CENTRO DE ENSINO UNIFICADO DE TERESINA – CEUT
CURSO DE CIÊNCIAS DA COMPUTAÇÃO

DANN LUCIANO DE MENEZES

ALGORITMOS GENETICOS NA OTIMIZAÇÃO DE CONSULTAS POR SIMILARIDADE

TERESINA
2010

DANN LUCIANO DE MENEZES

ALGORITMOS GENETICOS NA OTIMIZAÇÃO DE CONSULTAS POR SIMILARIDADE

Monografia, apresentada ao Centro de Ensino Unificado de Teresina – CEUT como um dos pré-requisitos para a obtenção do grau de bacharel de Ciências de Computação

Ronildo Pinheiro de Araújo Moura

TERESINA

2010

DANN LUCIANO DE MENEZES

ALGORITMOS GENETICOS NA OTIMIZAÇÃO DE CONSULTAS POR SIMILARIDADE

Monografia apresentada como exigência parcial para a obtenção do grau de Bacharel em Ciências da Computação, na area de concentração....., à banca examinadora do Curso de Ciências da Computação, do Centro de Ensino Unificado de Teresina – CEUT

Aprovada em _____/_____/_____

BANCA EXAMINADORA

_____

Ronildo Pinheiro de Araújo Moura

CEUT

_____

Nome do componente

Instituição

_____

Nome do componente

Instituição

Aos meus ...

AGRADECIMENTOS

A Deus, inteligência suprema do Universo,

"*Muito Tempo eu Levei Para Entender que Nada Sei*"
Dias de Luta, Ira!

RESUMO

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Fusce at erat dolor. Phasellus iaculis lorem id sapien vulputate accumsan. Sed vel erat augue. In hac habitasse platea dictumst. Etiam ac mauris nec arcu blandit facilisis. Mauris fermentum arcu nec libero iaculis malesuada. Phasellus sed sapien justo, non lacinia enim. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Duis tincidunt nulla eget magna placerat in facilisis turpis aliquet. Donec diam justo, volutpat faucibus sagittis id, rutrum quis urna. Curabitur quis odio nunc, sed viverra mauris. Pellentesque dictum ullamcorper vestibulum. Cras pulvinar tellus eget tortor dapibus vel porta nunc pulvinar. Sed pulvinar imperdiet sagittis. Maecenas id mi nunc, vitae ullamcorper elit.

PALAVRAS-CHAVE: Consultas por Similaridade, Algoritmos Genéticos, Otimização

ABSTRACT

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Fusce at erat dolor. Phasellus iaculis lorem id sapien vulputate accumsan. Sed vel erat augue. In hac habitasse platea dictumst. Etiam ac mauris nec arcu blandit facilisis. Mauris fermentum arcu nec libero iaculis malesuada. Phasellus sed sapien justo, non lacinia enim. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Duis tincidunt nulla eget magna placerat in facilisis turpis aliquet. Donec diam justo, volutpat faucibus sagittis id, rutrum quis urna. Curabitur quis odio nunc, sed viverra mauris. Pellentesque dictum ullamcorper vestibulum. Cras pulvinar tellus eget tortor dapibus vel porta nunc pulvinar. Sed pulvinar imperdiet sagittis. Maecenas id mi nunc, vitae ullamcorper elit.

KEYWORDS:

SUMÁRIO

1 INTRODUÇÃO

2 CONSULTAS POR SIMILARIDADE

2.1 CONSIDERACÕES INICIAIS

Os primeiros Sistemas Gerenciadores 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 um vetor de características.

Três perguntas nos ajudam e nos guiam no entendimento de consultas por similaridade. Elas são nomeadas neste trabalho como perguntas guias e são expostas logo abaixo:

1. Como comparar Dados Complexos?
2. Como expressar consultas sobre Dados Complexos?
3. Como indexar Dados Complexos?

A próxima seção expõem os Domínios Métricos que são a resposta a primeira pergunta guia.

2.2 DOMÍNIOS MÉTRICOS

Um Domínio Métrico como já mostrado na seção anterior, respondem a primeira pergunta guia e através deles podemos comparar Dados Complexos. Domínios Métricos são definidos 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 $z \in D$, uma função de distância (métrica) $d : D\ X\ D \rightarrow R+$ deve satisfazer as seguintes propriedades:

1. Simetria: $d(x,y) = d(y,x)$;
2. Não Negatividade: $0 \leq d(x,y) < \infty$, $d(x,x) = 0$;
3. Desigualdade Triangular: $d(x,y) \leq d(x,z) + d(z,y)$;

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).

Para cada tipo de aplicação uma função de distancia deve ser definida, fazendo com que um problema seja delimitado por um Domínio Métrico. Na seção logo abaixo será discutido sobre o numero de dimensões que um Domínio Métrico pode esta definido.

2.2.1 Domínios Métricos e suas Dimensões

Um domínio métrico pode possuir n dimensões. Onde cada uma das dimensões representam uma característica do dado complexo. Por exemplo se quiséssemos armazenar em um domínio métrico uma imagem com apenas duas características nos precisaríamos de duas dimensões como mostrado na imagem logo em seguida. As características utilizadas são relativas e podem variar dependendo do domínio do problema.

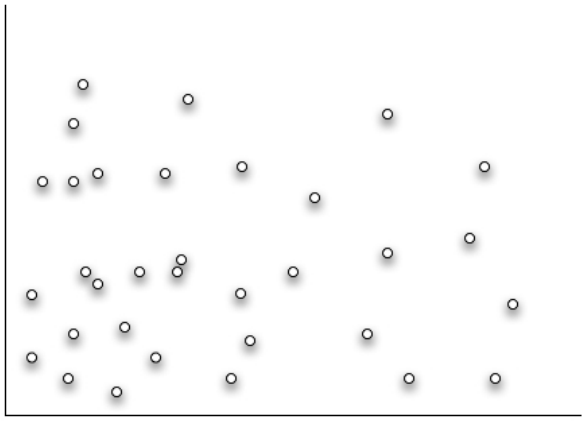

Figura 2.1

A distância entre dois pontos em um domínio métrico que possui somente suas dimensões mapeadas em pontos, é geralmente calcula pela distância euclidiana conforme expõe (NADVORNY 2005). A relação entre o tamanho e *n* e a complexidade da função de distância é diretamente proporcional, ou seja, quanto maior for o tamanho de *n* mais complexo a função de distância se tornará. Como já explicado acima para cada problema deverá ser criada uma função de distância, que represente a distancia entre dois pontos pertencentes a esse domínio. A imagem abaixo representa um domínio métrico com *n* igual a três dimensões.

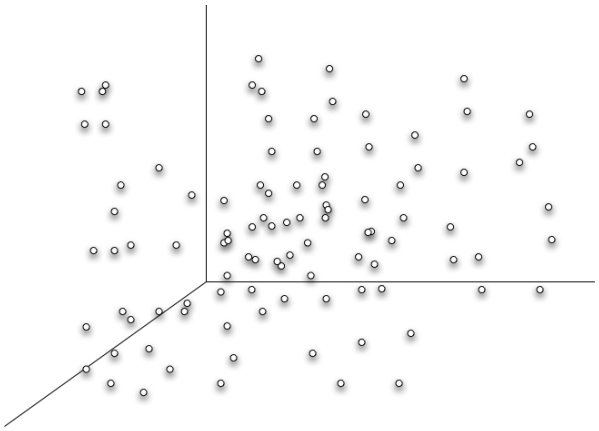

Figura 2.2

No próximo tópico será destinado a responder a segunda pergunta guia.

2.3 OPERADORES POR SIMILARIDADE

Os operadores por similaridade, respondendo a segunda pergunta guia, são responsáveis por expressarem consultas nos dados complexos pertencentes a um Domínio Métrico. 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.

Como já explicado acima, existem outros operadores por similaridade, mas a grande maioria deles derivam dos operadores citados acima. Para melhor compreendermos as consultas por similaridade precisamos estudar os Métodos de Acesso Métricos, visto que são eles os responsáveis pro indexar dados complexos pertencentes a um Domínio Métrico.

2.4 MÉTODOS DE ACESSO

Os métodos de acesso são responsáveis por indexar os Dados Complexos pertencentes a um determinado Domínio Métrico em uma estrutura de dados. Existem vários métodos de acesso, e eles se classificam conforme descrito logo abaixo:

- Métodos de Acesso Pontual: quando mapeamos o objeto armazenado em pontos do espaço;

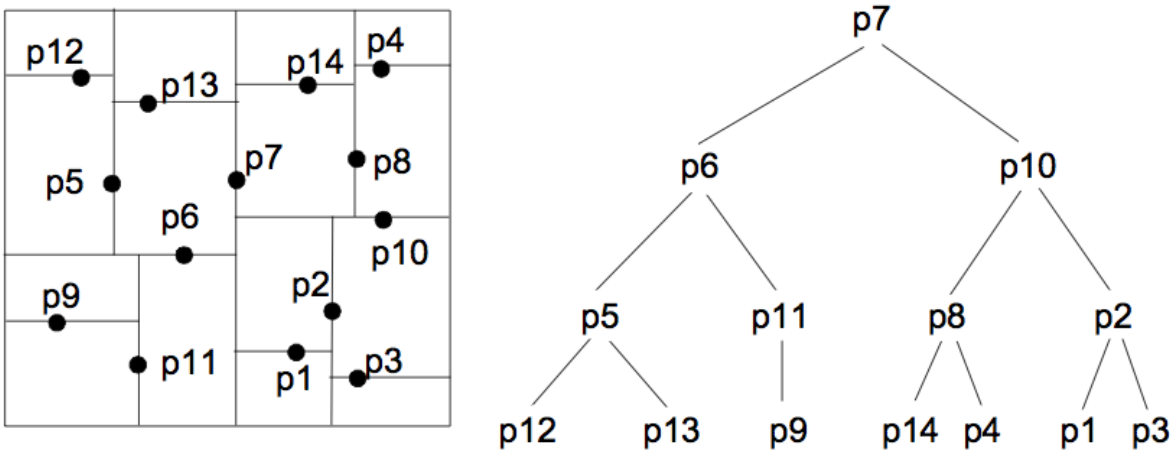

Figura 2.3

- Métodos de Acesso Espacial: nem sempre o mapeamento de objetos em pontos pode ser viável, em dados complexos como dados geográficos que contem polígonos em geral. Neste caso os dados indexados são as imagens vetorizadas;

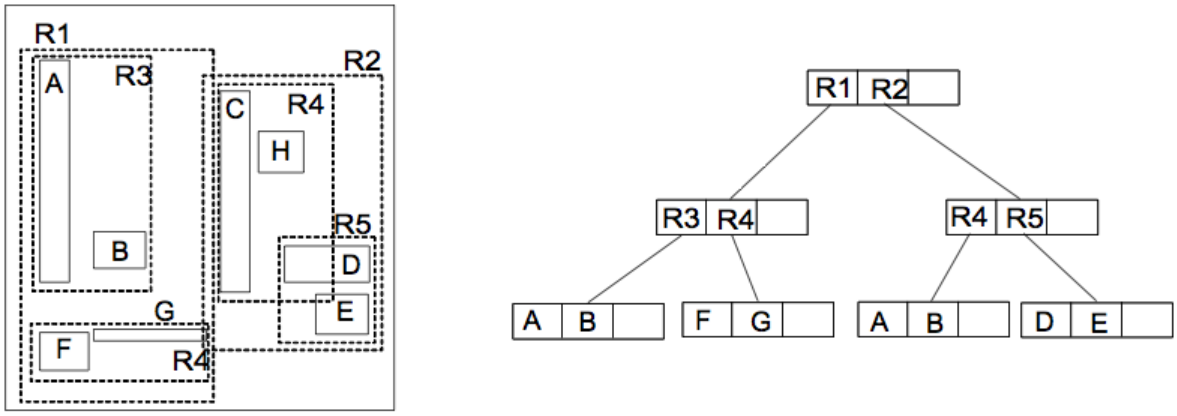

Figura 2.4

- Métodos de Acesso Métrico: os dois métodos anteriores possuem seus objetos mapeados em uma posição absoluta, ou seja, existe um ponto central (normalmente 0) no espaço, onde todos os objetos utilizam ele para se referenciar. Então os métodos de acesso métrico são aplicados nos casos onde não são possíveis de se utilizar uma posição absoluta, mais ainda sim existe uma noção de distância.

2.4.1 MÉTODOS DE ACESSO MÉTRICO

Conforme expõe Nadvorny (2005) as primeiras estruturas de dados datam de 1973 e são denominadas de arvores Burkhard-Keller. Ao longo do tempo varias outras arvores foram apresentadas com melhorias significativas comparadas com as primeiras que surgiram. Por exemplo as arvores VP (*Vantage Point*), GH (*Generalized Hyperplane*). As arvores M em especial merece um maior destaque visto que esta possui características que otimizam as operações não só internas da arvores mais também operações relacionadas a memória segundaria. Outro fator que merece um melhor detalhamento sobre as arvores M é que ela influencia as principais estruturas de indexação de dados similares existentes na atualidade.

2.4.1.1 ÁRVORES M

Como já descrito acima as árvores M introduziram características que a colocaram-na a frente de seus antecessores como relata (NADVORNY 2005). A primeira das importantes características é em relação ao **Uso de Memória Secundária**, onde a arvore possui nos com tamanhos fixos, permitindo assim o armazenamento como paginas de discos. A segunda característica presente nas arvores M é chamada de **Dinamismo** que consiste do fato de operações de inserção ou remoção não são tão custosas e não implicam em uma reorganização drástica da arvore. A terceira característica não menos importante se diz respeito a **Escalabilidade**, onde o custo da busca não aumenta exponencialmente quando novos objetos são inseridos, pois uma arvore M esta sempre balanceada. Isto ocorre devido a arvore ser construída de baixo para cima (*Bottom-Up*).

Os nos folha da arvore ou contem os objetos indexados ou apenas seus respectivos endereços (Ponteiros) os restantes dos nos possuem um endereço para uma sub-arvore, o raio de cobertura do no atual e a distancia métrica do no atual ate seu respectivo pai. As imagens abaixo representação a estrutura de uma arvore M (a) e a disposição dos objetos em um espaço métrico.

2.4.1.2 SLIM-TREE

A Slim-tree é um Método de Acesso Métrico derivada da M-tree e possui as mesmas características discutidas no tópico 2.4.1.1 como o fato de serem balanceadas, dinâmicas e desenvolvidas para indexar objetos pertencentes a um Domínio Métrico. Quando comparadas sobre as mesmas condições a Slim-tree sempre sobrepujou as M-tree, tanto em termos de número de acesso a disco quanto em termos de número de distância calculadas para responder consultas por abrangência conforme expõe BUENO(Ja existe um BUENO)(2001). Ela foi desenvolvida e exposta por Traina Jr et al no ano de 2000.

Assim como nas arvores B+ a Slim-tree possui todos seus objetos armazenados nas folhas da arvore e todos os outros nos são apenas nos de caminhamento ou index.

2.4.1.2.1 ORGANIZAÇÃO DA SLIM-TREE

2.4.2.2.2

2.4.2.2.3

2.4.2.2.3

2.5 CONSIDERAÇÕES FINAIS

Neste capitulo foi exposto o conceito e as definições de consultas por similaridade, mais especificamente o que seria um Domínio Métrico, os tipos de cultas por similaridade e os Métodos de Acesso Métrico. No próximo capitulo será apresentado os Algoritmos Genéticos bem como sua implicações na otimização de problemas.

4 ALGORITMOS GENÉTICOS

4.1 CONSIDERAÇÕES INICIAIS

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 e Carvalho (2002).

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 indivíduo se adaptar ao seu ambiente, maior será sua chance de sobreviver e gerar descendentes", ou seja, quanto melhor ou mais apto for um indivíduo 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 e Carvalho (2002) 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:

1. Geração da População Inicial
2. População Atual
3. Avaliação da População Atual
4. Seleção de Indivíduos para compor a próxima População
5. *Crossover* dos Indivíduos selecionados anteriormente
6. Mutação é aplicada nos novos indivíduos gerados

7. Critérios de Parada para saber se o ciclo se repete ou não.

As etapas serão explicadas separadamente mais tarde. A figura 4.1 representa o ciclo de um Algoritmo Genético.

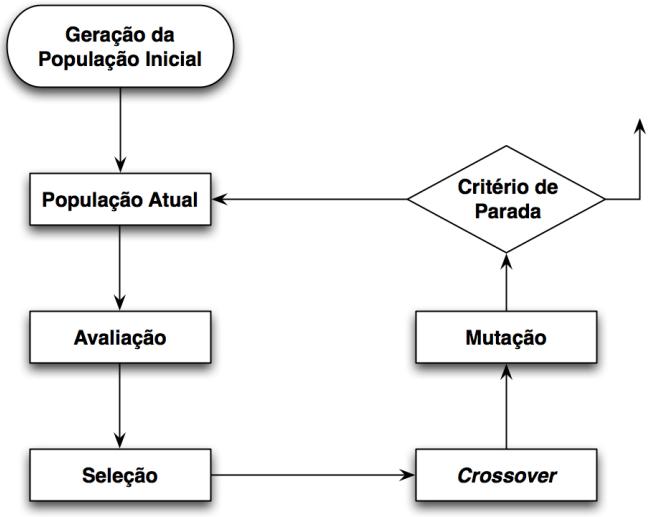

Figura 4.1

4.2 TERMINOLOGIA

Os algoritmos genéticos possuem muitos termos originados da biologia, porque, segundo Lacerda e Carvalho (2002), 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;

4.3 REPRESENTAÇÃO DAS SOLUÇÕES

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" (LACERDA e CARVALHO, 2002, p-107). A representação real em certos casos mostrasse bastante simples e robusta na resolução do problema. Na figura 4.2 e 4.3 temos um exemplo de como seria um conjunto de indivíduos em ambas representações binaria e real respectivamente.

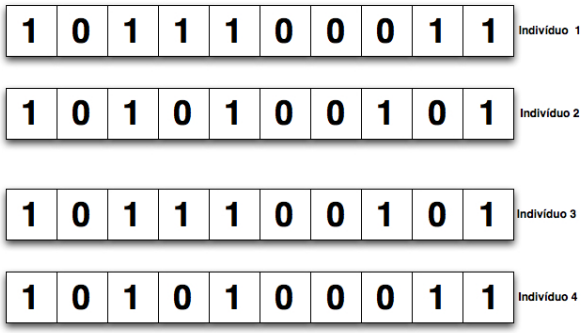

Figura 4.2

As duas representações mais difundidas são a binaria e a real, mas na realidade qualquer outro tipo de estrutura de dados(lista ligada, pilha, fila ou árvores) podem ser utilizada. A escolha de que tipo de representação escolher vai depender da natureza do problema proposto para resolução.

Neste trabalho foi feito uma analise sobre qual a melhor forma de representar as soluções para o problema dentro do cromossomo de cada indivíduo.

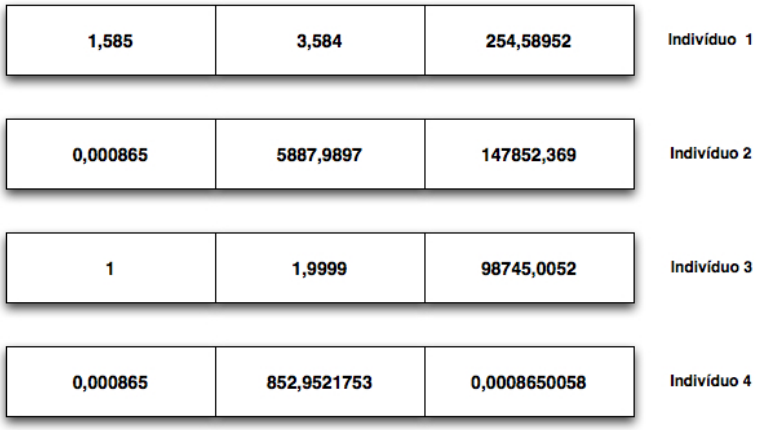

Figura 4.3

4.3.1 INFACTIBILIDADE

Um dos grandes problemas das representações da soluções está no conjunto de todos dos indivíduos que codificam uma solução inexistente. Os elementos pertencentes a este conjunto são chamados e infactíveis. A infactibilidade está presente na grande parte dos problemas do mundo real. Existem diversas formas de se tratar a infactibilidade de um elemento, neste trabalho relatamos as três mais utilizadas.

A primeira forma é simplesmente ignorando a infactibilidade pois é muito provável que pelo menos um elemento da população indivíduos seja uma solução utilizável (factível). A segunda maneira verifica se um determinado indivíduo é uma solução factível, senão for esse elemento é descartado e um outro indivíduo é criado. Dependendo do problema a segunda forma pode deixar nosso algoritmo quase que em *loop* infinito, pois dependendo do nosso espaço de solução muitos indivíduos infactíveis podem ser gerados. A terceira forma é denominada de reparação pois para cada indivíduo infactível é aplicado uma heurística, com a intenção de transformar a solução infactível em uma solução factível.

4.3.2 PRINCIPIO DA LOCALIDADE

Outro grande problema na representação das soluções está ligada diretamente com

o espaço de busca e as representações das soluções. Toda representação possui dois espaços de busca, o do genótipo e o do fenótipo. O problema existe quando uma pequena modificação do genótipo pode gerar uma mudança drástica do fenótipo. O problema ocorre também quando uma grande modificação do genótipo gera uma mudança suave do fenótipo. A figura 4.4 melhor exemplifica o problema. No indivíduo 1 alterando apenas um único bit é gerado um indivíduo (2) com o fenótipo muito menor comparado com o indivíduo 1. Já alterando os 8 bits do indivíduo 3 é gerado um indivíduo (4) com o fenótipo quase inalterado.

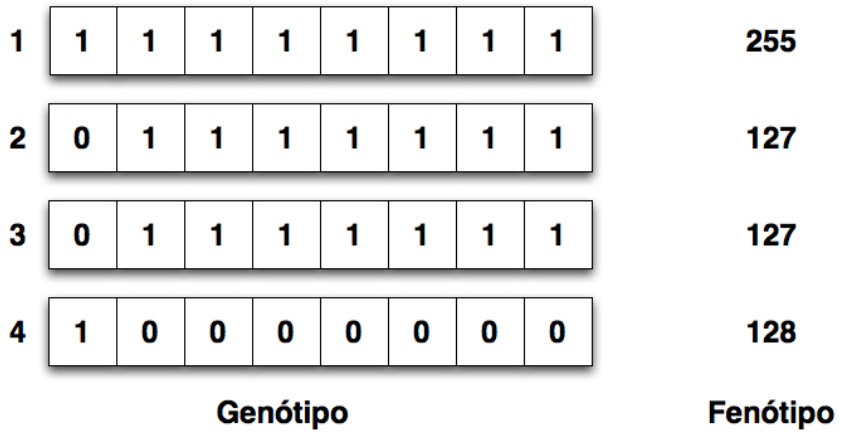

Figura 4.4

4.4 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. Nem sempre existe um método bom para se gerar uma nova população, pois a geração de indivíduos esta diretamente relacionada com o espaço de solução do problema tornando em muitos casos, a geração da população inicial uma dificuldade a ser resolvida especificamente para o problema em questão. 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 e Carvalho (2002).

4.5 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.

4.6 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.

4.7 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:

4.7.1 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 e Carvalho (2002) , "a probabilidade de seleção p_i , de um cromossomo s_i com aptidão f_i é dada por:

$$fi / \sum_{i=1}^{N} fi \quad "$$

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.

O método de seleção roda da roleta foi o primeiro método desenvolvido e possui uma serie de problemas. Nas próximas seções são expostos os dois problemas mais difundidos e estudados bem como as técnicas para contorna o problema.

4.7.1.1 AVALIAÇÃO NEGATIVA

Se existir um um ou mais indivíduos da população com avaliação negativa, nunca

será gerado um numero aleatório que represente a posição em que o indivíduo ocupa na população, e portanto ele nunca será selecionado para compor a próxima população.

Para resolver esse problema basta somar uma constante maior que zero ao valor da função de avaliação. O valor da constante deve ser escolhido com base no maior valor negativo que a função de avaliação pode gerar.

4.7.1.2 SUPER INDIVIDUO

Um super indivíduo é um indivíduo que possui uma avaliação muito superior a media do resto da população. Tomando como base a tabela 4.1 o indivíduo x5 será selecionado 99% das vezes, e pode ser que uma geração inteira de pais se resuma a este indivíduo.

Indivíduo	Avaliação
X1	20
X2	30
X3	40
X4	10
X5	9900

Tabela 4.1

Se uma população se resumir a um único indivíduo após uma seleção, acabando com a divergência da população, o algoritmo genético ira possuir uma convergência genética prematura.

4.7.2 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 e Carvalho (2002, 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 ganhará o torneio com probabilidade *q*, o segundo com probabilidade *q(1-q)*, o terceiro com $q(1-q)^2$, e assim por diante até toda população intermediária tenha sido preenchida.
Ainda segundo a teoria proposta por Lacerda e Carvalho (2002, p 131-132), 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 média terão mais chances de serem selecionados”.

4.8 OPERADORES GENÉTICOS

4.8.1 CROSSOVER

O operador de mutação juntamente com 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 intermediária, e geram dois novos cromossomos, nomeados de cromossomos filhos.

Os cromossomos filhos têm 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*.

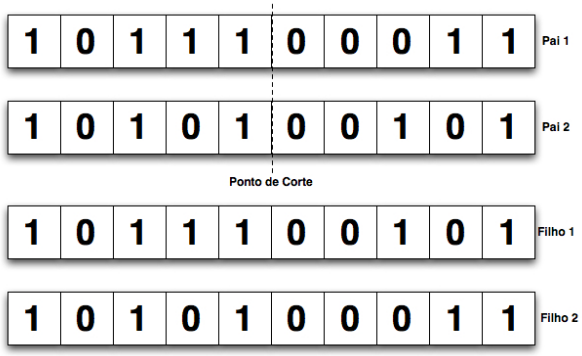

Figura 4.

O *crossover* multiponto também conhecido como *crossover* de *N* pontos, já foi descrito na seção anterior, sendo o *crossover* de 1 ponto apresentado. Os 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.

4.8.1.1 CROSSOVER DE UM PONTO

4.8.1.2 CROSSOVER DE DOIS PONTOS

4.8.1.3 CROSSOVER UNIFORME

No *crossover* uniforme, para cada par de cromossomos pais é criada uma máscara de bits aleatórios. Essa máscara de bits não passa de uma cadeia de tamanho igual ao da cadeia de genes dos cromossomos e possui somente zeros e uns como valores armazenados em cada posição da cadeia. Se o primeiro bit da mascara for 1 então o filho1 recebera o primeiro gene do pai1. Em caso contrario 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.

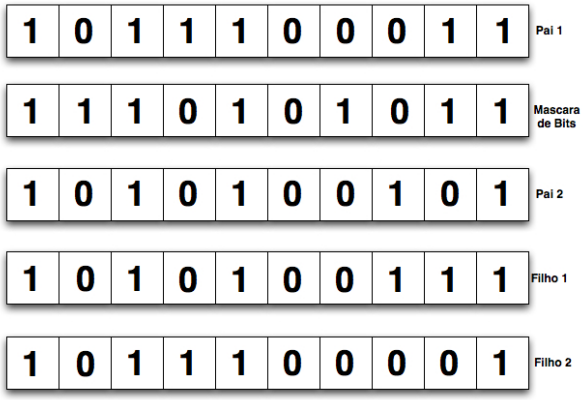

Figura 4.

4.8.2 MUTAÇÃO

Após aplicado a operação de *Crossover*, a operação de mutação é aplicada. Dada uma probabilidade normalmente 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.

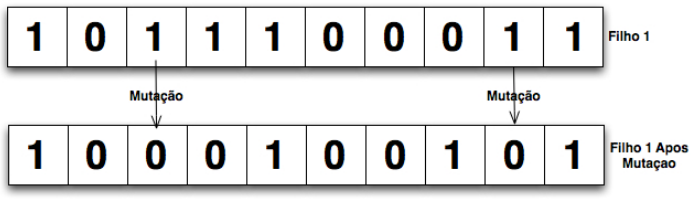

Figura 4.

"A mutação melhora a diversidade dos cromossomos na população, por outro lado, destrói informação contida no cromossomo" (LACERDA e CARVALHO, 2002, p97). Isso que dizer que devemos ter cuidado ao aplicar o operador de mutação sobre os cromossomos, pois forem muito altas o Algoritmo Genético se comportara como uma busca aleatória e se a taxa for muito baixa o operador de mutação não conseguira gerar diversidade nos cromossomos. Na grande maioria dos exemplo as taxas de probabilidade variam entre

0.1% a 5%.

4.9 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 e Carvalho (2002 p 99) "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.

4.10 CONSIDERACÕES FINAIS

Neste capitulo vimos a origem e o funcionamento dos algoritmos genéticos e o porque desta técnica evolutiva ter recebido tantas pesquisas, principalmente na otimizações de algoritmos computacionais modernos. No capitulo 5 veremos como integrar os algoritmos genéticos para otimizarem o processo de consulta por similaridade aproximada.

5 SOLUÇÃO

6 RESULTADOS

7 CONCLUSÕES

REFERÊNCIAS

BÄCK Thomas, FOGEL David B e MICHALEWICZ Zbigniew. ***Evolutionary Computation 1:*** b*asic algorithms and operators*. Institute of Physics Publishing. 2000.

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

CHINO Fabio Jun Takada. **Visualizando a Organização e o Comportamento de Estruturas Métricas: aplicações em consultas por similaridade**. 2004.

Ferreira Mônica Ribeiro Porto. **Suporte a Consultas por Similaridade Unárias em SQL**. 2008.

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

JARDINI Evandro, GONZAGA Adilson e TRAINA JR Caetano. **Metodologia para Indexação de Busca de Impressões Digitais através do uso de Função de Distancia Métrica**. 2006.

LACERDA Estéfane e CARVALHO André. **Introdução aos Algoritmos Genéticos**. Capítulo 3. Revista de Informática Teórica e Aplicada, v.9, p.7 - 39, 2002.

NADVORNY César Feijó. **TM-*tree*:** um método de acesso para consultas por similaridade. 2005.

PACHECO Marco**. Algoritmos Genéticos:** princípios e aplicações. 1999

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

SILVA Maria, BORGES Eduardo e GALANTE Renata. **XSimilarity:** Uma ferramenta para Consultas por Similaridade embutidas na Linguagem Xquery. 2008.

TRAINA JR Caetano, TRAINA Agma, SEEGER Bernhard e FALOUTSOS Christos. ***Slim-trees: High Performace Metric Tree Minimizing Overlap Between Node***s. 1999.

VIERA MARCOS, TRAINA JR Caetano, CHINO Fabio e TRAINA Agma. ***DBM-Tree:*** *a dynamic metric access method sensitive to local density data*. 2004.

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

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