**Centro de Ensino Unificado de Teresina - CEUT**

Dann Luciano de Menezes

**Algoritmos Genéticos na Otimização de Consultas por Similaridade**

Teresina, PI

2010

**Centro de Ensino Unificado de Teresina - CEUT**

Dann Luciano de Menezes

**Algoritmos Genéticos na Otimização de Consultas por Similaridade**

Projeto apresentado como requisito para conclusão da disciplina de Trabalho de Conclusão de Curso I e pré requisito para cursar a disciplina de Trabalho de Conclusão de Curso II

Teresina, PI

2010

**Sumário**

**1 Tema**

**2 Problema**

**3 Objetivos**

**4 Justificativa**

**5 Metodologia**

**6 Referenciais Teóricos**

**6.1 Consultas por Similaridade**

**6.2 Algoritmos Genéticos**

**7 Cronograma**

**Referências**

**1 Tema**

Algoritmos Genéticos na Otimização de Consultas por Similaridade.

**2 Problema**

Como otimizar o tempo e os recursos computacionais e a qualidade da informação obtida em consultas por similaridade aproximada?

**3 Objetivos**

* Implementar uma ferramenta de busca em Métodos de Acesso Métricos (MAMs) utilizando a técnica dos Algoritmos Genéticos;
* Criar uma representação do problema, afim de que este possa ser resolvido através dos Algoritmos Genéticos;
* Minimizar o tempo e os recursos computacionais envolvidos na obtenção das possíveis soluções para o problema.

**4 Justificativa**

O tempo computacional gasto para realizar a operação de recuperação de dados em SGBDs é relativamente baixa, considerando o volume de informações armazenadas, mas isso só ocorre somente nos tipos de dados mais simples como números e letras. Quando aumentamos a complexidade do dado a ser trabalhado aumentamos também o tempo computacional do processo de busca. O pior caso é quando temos que trabalhar com dados binários, como imagens, músicas ou vídeos. Esses dados complexos são armazenados em estruturas de dados especificas nomeadas de Métodos de Acesso Métricos. Apesar das características de indexação dos Métodos de Acesso Métricos, as operações de consultas dos dados são muitas vezes ineficientes devido ao problema da “maldição da dimensionalidade” (apud Bueno et al 2005, pag). Este ainda afirma que “em alguns casos a busca seqüencial é mais vantajosa”. Neste trabalho, procuramos mostrar uma abordagem diferente e eficiente, pois segundo Zuben (***~~ANO~~***) são os problemas de otimização que mais recebem contribuições a partir das técnicas de computação evolutiva, sendo os Algoritmos Genéticos a técnica que utilizaremos neste trabalho.

**5 Metodologia**

**6 Referenciais Teóricos**

Para implementar uma ferramenta de busca em Métodos de Acesso Métricos, conforme proposto anteriormente, faz-se necessário um dialogo com autores da área de Consultas por Similaridade, afim de que compreendamos melhor toda a teoria existente por trás da resolução do problema.

**6.1 Consultas por Similaridade**

Os primeiros Sistemas Gerenciados de Base de Dados (SGBDs) tinham a finalidade de facilitar e agilizar o processo de recuperação exata dos dados. Essa recuperação exata foi e continua sendo uma das necessidades das aplicações que os SGBDs devem suportar. Com o passar do tempo novas necessidades foram surgindo, e a recuperação exata dos dados deixa de ser necessária em algumas aplicações especificas como, sistemas de apoio à decisão dentre outras. Segundo Bueno et al (2005) isso se dá porque essas aplicações possuem uma característica exploratória. Outra funcionalidade que as aplicações atuais necessitam é o armazenamento e recuperação de tipos de dados complexos, como já foi afirmado acima.

As operações de recuperação não são realizadas em cima dos dados complexos propriamente dito, e sim de características extraídas quando os dados são inseridos ou alterados. Os dados complexos devem passar antes por extratores de características que codificam as características dos dados complexos em outras que os descrevem e depois adicionam essas características descritivas em estruturas de dados denominadas de Métodos de Acesso Métricos. Os Métodos de Acesso Métricos, por outro lado servem para indexar os dados pertencentes a um Domínio Métrico, que será discutido em seguida.

**6.1.1 Domínios Métricos**

Um Domínio Métrico é definido como M = (D, d( )), onde D é o conjunto de todos os objetos que atendem às propriedades do domínio e d( ) é uma função de distância, ou métrica, entre esses objetos. Dados x,y e z E D, uma função de distância (métrica) d : D x D 🡪 R+ deve satisfazer as seguintes propriedades:

*1 Simetria*: *d(x,y) = d(y,x);*

*2 Não Negatividade*: *0 ≤ d(x,y) ≤ ∞, d(x,x) = 0;*

*3 Desigualdade Triangular*: *d(x,y) ≤ d(x,z) + d(z,y);*

A propriedade de número 1 garante que a distância entre dois pontos tem que ser igual, independente da ordem em que é calculada. Já a segunda propriedade, diz que a distância entre dois pontos deve pertencer ao intervalo de zero até o infinito, e que a distância entre dois pontos somente é zero se esses forem os mesmos pontos. A terceira propriedade merece um maior destaque, visto que a partir dela é possível diminuir os cálculos de distâncias, em busca por similaridade em Domínios Métricos, como afirma Bueno et al (2005).

Em domínios métricos, existem vários operadores de consulta por similaridade, mas dois se destacam:

**RQ(*sq, rq*) – Consulta por abrangência (“Similarity Range Query”):** retorna todos os objetos cujas distâncias até o objeto central da consulta *sq* sejam menores ou iguais ao raio da consulta *rq*;

**k-NNQ(*sq, k*) – Consulta aos k-vizinhos mais próximos (“k-Nearest Neighbor Query”):** retorna os *k* objetos mais próximos do objeto central de consulta *sq*;

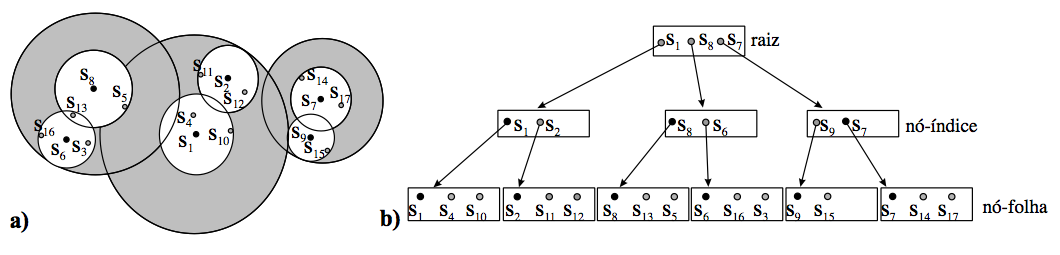
Para melhor compreendermos os Domínios Métricos precisamos estudar os Métodos de Acesso Métricos, visto que são eles os responsáveis pro representar computacionalmente os Domínios Métricos.

**6.1.2 Métodos de Acesso Métricos**

Como já mencionado acima, os Métodos de Acesso Métricos (MAMs) são estruturas de indexação. Segundo Bueno et al (2005) o ponto de partida para a criação dos MAMs foram os estudos de Burkhard e Keller em 1973, e só depois, em 1997, é que os MAMs dinâmicos foram criados com a M-*tree*, e em 2000 com Slim-*tree*.

Conforme Bueno et al (2005), depois destes trabalhos pioneiros vários outros MAMs foram criados, como é o caso da *Pyramid-tree* e a *P-Sphere tree* (*Probabilistic sphere tree*).

A implementação do algoritmo para resolução do problema da consulta por similaridade, pode utilizar qualquer um MAMs propostos. Será feito um estudo sobre qual estrutura de dados melhor complementa a representação do problema.

Figura 1: a) Representação de uma Slim-tree e b) sua estrutura lógica. Apud Bueno et al (2005).

A imagem acima, serve para exemplificar como seria a representação e a estrutura lógica do MAMs, na imagem vemos uma Slim-tree.

**6.2 Algoritmos Genéticos**

Os algoritmos genéticos (AG) do inglês *genetic algotithms* (*GA*) são uma técnica de busca extremamente eficiente no seu objetivo de varrer o espaço de soluções e encontrar soluções próximas da solução ótima. Seus mecanismos foram fortemente inspirados na teoria da evolução proposta por Charles Darwin em 1859, os primeiros cientistas a introduzir e disseminar os Algoritmos Genéticos foram John Holland e seu aluno David Goldberg em 1975 e 1989 respectivamente, conforme expõe Lacerda Estéfane e Carvalho André (ANO).

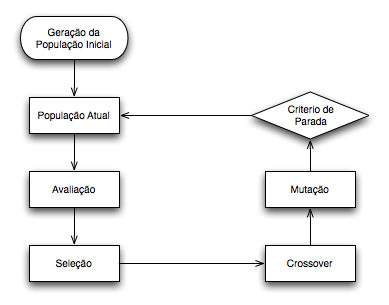
Alguns dos princípios dos algoritmos genéticos se baseiam na frase do livro **A Origem das Espécies** de Charles Darwin, que diz que “Quanto melhor um individuo se adaptar ao seu ambiente, maior será sua chance de sobreviver e gerar descendentes”, ou seja, quanto melhor ou mais apto for um individuo mais chances de sobreviver e de gerar descendentes ele vai ter.

Os algoritmos de otimização se caracterizam, por buscarem a melhor solução para um determinado problema, por isso os Algoritmos Genéticos se encaixam tão bem nessa classe de problemas. Conforme expõe Lacerda Estéfane e Carvalho André (ANO) os algoritmos genéticos consistem em tentar varias soluções e utilizar a informação obtida neste processo de forma a encontrar soluções cada vez melhores, por isso são tão eficientes em encontrar múltiplas soluções que são provavelmente as melhores para solucionar o problema.

Os ciclo de execução de um Algoritmo Genético possuem sete etapas bem definida:

Geração da População Inicial 🡪 População Atual 🡪 Avaliação da População Atual 🡪 Seleção para Próxima População 🡪 *Crossover* dos selecionados anteriormente🡪 Mutação dos novos indivíduos gerados 🡪 Critérios de Parada para saber se o ciclo se repete ou não 🡪 População Atual.

As etapas serão explicadas separadamente mais tarde. Uma imagem representativa pode ser visualizada logo abaixo.



FIGURA

**6.2.1 Terminologia**

Os algoritmos genéticos possuem muitos termos originados da biologia, porque, segundo Lacerda Estéfane e Carvalho André(ANO), eles são a metáfora dos processos de evolução e reprodução dos indivíduos mais adaptados ao ambiente. Durante a reprodução ocorrem processos de *crossover* e mutação. Para entendermos melhor esse paradigma, precisamos conhecer alguns dos principais termos encontrados no âmbito dos Algoritmos Genéticos.

* **Cromossomo e Genoma:** Representam um ponto no espaço de busca, são a representação da solução codificada em uma estrutura de dados, geralmente não passam de um vetor ou uma árvore;
* **Gene:** Nada mais são do que características ou atributos do problema em questão, codificados em uma posição ou nó do cromossomo;
* **Indivíduo:** Um membro da população. São formados por um ou mais cromossomos e sua aptidão;
* **Genótipo:** Representa a informação contida dentro dos cromossomos;
* **Fenótipo:** Representa o objeto construído a partir da informação dos do genótipo;
* **Alelo:** São os diferentes valores que um gene pode assumir;
* **Epistasia:** Interação ente genes do cromossomo, ou seja, quando um gene é influenciado por outro gene, “problemas com alta Epistasia são de difícil solução por algoritmos genéticos”, segundo Lacerda Estéfane e Carvalho André(ANO).

**6.2.2 Representação Real**

Quando foram idealizadas as representações dos problemas eram codificados utilizando dados binários (zeros e uns), mas existem casos em que uma representação real é mais cabível, por exemplo em problemas com parâmetros contínuos, matemáticos cuja a precisão de casas decimais é muito grande. A representação binária, nestes casos, chega a ser inviável, “pois para cada ponto decimal acrescentado na precisão, é necessário adicionar 3,3 bits na cadeia” (apud LACERDA Estéfane e CARVALHO André, ANO, pag). Em nosso trabalho, será realizado um estudo para melhor representação do problema, afim de que se alcance uma representação mais o simples possível.

**6.2.3 População Inicial**

A população inicial, como o próprio nome já diz, é a primeira população de indivíduos que os algoritmos genéticos trabalham. Elas podem ser geradas a partir de vários métodos. O primeiro desses métodos é o aleatório, ou seja, a cada execução do algoritmo uma nova e diferente população inicial é formada, e não se tem garantia de que essa população inicial seja igual a população inicial de outra execução. Outro método é de utilizar como população inicial soluções encontradas por outras técnicas de otimização, ou algumas soluções aceitáveis para o problema. Esse método é conhecido como “*seeding*”, segundo Lacerda Estéfane e Carvalho André(ANO).

**6.2.4 Função Objetivo**

Os problemas do mundo real são bastante complexos na sua maioria, fazendo com que a função objetivo seja também bastante complexa e com alto custo dos recursos computacionais. É importante simplificar ao máximo a função objetivo, mas essa simplificação trará perdas à informação na solução final. Para reduzirmos o tempo computacional, é interessante utilizarmos uma simplificação da função objetivo nas primeiras gerações de indivíduos e deixarmos a função objetivo real para as últimas gerações. Uma outra técnica bastante difundida para a redução dos recursos computacionais é utilizar os algoritmos genéticos para localizar a região da função objetivo mais promissora a ser o máximo da função e depois utilizar um algoritmo de Subida de Encosta para localizar a possível melhor solução. Isso porque os métodos de Subida de Encosta são extremamente eficientes na obtenção do ponto de máximo local.

**6.2.5 Critérios de Parada**

Com relação aos critérios de parada, observamos que estes são responsáveis por fazer com que o Algoritmo Genético não fique evoluindo infinitamente. Para cada tipo de problema pode ser utilizado um critério de parada diferente. Em alguns casos pode se priorizar o tempo ou os recursos computacionais, em outros a quantidade de gerações, ou então a chegada ao valor ótimo da função (somente se conhecermos esse valor). E, por último, o algoritmo poderia parar sua execução quando a convergência estiver acima de um certo ponto. A convergência ocorrerá quando a aptidão dos indivíduos de uma determinada população não melhorar por um dado número de gerações.

**6.2.6 Seleção**

Baseando-se no processo de seleção natural da biologia, o Algoritmo Genético seleciona os melhores indivíduos da população inicial, ou seja, os que possuem uma maior aptidão, para povoarem a próxima geração e para criarem novos indivíduos filhos (parecidos com seus pais). Os operadores de *crossover* e mutação são responsáveis pela criação dos novos indivíduos que irão povoar a próxima geração. Existem vários métodos de seleção, mas dois deles se destacam:

**6.2.6.1 Roda da roleta**

A seleção por Roda da Roleta utiliza uma população intermediária conhecida como *mating pool*. A população intermediária serve para alocar os cromossomos pais selecionados. Os pais são selecionados através da probabilidade proporcional a sua aptidão. Segundo Lacerda Estéfane e Carvalho André(ANO) , “a probabilidade de seleção *p*i , de um cromossomo *s*i com aptidão *f*i é dada por:

*![](data:application/pdf;base64,)*”

A seleção por Roda da Roleta é realizada calculando uma coluna de aptidão acumulada. Depois, gera-se um número aleatório *r* no intervalo [0, Soma Total], cuja Soma Total é a soma de todas as aptidões. Por último, o indivíduo selecionado é o primeiro que possuir a aptidão acumulada maior que *r*. Os passos são repetidos até toda a população intermediária ter sido preenchida.

**6.2.6.1 Seleção por Torneio**

A seleção por torneio forma a população intermediária da seguinte maneira; 1) são escolhidos aleatoriamente e com probabilidade iguais *n* cromossomos da população; 2) dados os cromossomos selecionados anteriormente, é escolhido o mais adaptado dentre eles, por isso a nomenclatura “torneio”. O valor mais utilizado para *n* é de *n = 2*. Uma grande vantagem da seleção por torneio é o fato de “não precisa de escalonamento da aptidão e nem de ordenamento” conforme proposto por Lacerda Estéfane e Carvalho André(ANO, p 131), ou seja, a seleção por torneio é processada sem nenhuma modificação na população, o que reduz o consumo dos recursos computacionais.

Em uma outra abordagem, a seleção por torneio utiliza probabilidades diferenciadas. Dados *n* cromossomos para participarem do torneio, o primeiro cromossomo a ganhara o torneio com probabilidade *q*, o segundo com probabilidade *q(1-q)*, terceiro com *q(1-q)2*, e assim por diante ate toda população intermediaria tenha sido preenchida.

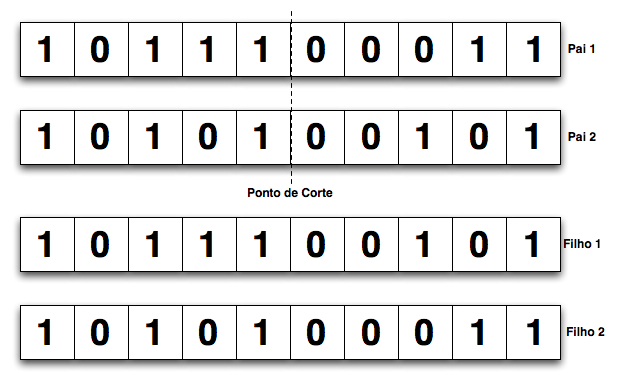
Ainda segundo a teoria proposta por Lacerda Estéfane e Carvalho André(ANO, p 131 e p132), duas conclusões importantes podem ser tiradas:

* “Vale notar que se *n = N*, em que *N* é o tamanho da população, tal seleção é equivalente à seleção com ordenamento exponencial”
* “Aumentando o número *n* de cromossomos do torneio ou a probabilidade *q* do primeiro cromossomo vencer, aumenta-se a pressão de seleção, isto é, cromossomos com aptidão acima da media terão mais chances de serem selecionados”.

**6.2.7 *Crossover***

Juntamente com o operador de mutação, o operador de *crossover* são as principais funções de busca dos Algoritmos Genéticos na exploração de regiões desconhecidas do espaço de busca. As operações de *crossover*, ocorrem em um par de cromossomos obtidos a partir da população intermediaria, e geram dois novos cromossomos, nomeados de cromossomos filhos.

Os cromossomos filhos tem seus genes herdados dos cromossomos pais, sendo a cadeia de genes constituída por partes das cadeias de genes dos cromossomos pais. A cadeia de genes dos pais é cortada em uma posição aleatória sendo que o primeiro cromossomo filho recebe a primeira e a segunda parte do primeiro e do segundo cromossomos pais respectivamente. Já o segundo cromossomo filho, deverá receber a segunda parte do primeiro cromossomo pai e a primeira parte do segundo cromossomo pai. A figura abaixo melhor representa o comportamento do operador de *crossover.*

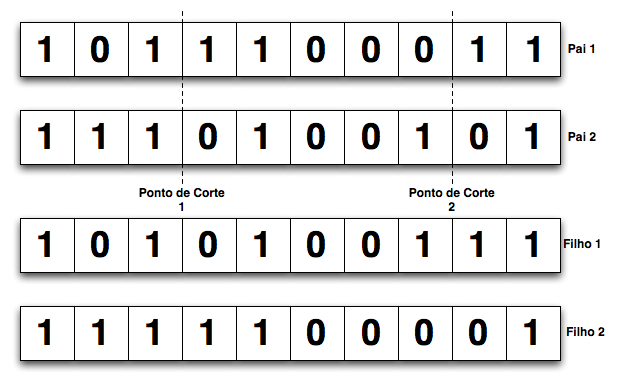
**

*FIGURA*

O *crossover* é aplicado com base em uma probabilidade a cada par de cromossomos selecionados. A taxa de *crossover*, pode variar de 60% a 90% segundo Lacerda Estéfane e Carvalho André(ANO). Caso não ocorra o *crossover* os filhos serão iguais os pais, permitindo assim que algumas soluções para o problema sejam preservadas. Existem varias vertentes relacionadas ao ponto de corte dos operadores de *crossover*. Discutiremos sobre elas logo abaixo.

**6.2.7.1 *Crossover* Multiponto**

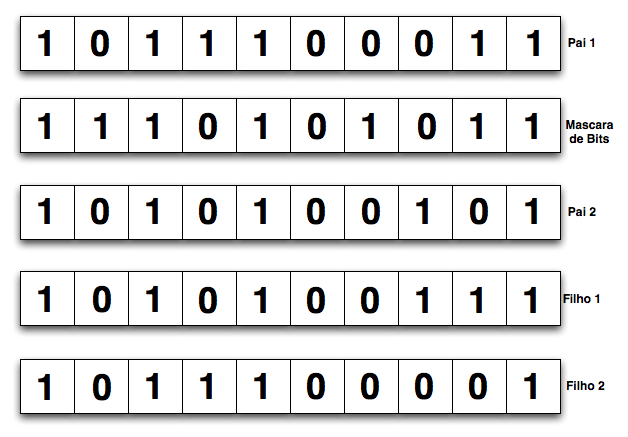
O *crossover* multiponto também conhecido como *crossover* de *N* pontos, já foram descritos na seção anterior, sendo o *crossover* de 1 ponto apresentado. O de 2 e 4 pontos são apresentados nas figuras logo abaixo. A principal diferença entre os operadores de *crossover* de 1, 2 e 4 pontos, está na quantidades de pontos de corte que aparecem na cadeia de genes no momento da permutação de características.



FIGURA

**6.2.7.2 *Crossover* Uniforme**

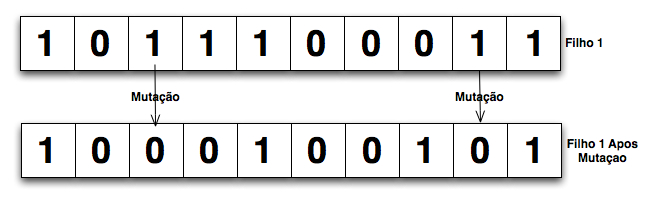
No *crossover* uniforme para cada par de cromossomos pais é criada uma mascara de bits aleatórios. Essa mascara de bits não passa de um cadeia de tamanho igual ao da cadeia de genes dos cromossomos e possui somente zeros e uns como valores armazenadas em cada posição da cadeia. Se o primeiro bit da mascara for 1 então o filho1 recebera o primeiro gene do pai1 senão recebera o primeiro gene do pai2, e assim por diante, até que toda a cadeia de gene do filho tenha sido preenchida. Já o segundo filho recebera o processo oposto ao do filho1, como descrito na figura abaixo.



FIGURA

**6.2.8 Mutação**

Apos aplicado a operação de *Crossover*, a operação de mutação é aplicada. Dada uma probabilidade entre 0.1% a 5% sobre cada gene dos cromossomos filhos gerados pelo operador de *crossover*. A operação de mutação inverte o valor dos genes, isso quer dizer que, se um gene possuir o valor 0 e ocorrer mutação ele passara a armazenar o valor 1, a figura abaixo explica e exemplifica a operação de mutação.



FIGURA

“A mutação melhorar a diversidade dos cromossomos na população, no entanto por outro lado, destrói informação contida no cromossomo” (apud LACERDA Estéfane e CARVALHO André, ANO, pag 97). Isso que dizer que devemos ter cuidado ao aplicarmos o operador de mutação sobre os cromossomos e que as taxas de probabilidade devem ser pequenas (0.1% a 5%).

**6.2.9 Elitismo**

A cada interação do algoritmo genético uma nova população de soluções candidatas à resolução do problema é formada. Nessa nova população podemos perder informações valiosas, como a melhor solução já encontrada para o problema. Isso ocorre devido às operações de *crossover* ou mutação. Ainda segundo Lacerda Estéfane e Carvalho André(ANO pag) “é interessante transferir o melhor cromossomo de uma geração para a outra”. Finalmente quando mantemos a melhor solução (cromossomo) da população anterior na nova população, dizemos que o algoritmo é elitista.

**7 Cronograma**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Atividades** | **Janeiro** | **Fevereiro** | **Março** | **Abril** | **Maio** | **Junho** |
| Estudo da Teoria Envolvida | **X** | **X** |  |  |  |  |
| Estudo sobre a Representação do Problema |  | **X** |  |  |  |  |
| Implementação dos Algoritmos |  | **X** | **X** | **X** |  |  |
| Suíte de Testes |  |  |  | **X** |  |  |
| Analise dos Resultados |  |  |  |  | **X** |  |
| Elaboração da Monografia |  |  | **X** | **X** | **X** | **X** |
| Defesa |  |  |  |  |  | **X** |

**Referências**

BUENO Renato, TRAINA Agma J. M. e TRAINA JR Caetano. **Algoritmos Genéticos para Consultas por Similaridade**. 2005.

ZUBEN Fernando J. Von. **Computação Evolutiva:** Uma Abordagem Pragmática.

GOLDBERG David E. ***Genetic Algorithms in Search, Optimization, and Machine Learning****.* Addison-Wesley Company Inc. 1989.

BÄCK Thomas, FOGEL David B e MICHALEWICZ Zbigniew. **Evolutionary Computation 1:** Basic Algorithms and Operators. Institute of Physics Publishing. 2000.

RUSSEL Stuart e NORVIG Peter. **Inteligência Artificial**. 2˚ Ed. Rio de Janeiro: Editora Campus. 2004.

LACERDA Estéfane e CARVALHO André. **Introdução aos Algoritmos Genéticos**. Capítulo 3.

YAMAROTO Cláudio Haruo. **Uso de Lógica Nebulosa na Construção e na Utilização da Arvore Métrica *Slim-tree.*** *2002.*