visto que para cada problema deve ser avaliada uma configuração ideal.

Como uma forma abstrata de definir a complexidade de um AG (Aguiar, 1998), podemos considerar uma função geral da complexidade como:

$$f(n) = O(CompInic+[NumGer*(CompReprod+CompTerm)]+CompRecupera)$$

onde:

- NumGer : número de gerações.
- CompInic : complexidade para criar população inicial.
- CompReprod : complexidade dos operadores e método seleção.

- CompTerm: complexidade para verificar se o objetivo foi alcançado e/ou condição de parada.
- CompRecupera: complexidade de retornar os dados referentes a resolução ou não do problema.

Portanto para cada configuração do AG teremos um tempo de execução diferente, além de afetar no desempenho do próprio algoritmo, o que irá afetar diretamente a sua capacidade de encontrar a solução em tempo hábil. É possível observar que segundo esta fórmula, não somente os parâmetros iniciais influenciariam no desempenho como também a escolha dos métodos, como método de seleção, tipo de operador genético, função de avaliação fitness, ou seja, vai muito além de somente o numero limite de gerações e de população.

4.2 ALGORITMO DE BUSCA LINEAR

Conforme apresentado na Figura 19, o fluxograma deste algoritmo é um dos mais simples. Essa não é a melhor solução, visto que existem outras formas de busca mais eficiente em um vetor. Porém, para o AG o vetor de valores não deverá estar necessariamente ordenado, como é necessário para o uso de busca binária, por exemplo.

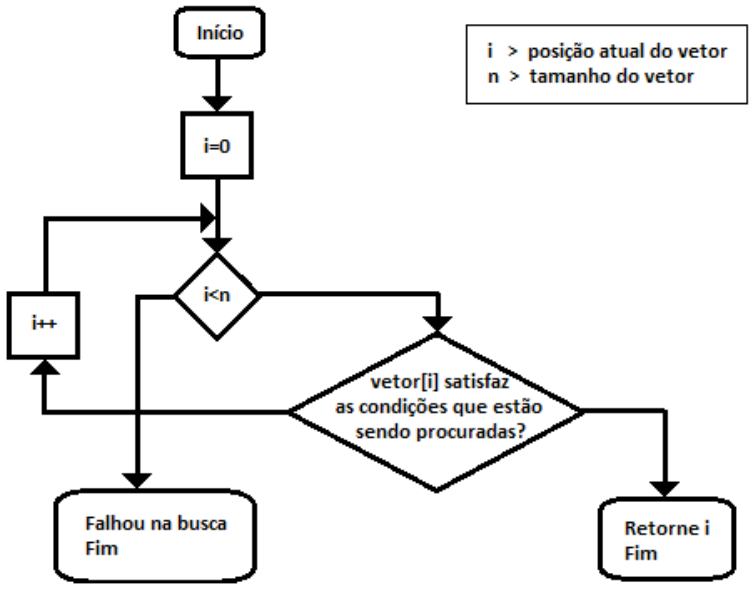

Figura 19 – Fluxograma de busca linear
FONTE: http://pt.wikipedia.org/wiki/Busca_linear

Neste caso, a medida que cresce o vetor de valores a ser pesquisado, o tempo necessário irá crescer diretamente proporcional. Sua complexidade é O(n) para o pior caso,

podendo se tornar lento quando o problema a ser tratado tem uma área de busca muito ampla, devido a taxa de crescimento da função.

4.3 CONDIÇÕES DO TESTE

Java foi a linguagem escolhida para codificar os algoritmos do teste, principalmente por ser independente de plataforma. Além disso, é totalmente orientada a objetos, o que facilita a construção das estruturas necessárias para o algoritmo genético, como por exemplo a lista de indivíduos que tem transformação constante.

Os parâmetros escolhidos para o algoritmo genético são os seguintes:

- Para controle da população foi escolhido a Steady State, onde a população terá um limite de dez indivíduos. Cada geração irá criar sete filhos, com uma taxa de probabilidade crossover de 80% e cada indivíduo tem chance de 40% de mutação, onde cada gene terá uma chance de ser ou não alterado, a mutação será crucial para evitar que a população se concentre em um máximo local.
- A condição de término é o limite de dez gerações.
- O *fitness* é bem simples, para cada indivíduo o resultado da função f(x) será o seu valor em fitness, uma vez que procuramos a maximização da função . Ou seja para um individuo de cromossomo igual a 15, após ser convertido ao valor inteiro, é feito f(x) com x = 15 o que resultara no seu fitness.

4.4 OBJETO DE TESTE

Para comparar os dois algoritmos serão utilizadas algumas funções matemáticas, que irão resultar em cenários distintos. A seguir são apresentadas três funções juntamente com seus gráficos (Figura 20). O objetivo da busca é encontrar o MAX de f(x). Para reduzir o espaço de busca, os valores de x são limitados ao intervalo de 0 até 100 e foi utilizada a precisão de cinco casas decimais, pois essa precisão apresentou melhor desempenho na execução, preservando uma boa precisão nos resultados. Ao final serão mostradas as qualidades e defeitos destas técnicas, especialmente para o AG.

1º caso: f(x) = (x/8);

2º caso: f(x) = sin(x/4)*10;

3º caso: f(x) = sin(x/4)*sin(x/5);

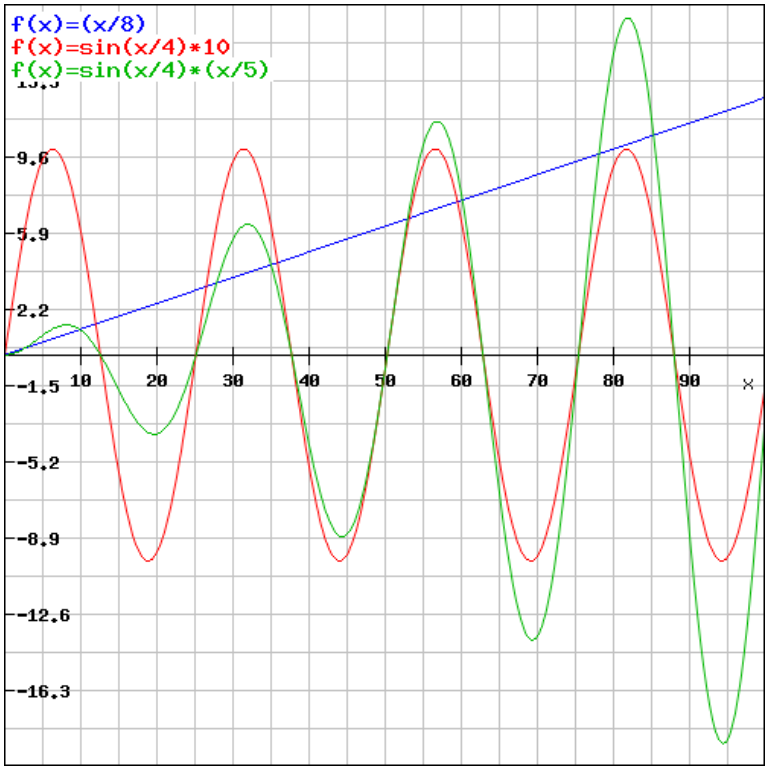

Figura 20 – Gráfico das funções de teste

É possível observar que na primeira e terceira funções só existe um valor máximo, e na segunda função temos várias soluções possíveis. Entretanto, somente a segunda e a terceira funções tem máximos locais, o que é um problema para alguns algoritmos inteligentes, pois um AG mal configurado pode ficar preso em um máximo local e não encontrar o máximo global, que no caso é a solução correta.

4.5 COMPARANDO RESULTADOS

A Tabela 3 apresenta os resultados da execução dos dois algoritmos. Como o AG é um algoritmo aleatoriamente guiado, podendo ter resultados diferentes a cada execução utilizando as mesmas configurações, não seria possível comparar com um único teste o seu real desempenho. Para isto foi verificadao a média de acertos em uma bateria de 100 testes consecutivos, com uma margem de erro de 3%.

Tabela 3 - Resultados da execução (ms = milissegundo)

Algoritmo de Busca Linear Tradicional					
Função	**x**			**f(x)**	**tempo (ms)**
1	100,0			12,5	370,8
2	6,28319; 31,41593; 56,54867; 81,68141			9,99999	870,62
3	81,87667			16,35582	886,84
Algoritmo Genético					
Função	**x**	**f(x)**	**precisão**	**desvio**	**tempo (ms)**
1	100,0	12.5	99,0%	0,0%	11,43
2	5,97033	9,96942	99,0%	0,3057%	17,27
3	81,51248	16,28795	90,0%	0,414%	17,63

Nos resultados do primeiro caso é possível notar uma grande diferença no tempo de busca do algoritmo genético comparado com a busca linear, com uma precisão de acerto de 99%, ou seja, dentro dos cem testes somente um falhou a atingir a meta de 3% de erro na busca do valor máximo. Deste modo é possível observar que, neste caso, o algoritmo genético chegou a ser 25 vezes mais rápido que o de busca linear, a custo de uma taxa de 1% de erro com mais que 3% de desvio do valor desejado. Entretanto, os outros 99% tem o maior desvio encontrado com valor aproximado a 0%.

No segundo caso, é possível notar que o desempenho da busca linear caiu muito, chegando a ultrapassar o dobro do tempo. Isto ocorreu devido ao aumento do custo de execução da função matemática f(x) do segundo caso, uma vez que o ABL executa exaustivamente esta função, este algoritmo teve grande perda em seu desempenho. Entretanto o AG executará a função f(x), somente após o êxito da criação de cada individuo, definindo então seu fitness no resultado. O que não ocorre na maior parte do tempo de execução do algoritmo, pois o maior foco deste é a seleção e operação genética entre os indivíduos.

Já a taxa de precisão permaneceu a mesma, porém ouve uma elevação no desvio do resultado. Isto ocorreu devido a condição do meio que é pesquisado, como o segundo caso se trata de diversas soluções possíveis. Deste modo tem diversos máximos locais, tornando as gerações mais imprecisas para otimizar o caminho adequado, fazendo com que demore mais para chegar a um resultado. Mesmo assim, o AG superou o algoritmo linear em tempo, a um custo baixo semelhante ao primeiro caso.

No terceiro caso, entretanto, houve um crescimento na taxa de erro acima do desejado. Isto ocorreu porque nesta função, temos diversos máximos locais como no segundo caso, porém somente um deles é considerado o máximo global. No segundo caso, esta diversidade de soluções dificultava a precisão na busca, uma vez que a população ficava migrando de um topo ao outro, até se estabilizar em um único ponto e atingir a convergência genética. Contudo, diferente do segundo caso onde isto poderia ocorrer em qualquer um dos topos, no terceiro caso somente um forneceria a resposta desejada, tornando assim a migração das populações entre os falsos máximos globais o mais rápido possível o maior desafio. Caso isso demorasse a ocorrer, os indivíduos iriam atingir a convergência genética no espaço errado. O método responsável por esta migração é a mutação, que forçava os indivíduos mudarem de local abruptamente, o que posteriormente com o cruzamento poderia levar toda a população a um novo local ideal.

4.6 OTIMIZANDO O PIOR CASO

Para mostrar que a configuração inicial do algoritmo genético tem grande importância em seu desempenho, e se bem configurado pode evitar aumento de erros em seus resultados, foram efetuadas modificações nas configurações do AG. A partir dessas modificações foram avaliadas as mudanças no desempenho e precisão. A Tabela 4 apresenta o comparativo dos testes efetuados. Para aumentar a precisão nos resultados do AG foi calculada uma média de 500 testes consecutivos para cada configuração.

Tabela 4 – Tabela de comparação resultado de configurações

Configuração	CM	CC	LG	TP	FG	Precisão	DM	Tempo (ms)
1	40%	80%	10	10	7	91,4%	0,695%	17,79
2	40%	80%	20	10	7	99,2%	0,414%	20,12
3	40%	80%	20	6	3	88,7%	0,751%	9,73
4	70%	70%	20	6	3	95,8%	0,628%	10,32

ms = milissegundo, CM = Chance mutação, CC = Chance crossover, LG = Limite gerações, TP = Tamanho da população, FG = Filhos por geração, DM = Desvio médio

O primeiro caso é a repetição dos testes anteriormente mencionados. Seus resultados permaneceram semelhantes, na média de resultado dos novos testes. A segunda configuração teve um grande ganho na precisão, e isso foi possível somente alterando o limite de gerações,

permitindo maior tempo para aperfeiçoamento da população. Isto ocasionou um aumento de tempo de processamento e também uma diminuição razoável da média do desvio padrão.

No terceiro caso, na tentativa de reduzir o processamento envolvido na criação de cada indivíduo e consecutivamente na obtenção de seu *fitness*, foram feitas alterações no tamanho da população e na quantidade de filhos por geração. Isso provocou o dobro da velocidade na obtenção da solução, porém apresentou uma queda brusca na precisão e o desvio padrão superou o primeiro caso.

Por fim, no último caso foi aumentada a chance de mutação e alterada ligeiramente a chance de *crossover*, o que conseguiu conciliar qualidades do primeiro e terceiro caso, como o aumento da precisão junto com a redução no tempo necessário para execução do AG. Portanto, analisando os resultados podemos chegar a bons conjuntos de parâmetros capazes de atingir resultados ainda mais otimizados que o anterior.

4.7 CONCLUSÃO DO ESTUDO DE CASO

Após a otimização do caso em que se obteve inicialmente o pior desempenho, é possível observar que a aplicação do AG é muito dependente de sua configuração. Por sua vez, esta é ligada diretamente ao estudo do problema a ser tratado pelo AG, sendo crucial como ponto de partida para a decisão dos métodos utilizados.

Na comparação entre os algoritmos é possível observar uma diferença significativa de desempenho entre eles, observando-se no tempo demandado por cada um. Entretanto, o AG não encontra sempre a melhor resposta, o que é um fator negativoem comparação com o ABL que, mesmo lento, é capaz de encontrar sempre o resultado mais otimizado. Entretanto, se o ambiente da busca for bem analisado a fim de definir um bom conjunto de configurações que aumentem a chance de êxito do AG é possível superar o ABL com uma busca extremamente rápida, com uma ótima precisão e baixa taxa de erro.

5 CONCLUSÃO

Durante esse trabalho foi possível observar que existe uma grande variedade de opções para aplicação do AG, além de uma série de estudos e trabalhos a respeito dessa técnica, que procuram demonstrar sua versatilidade. Assim, o AG é uma técnica eficiente, que deve ser considerada na solução de problemas computacionais que busquem a otimização a partir de grandes volumes de dados. Entretanto, vale ressaltar que a aplicação dessa técnica depende de pessoal com conhecimento avançado sobre o problema a ser tratado, como no exemplo apresentado sobre a evolução da antena, onde era necessário não somente o conhecimento do AG, como um vasto conhecimento técnico dessa área específica.

Como apresentado no estudo de caso o AG, se devidamente configurado, tem um desempenho muito eficiente mesmo para problemas de complexidade O(n), principalmente para entradas de dados muito volumosas, onde há um aumento natural do tempo de execução. Por este motivo, essa técnica pode ser aplicada para casos de complexidade como $O(n^2)$ ou O(n!), uma vez que suas complexidades são muito superiores a O(n), tornando o AG uma técnica viável de otimização de soluções no universo dos problemas intratáveis.

Na comparação entre os algoritmos AG e ABL foi possível observar uma diferença significativa de desempenho entre eles. Entretanto, o AG necessita de uma análise detalhada para definir configurações que aumentem sua chance de êxito com alta precisão e baixa taxa de erro. Para contornar esse risco é necessário estudar detalhadamente os resultados no ambiente de testes até atingir as condições desejadas e, deste modo, definir uma solução que retorne resultados em tempo e precisão aceitáveis.

6 REFERÊNCIAS

AGUIAR, Marilton S. Análise Formal da Complexidade de Algoritmoa Genéticos. UFRGS, 1998.

BREMERMANN, H. J. Optimization through evolution and recombination. Spartan Books, 1962.

FILHO, Paulo A. C.; POPPI, Ronei J. Algoritmo Genético em Química. Universidade Estadual de Campinas, São Paulo, 1998.

FOGEL, L. J.; OWENS, A. J.; WALSH, M. J. Artificial Intelligence through Simulated Evolution. New York, USA, 1966.

GROSKO, Ana Paula; GORSKI, José Robeto; DIAS, João S. Algoritmo Genético: Revisão Histórica e exemplificação. Santa Catarina, 2006.

HERRERA, F.; LOZANO, M; VERDEGAY, J.L., Tackling Real-Coded Genetic Algorithms - Operators and Tools for behavioural Analysis. Amsterdã, Holanda, 1998.

HOLLAND, John Henry. Genetic Programming. London, England, 1998.

HORNBY, Gregory. S. Automated Antenna Design with Evolutionary. University of California, Santa Cruz, 2006.

JOHANSSON, Roger. Genetic Programming Evolution of Mona Lisa. Disponível em: http://rogeralsing.com/2008/12/07/genetic-programming-evolution-of-mona-lisa/>. Acesso em: 28 janeiro de 2015.

LINDEN, R. Algoritmo Genéticos.Brasport, 2º ed., Rio de Janeiro, 2008.

MELANIE, Mitchell. An Introduction to Genetic Algorithms. London, England, 1999.

MENDEL, Gregor. Experimentos de Hibridação em Plantas, 1865.

NETO, Sílvio Petroli. Computação Evolutiva: desvendando os algoritmos genéticos, Mestrando. São Paulo, 2011.

PAUL, Joaquim M. Tradução de A Origem das espécies de Charles Darwin de 1859. Portugal, 2003.

POZO, Aurora; FATIMA, Andrea de; ISHIDA, Celso. Computação Evolutiva: Grupo de Pesquisas em Computação Evolutiva. Universidade Federal do Paraná, 2004.

TANOMARU, Julio. Motivação, Fundamentos e Aplicações de Algoritmos Genéticos. Tokushima, Japão, 1995.

TURING, Alan. M. Computing Machinery and Intelligence. 1950.

ZUBEN, Fernando J. V. Computação Evolutiva: Uma Abordagem Pragmática. SP, 2000.