Algoritmos Genéticos para Consultas por Similaridade Aproximadas *

Renato Bueno , Agma J. M. Traina , Caetano Traina Jr

Instituto de Ciências Matemáticas e de Computação - ICMC
Universidade de São Paulo - USP
Av. Trabalhador São-carlense, 400 - Centro
13560-970 - São Carlos - SP

{rbueno|agma|caetano}@icmc.usp.br

Abstract. *The similarity search process on complex domains for exact answer is an expensive process. However, the multimedia data comparison operations usually consider some extracted features from the elements, instead of the elements themselves. Therefore, trading exact answering with query time response can be a worth exchange. In this work we developed two techniques based on genetic algorithms to allow retrieving approximate data indexed in a Metric Access Methods (MAMs) within a limited, user-defined, amount of time. Experimental evaluation shows that good results can be obtained in a fraction of the time required to obtain the exact answer.*

keywords: Approximate similarity queries, genetic algorithms, metric access methods.

Resumo. *O custo do acesso exato a dados complexos tende a ser alto. Além disso, a operação de busca não é efetuada realmente sobre os dados originais, mas sobre características extraídas desses dados. Portanto, em muitas aplicações, trocar a exatidão das respostas por um melhor desempenho pode ser muito vantajoso. Neste trabalho foram desenvolvidas duas técnicas de recuperação aproximada de dados em domínios métricos utilizando algoritmos genéticos, cujo refinamento das respostas é dependente do tempo de processamento disponível, definido pelo usuário. Experimentos realizados mostraram ser possível obter bons resultados em apenas uma fração do tempo necessário para a obtenção da resposta exata.*

palavras-chave: Consultas por similaridade aproximadas, algoritmos genéticos, métodos de acesso métricos

1. Introdução

Os sistemas gerenciadores de bases de dados (SGBDs) foram desenvolvidos desde o início com o objetivo de facilitar e agilizar a recuperação exata de dados em grandes arquivos de dados. A recuperação exata é um requisito das aplicações que os SGBDs originalmente deveriam suportar. Com o passar do tempo, outras aplicações foram requisitando suporte aos gerenciadores, culminando atualmente no esforço para armazenar e recuperar adequadamente dados muito mais complexos que os atributos numéricos ou textuais tratados no início, como imagens, audio, estruturas genéticas, etc.

*Este trabalho foi apoiado pela FAPESP (Fundação de Amparo à Pesquisa do Estado de São Paulo) processos 02/11771-3 e 02/07318-1, e pelo CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico) processos 52.1685/98-6, 52.1267/96-0 e 860.068/00-7.

Dentro desse rol de aplicações muito mais abrangente às quais os SGBDs atualmente devem atender, o requisito de recuperação exata, originalmente omni-presente em muitas aplicações, deixa de ser necessário. Em aplicações tais como sistemas de apoio à decisão, devido a sua natureza geralmente exploratória, a resposta exata "até o último bit" pode não ser necessária, pois o usuário pode preferir uma resposta aproximada obtida mais rapidamente.

Acelerar as operações de busca de dados nesses domínios é de especial importância por diversos fatores. O custo do acesso exato a dados multimídia tende a ser muito alto. Além disso, em decorrência da maneira como a tecnologia atual vem sendo desenvolvida para prover suporte à recuperação de dados multimídia, as operações de busca não são efetuadas realmente sobre os dados originais, mas sobre características extraídas desses dados, as quais os descrevem. A transformação intermediária de extração das características corresponde a um processo de redução de dimensionalidade, o qual necessariamente gera perda de detalhes e, portanto, leva à introdução de um fator de erro ao processo. Ao mesmo tempo, a definição de similaridade em conjuntos de dados complexos é em geral menos precisa do que em conjuntos de dados mais simples [Rubner and Tomasi, 2001], o que faz com que essas operações sejam candidatas naturais para serem tratadas por algoritmos de busca não exata.

Apesar das estruturas de indexação propostas para agilizar as buscas por similaridade, existem casos onde essas estruturas não são eficientes, por exemplo, devido aos problemas oriundos da "maldição da dimensionalidade" [Chávez and Navarro, 2001]. Em altas dimensões, as distâncias entre os objetos tendem a se homogenizar, fazendo com que praticamente toda a estrutura de indexação seja percorrida para a obtenção de um conjunto-resposta exato. Em espaços métricos, embora a inexistência de coordenadas não permita a análise de complexidade em termos de dimensões, existe comportamento semelhante em consultas por similaridade sobre dados complexos [Chávez and Navarro, 2001], sendo em alguns casos a busca seqüencial mais vantajosa.

Neste trabalho foram estudadas maneiras de agilizar a busca por similaridade de dados indexados em estruturas métricas. Utilizando algoritmos genéticos, foram desenvolvidos dois algoritmos de consultas aproximadas, cujo refinamento das respostas obtidas é dependente do tempo de processamento disponível. Com apenas uma fração do tempo necessário para a consulta exata, pode ser obtida uma boa aproximação do conjunto-resposta, como pede ser verificado pelos experimentos realizados.

Na Seção 2, são discutidos conceitos básicos envolvidos no desenvolvimento deste trabalho. Na Seção 3 são apresentados alguns trabalhos correlatos. Em seguida, na Seção 4, são apresentadas as duas técnicas desenvolvidas para consultas aproximadas utilizando algoritmos genéticos, e os resultados dos experimentos realizados são discutidos na Seção 5. Finalmente, na Seção 6, têm-se as conclusões do trabalho.

2. Conceitos Básicos

Dados complexos, como dados multimídia, geralmente não possuem uma relação de ordem total, sendo representados como dados em um domínio métrico. A classe de operadores mais adequada para manipular esses dados são os operadores por similaridade [Faloutsos, 1997]. Para indexar dados em domínios métricos e utilizar os operadores por similaridade, foram desenvolvidos os Métodos de Acesso Métricos (MAMs). Nesta seção serão apresentados esses conceitos, assim como uma breve introdução sobre algoritmos genéticos, utilizados para o desenvolvimento das técnicas de consultas por similaridade aproximadas.

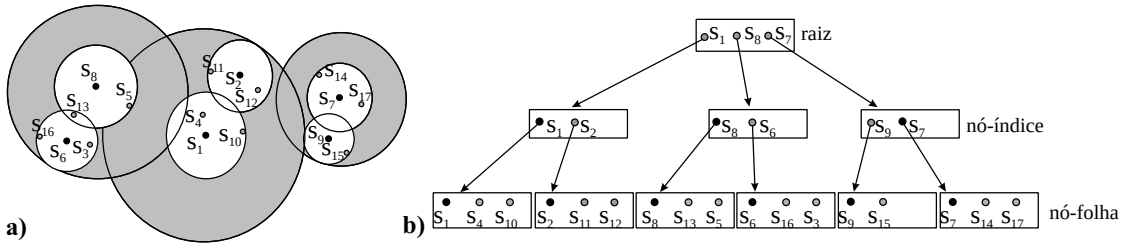

Figura 1: a)Representação de uma *Slim-tree* e b) sua estrutura lógica.

2.1. Domínios Métricos e Consultas por similaridade

Um domínio métrico é definido como $\mathcal{M} = (\mathcal{D}, d())$, onde $\mathcal{D}$ 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 $z \in \mathcal{D}$, uma função de distância (métrica) $d : \mathcal{D} \times \mathcal{D} \rightarrow \mathbf{R}^+$ deve satisfazer as seguintes propriedades:

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

Em domínios métricos, os principais operadores de consulta por similaridade são:

- RQ(s$_q$, r$_q$) - Consulta por abrangência (“*Similarity Range Query*”): retorna todos os objetos cujas distâncias até objeto central da consulta s_q sejam menores ou iguais ao raio da consulta r_q;

- k-NNQ(s$_q$, k) - Consulta aos k-vizinhos mais próximos(“*k-Nearest Neighbor Query*”) - : retorna os k objetos mais próximos do objeto central de consulta s_q.

2.2. Métodos de Acesso Métricos

Para conjuntos de dados em domínios métricos, onde não são consideradas relações geométricas e somente existem os objetos e as distâncias entre eles, diversas pesquisas levaram ao desenvolvimento das estruturas de indexação chamadas genericamente de Métodos de Acesso Métricos (MAMs). Os métodos propostos por Burkhard e Keller [Burkhard and Keller, 1973] foram o ponto de partida, apresentando as técnicas de particionamentos recursivos que permitem a construção de MAMs. Desde esse trabalho pioneiro, vários outros MAMs foram desenvolvidos, culminando nos MAMs dinâmicos, como a *M-tree* [Ciaccia et al., 1997] e a *Slim-tree* [Traina et al., 2000]. Em [Hjaltason and Samet, 2003] é apresentada uma extensa revisão bibliográfica dos métodos de acesso métricos.

A *Slim-tree* é uma estrutura balanceada e dinâmica, que tem crescimento das folhas para a raiz, e permite inserções de dados de um domínio métrico. Os objetos são armazenados nas folhas, organizados numa estrutura hierárquica que utiliza um objeto representativo como centro de uma região de cobertura dos objetos em uma subárvore, delimitada por um raio. A Figura 1 apresenta a disposição de 17 objetos em uma *Slim-tree* e sua estrutura lógica. A *Slim-tree*, assim como a maioria dos MAMs, utiliza a propriedade de desigualdade triangular para “podar” cálculos de distâncias em buscas por similaridade em domínios métricos.

Para a implementação das técnicas de consultas aproximadas desenvolvidas neste trabalho foi utilizada a *Slim-Tree*. Isso não limita a aplicabilidade dos conceitos apresentados, pois os algoritmos dos principais operadores de seleção por similaridade são basicamente os mesmos em todas as árvores métricas e espaciais [Roussopoulos et al., 1995], o que implica que as técnicas de consulta aproximada

desenvolvidas neste trabalho podem ser implementadas em outros MAMs de maneira equivalente.

2.3. Algoritmos Genéticos

Algoritmos genéticos têm sido cada vez mais explorados na solução de problemas envolvendo buscas e otimizações, principalmente pela robustez e simplicidade que oferecem. São modelos de processamento computacional que visam simular os mecanismos de seleção natural e evolução encontrados na natureza [Goldberg, 1989].

Cada resposta candidata presente no espaço de busca do problema é considerada como um indivíduo, e representada univocamente por uma cadeia de símbolos, chamada de cromossomo. Cada símbolo presente no cromossomo é chamado de gene. Partindo de uma população inicial de indivíduos, são aplicados ciclicamente os operadores genéticos, dando origem a novas gerações de indivíduos. A pressão seletiva, que na natureza é exercida pelo ambiente, é simulada pela aplicação de uma função objetivo, que avalia a aptidão de cada indivíduo da população. Espera-se que depois de várias gerações, a população inicial evolua, dando origem a uma população final de indivíduos mais aptos.

Os principais operadores genéticos são:
- **Seleção**: responsável pela escolha probabilística de quais indivíduos serão selecionados para reprodução;
- **Cruzamento**: gera novos indivíduos a partir da troca de informações genéticas dos indivíduos pais;
- **Mutação**: alteração aleatória do código genético de um indivíduo. É uma movimentação aleatória pelo espaço de busca, necessária para a diversidade genética da população.

O algoritmo genético é executado até que ocorra uma condição de parada, como por exemplo o esgotamento do tempo de execução.

3. Trabalhos correlatos

A área de pesquisa que trata da seleção de vizinhos mais próximos em domínios espaciais de altas dimensões aborda principalmente problemas oriundos da “maldição da dimensionalidade”. Para tratar esses problemas, várias soluções têm sido tentadas, como por exemplo, a criação de estruturas específicas, como a *Pyramid-tree* que particiona um espaço multidimensional em pirâmides bi-dimensionais [Berchtold et al., 1998], a criação de estruturas em *hashing* como em [Gionis et al., 1999], através do tratamento algébrico das funções de distância ou aceitando taxas de erro controladas nos cálculos de similaridade[Ciaccia and Patella, 2000], e com a determinação de vizinhança baseada em agrupamentos.

Para consultas aproximadas em espaços métricos, alguns dos métodos citados podem ser utilizados, como é o caso do PAC-NN (Probably Approximately Correct Nearest Neighbor), proposto em [Ciaccia and Patella, 2000]. Em [Zezula et al., 1998], três técnicas de aproximação foram propostas utilizando a *M-Tree*[Ciaccia et al., 1997], todas visando reduzir o número de comparações, e diferenciadas pela heurística utilizada para abreviar a execução da consulta exata. Em [Goldstein and Ramakrishnan, 2000], foi proposta a *P-Sphere tree*, (*Probabilistic sphere tree*), estrutura de indexação que realiza consultas aproximadas de 1-NNQ. Em [Bustos and Navarro, 2004], são apresentados algoritmos de consulta probabilísticos baseados em adaptações do algoritmo de k-NNQ incremental [Hjaltason and Samet, 2000], aplicáveis a MAMs baseados em particionamento compacto do espaço. Outro expediente amplamente utilizado e estudado é o mapeamento de objetos presentes em espaços complexos, em que o cálculo da

distância entre os objetos tende a ser mais custoso, para espaços onde tal cálculo seja menos complexo, como a técnica *FastMap* [Faloutsos and Lin, 1995].

A maioria das técnicas de aproximação citadas tem restrições quanto a sua aplicação. Como foi dito, muitas são aplicáveis somente em espaços multidimensionais ou em certos domínios específicos. Algumas, aplicáveis ao caso mais geral de domínios métricos, são restritas às 1-NNQs [Ciaccia and Patella, 2000] [Goldstein and Ramakrishnan, 2000]. Quanto às técnicas propostas em [Zezula et al., 1998] e [Bustos and Navarro, 2004], as heurísticas são utilizadas somente para abreviar a execução da consulta exata convencional.

A área de aplicação dos algoritmos desenvolvidos neste trabalho abrange qualquer conjunto de dados onde uma função de distância tenha sido definida. São aplicáveis às consultas por similaridade K-NNQ, RQ e várias de suas variações, como por exemplo k-farthest neighbor queries (k - FNq) e external range queries (ERq). Além de permitir abreviar a execução da consulta exata, limitando o tempo de processamento da consulta, os algoritmos desenvolvidos visam acelerar a formação do conjunto-resposta através da utilização de algoritmos genéticos para otimização do percurso na árvore.

Uma versão preliminar desse trabalho foi publicada em [Bueno et al., 2005]. Já neste presente artigo, é apresentado um novo algoritmo para consultas por similaridade aproximadas, e ambos os algoritmos desenvolvidos são apresentados detalhadamente. Além disso, os experimentos foram realizados de maneira mais ampla, sendo capazes de melhor avaliar o desempenho dos algoritmos desenvolvidos.

4. Algoritmos genéticos para consultas por similaridade aproximadas

Mesmo com a utilização dos MAMs, a execução de consultas (exatas) por similaridade em grandes bases de dados complexos tende a ser proibitiva. Para melhorar a eficiência de tais consultas, podem ser realizadas buscas inexatas, com resultados aproximados, utilizando uma quantidade de tempo reduzida. O algoritmo genético procura pelos melhores caminhos na árvore, com o intuito de melhorar e agilizar a formação do conjunto-resposta.

Os algoritmos desenvolvidos foram integrados aos MAMs *Slim-tree* como opções de consulta alternativas e podem ser utilizados por qualquer usuário do MAM, com a necessidade apenas da definição adicional do critério de parada, além daquelas requeridas para a utilização convencional da *Slim-tree*. Também podem ser parametrizados os operadores e pesos dos algoritmos genéticos, mas caso o usuário não queira, os valores "default" definidos permitem usar os algoritmos com um bom desempenho. Com isso, a área de aplicação dos algoritmos desenvolvidos abrange qualquer conjunto de dados onde uma função de distância tenha sido definida.

Para a implementação do algoritmo genético é necessária a representação cromossômica do espaço de busca, ou seja, cada possível solução do algoritmo genético deve ser representada univocamente por um cromossomo. Portanto, cada caminho na árvore que comece na raiz e pode chegar a um nó-folha, é considerado como um indivíduo do algoritmo genético, e é representado por um cromossomo com $h - 1$ posições, onde h é a altura da árvore. A codificação dos indivíduos em cromossomos é feita a partir do caminho percorrido na estrutura, desde a raiz até o nó-folha. O valor inteiro presente em cada posição do cromossomo indica a posição na página do respectivo nível na árvore que é percorrida pelo caminho, como pode-se observar na Figura 3 b). Com isso, dado um indivíduo que tenha um cromossomo factível, o caminho representado por esse cromossomo levará até um nó-folha, onde serão avaliados os objetos para possível inclusão no conjunto-resposta.

A população inicial é gerada aleatoriamente, cada indivíduo da população é avaliado e recebe um valor de aptidão, seu *fitness*. Tal avaliação é feita com uma incursão na árvore, percorrendo o caminho representado pelo cromossomo do indivíduo e calculando sua aptidão com a função objetivo.

4.1. Função Objetivo

A aptidão de um caminho é calculada baseando-se na "qualidade" dos objetos alcançados e pela quantidade de objetos qualificados para compor o conjunto-resposta.

Quando um cromossomo representa um caminho factível, ou seja, um caminho que leva até um nó-folha, o algoritmo genético analisa o nó-folha, qualificando esse nó utilizando sua distância até o objeto de referência para consulta, e contando quantos objetos são selecionados para compor o conjunto-resposta. O peso da seletividade no cálculo da função objetivo é definido dinamicamente, com pequena contribuição nas primeiras gerações do algoritmo genético e com aumento gradativo durante a evolução. A seguir temos a definição da função objetivo:

$$f() = (D_{Max} - d()) \times (1 + Ws \times Sel) \tag{1}$$

onde $f()$ é a função objetivo, D_{Max} é o maior valor da função de distância calculado na geração atual, $d()$ é a função de distância métrica calculada, Sel representa a seletividade do caminho e Ws é o peso dado para a seletividade. A definição de Ws depende de qual o critério de parada vai ser utilizado na consulta. Abaixo temos a definição de Ws quando utiliza-se o tempo de execução como critério de parada:

$$Ws = \frac{t}{t_{Max}} \tag{2}$$

sendo Ws o peso da seletividade, t o tempo decorrido no processo de evolução até a geração atual e t_{Max} o tempo máximo para a execução da consulta. Com essa definição, a seletividade não tem grande influência no início da execução do algoritmo, quando a seletividade de um nó não indica necessariamente a qualidade dos objetos adicionados.

4.2. Aplicação dos operadores genéticos

Partindo de uma população inicial de indivíduos, os operadores genéticos são aplicados ciclicamente, dando origem a novas gerações de indivíduos. Depois de avaliados todos os indivíduos, é aplicado o operador genético de seleção. A seleção consiste na escolha dos indivíduos da população atual para reprodução, de modo que os mais aptos tenham mais chances de se reproduzir. Foi utilizada a técnica da *roleta* [Goldberg, 1989]. Indivíduos com pouca aptidão tendem a se extinguir, enquanto os mais adaptados tendem a se reproduzir e manter suas características nas gerações seguintes. Cada indivíduo selecionado da população é copiado na nova população, para a aplicação dos operadores genéticos de cruzamento e mutação. Foi utilizada a estratégia elitista, de maneira que o melhor indivíduo da geração atual seja transferido automaticamente para a geração seguinte. Na implementação do elitismo, o indivíduo mais apto é copiado como um indivíduo adicional, não descartando assim nenhum indivíduo gerado pela aplicação dos operadores genéticos.

O operador genético de cruzamento possibilita a criação de novos indivíduos, os filhos, pela combinação dos cromossomos dos indivíduos pais. Foi utilizado o cruzamento de 1 ponto, pareando os indivíduos aleatoriamente, e para cada par de indivíduos definindo um ponto de cruzamento. Os cromossomos dos pais são divididos em duas partes, que

são trocadas entre eles, dando origem a dois novos cromossomos filhos. A operação de mutação altera o código genético de um indivíduo, e é aplicada de acordo com uma probabilidade definida, geralmente baixa. Uma mutação consiste na alteração aleatória do valor de um gene do cromossomo. Com a aplicação da mutação, não existem áreas inalcançáveis do espaço de busca.

Depois da aplicação dos operadores genéticos é formada uma nova população, e inicia-se um novo ciclo, como pode ser visto na Figura 2, que apresenta o esquema dos algoritmos genéticos para consultas por similaridade aproximadas desenvolvidos. O critério de parada dos algoritmos genéticos descritos neste artigo é o tempo de processamento, definido pelo usuário. Como versões alternativas, também podem ser utilizados como critérios de parada o número de cálculos de distância realizados, o número de acessos a disco, o número de gerações do algoritmo genético ou a redução da taxa de descoberta de indivíduos mais aptos. É importante ressaltar que, quando é passado como parâmetro tempo suficiente, o algoritmo realiza a consulta obtendo o mesmo conjunto-resposta do algoritmo de consulta exata.

4.3. Tratamento de indivíduos infactíveis

Após a aplicação dos operadores genéticos de cruzamento e mutação, ou inicialmente, depois da formação da população inicial, os indivíduos gerados podem ter cromossomos que levem a caminhos infactíveis na árvore. Quando nós inexistentes na árvore estão representados pelo cromossomo de um indivíduo, a técnica utilizada foi a substituição do gene que representa este nó inválido pelo gene mais próximo do original que represente um nó válido na árvore, como pode ser visto na Figura 3.

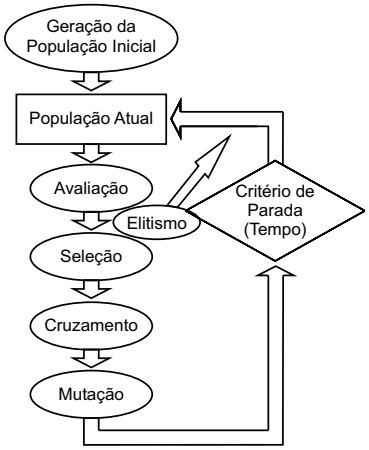

Figura 2: Esquema de execução dos algoritmos desenvolvidos.

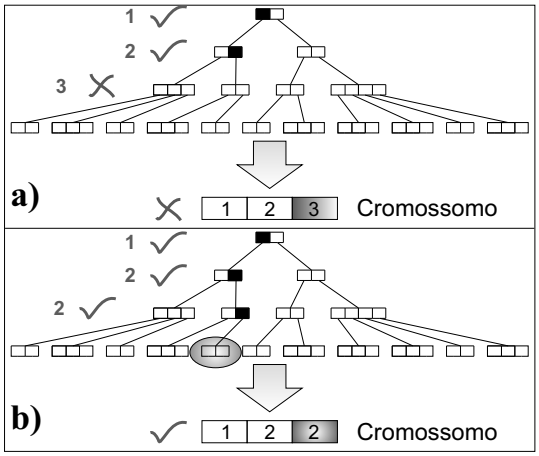

Figura 3: a)Cromossomo infactível; b)Cromossomo corrigido.

Quando o indivíduo tem um cromossomo que representa um caminho que leva a uma subárvore inválida (com possibilidades de busca esgotadas ou que pode ser podada utilizando-se a propriedade de desigualdade triangular), foram utilizadas duas técnicas, que deram origem aos dois algoritmos desenvolvidos: correção dos caminhos que podem ser podados e função objetivo com aplicação de penalidades.

4.3.1. Correção dos caminhos que podem ser podados

Com esta técnica, no final da fase de avaliação dos indivíduos, todos os cromossomos representam caminhos que levam a nós-folha da árvore, a não ser no caso de todos os caminhos factíveis já terem sido percorridos, tendo sido realizada, nesse caso, uma consulta com obtenção do conjunto-resposta exato.

Durante a fase de avaliação, quando um gene do cromossomo representa um nó inválido, esse gene é substituído por um outro que leve à uma subárvore factível. Esse nó válido é procurado entre os nós "irmãos" do nó inválido. No caso de não haver mais nenhum nó irmão factível, sobe-se um nível na árvore, repetindo o mesmo mecanismo no nível superior, e a subárvore onde foram esgotadas as possibilidades de busca é marcada como inválida, impedindo uma futura tentativa de explorá-la. No caso de encontrar uma subárvore válida, o cromossomo original é modificado, passando a representar um caminho factível.

4.3.2. Função objetivo com aplicação de penalidades

Essa técnica busca penalizar os indivíduos com cromossomos que representam caminhos infactíveis, que não atinjam um nó-folha válido. Apesar disso, é dada uma chance de aprendizado ao indivíduo com cromossomo infactível. Se um nó é inválido, são analisados os irmãos desse nó, e se algum deles for válido, este substitui o nó inválido e o cromossomo é alterado. Essa estratégia é indicada devido ao modo como os nós são lidos do disco, sendo que os irmãos do nó inválido já têm suas distâncias calculadas de qualquer forma, portanto suas avaliações não implicam em um custo computacional adicional significativo.

Quando não é encontrado nenhum nó válido, ao contrário da técnica anterior, a incursão na árvore é interrompida, não retornando ao nível superior para nova tentativa. Por não atingir um nó-folha, a seletividade do caminho é nula e não contribui para o cálculo da função objetivo, que baseia-se somente na distância entre o objeto de consulta e o nó-índice de maior profundidade na árvore alcançado pela incursão no caminho infactível. A penalidade a ser aplicada é relativa ao nível que o caminho conseguiu alcançar em sua incursão na árvore, sendo que quanto mais próximo do nó-folha esteja o nó-índice alcançado pela incursão, menor a penalidade. A aplicação das penalidades é feita da seguinte maneira:

$$f'() = f() \times \frac{1}{(h - l) \times Wp} \tag{3}$$

onde $f'()$ é a nova função objetivo com a aplicação da penalidade, $f()$ é a função objetivo calculada baseando-se na distância entre o nó alcançado e o objeto de consulta (Equação 1), sendo h a altura da árvore, l o nível atingido pela incursão do caminho infactível na árvore e Wp o peso dado à penalidade, definido com o valor 1, podendo variar, gerando penalidades mais ou menos severas.

4.4. Algoritmos Desenvolvidos

O Algoritmo 4.1 apresenta o funcionamento básico de ambos os algoritmos desenvolvidos para consultas por similaridade aproximadas, que diferem quanto ao tratamento dos indivíduos inválidos realizado no procedimento $Avalia(Pop)$, e apresentado nos Algoritmos 4.2 e 4.3, detalhados a seguir.

A população inicial é gerada aleatoriamente (linha 1), e os indivíduos gerados são avaliados. A avaliação é feita com a incursão na árvore seguindo o caminho representado pelo cromossomo de cada indivíduo. Após a avaliação dos indivíduos da população, são selecionados pares de indivíduos para reprodução (linha 6 e 7), que serão submetidos aos operadores genéticos de cruzamento e mutação. $OpSelecao$, $OpCruzamento$ e $OpMutacao$ são aplicados da forma como foram apresentados na seção anterior. Por fim, os indivíduos gerados, armazenados na população auxiliar $AuxPop$, são copiados

para população corrente Pop, para o recomeço do ciclo do algoritmo genético. Esses procedimentos são repetidos ciclicamente gerando várias gerações de indivíduos, até que a condição de parada seja verdadeira, ou seja, até que o tempo de execução disponível termine, ou então que a árvore tenha suas possibilidades de busca esgotadas.

Algoritmo 4.1 *k-NN para consultas aproximadas*

Entrada: Objeto de consulta s_q e número de vizinhos mais próximos k.
Saída: Conjunto-resposta contendo os objetos que satisfazem os critérios da consulta.

```
 1: GeraPopulacaoInicial(Pop) {População gerada aleatoriamente}
 2: repita
 3:    Avalia(Pop)
 4:    se possibilidades de busca não estejam esgotadas então
 5:       repita
 6:          pai1 ← OpSelecao(Pop)
 7:          pai2 ← OpSelecao(Pop) {2 indivíduos selecionados para se reproduzirem}
 8:          OpCruzamento(pai1, pai2, AuxPop) {descendentes armazenados em AuxPop}
 9:          OpMutacao(AuxPop)
10:       até que AuxPop esteja completa
11:       Pop ← AuxPop
12: até que Condição de parada seja verdadeira ou possibilidades de busca estejam esgotadas
```

No Algoritmo 4.2 é apresentado o procedimento $Avalia(Pop)$ utilizando Algoritmo de correção. D_{Max} armazena o maior valor da função de distância calculado na geração atual, no é um ponteiro para subárvores e aponta inicialmente para a raiz da árvore. Para cada um dos indivíduos da população, é realizada uma incursão na árvore de acordo com o caminho representado no cromossomo, com o objetivo de atingir um nó-folha. Os indivíduos ind_t têm os atributos de distância, seletividade e aptidão (*fitness*).

Se um gene do cromossomo indica uma subárvore que não seja válida, procura-se por uma subárvore válida no mesmo nó (linha 10). Se é encontrada, o gene (antes inválido) do cromossomo original é modificado (linha 12), passando a representar o caminho que leva a subárvore válida selecionada, e a incursão continua no nível seguinte. Se não é encontrada nenhuma subárvore válida no mesmo nó, sobe-se um nível na árvore (linha 21) e repete-se a busca por uma subárvore válida no nível superior. A subárvore com possibilidades de busca esgotadas é marcada como inválida (linha 20). Se a raiz da árvore tem suas possibilidades de busca esgotadas, o algoritmo de consulta é encerrado, e a resposta obtida pelo algoritmo é exata (linha 17). Ao chegar na linha 22, no aponta para um nó-folha, que será examinado para a composição do conjunto-resposta e cálculo da adaptação (*fitness*) do indivíduo. Na linha 30 é realizado o cálculo do *fitness* de cada indivíduo, segundo a Equação 1. Portanto, ao final do procedimento $Avalia(Pop)$ do Algoritmo 4.2, a não ser que a consulta tenha sido encerrada por ter pesquisado todos os nós válidos (linha 17), todos os indivíduos têm suas aptidões calculadas e estão prontos para serem submetidos ao processo de seleção.

No Algoritmo 4.3 é apresentado o procedimento $Avalia(Pop)$ com aplicação de penalidades no cálculo da função objetivo do algoritmo genético. O algoritmo tem mecanismo semelhante ao anterior, diferindo no seguinte ponto: quando um gene indica uma subárvore que não seja válida, e não é encontrada nenhuma subárvore válida no mesmo nó, o algoritmo interrompe sua incursão na árvore, marcando a subárvore como inválida e calculando a distância até o nó-indice alcançado na incursão (entre linhas 21 e 28). É também armazenado o nível alcançado pela incursão, pois esse valor será utilizado na aplicação da penalidade no cálculo da função objetivo dos indivíduos (linha 40), segundo a Equação 3.

Algoritmo 4.2 *$Avalia(Pop)$ utilizando Algoritmo de correção*

Entrada: População Pop de indivíduos ind_t.
Saída: Conjunto-resposta com possível inserção de novos objetos.

```
 1: D_max ← 0
 2: no ← nó raiz da árvore
 3: para cada Indivíduo ind_t ∈ Pop faça
 4:    ind_t.seletividade ← 0
 5:    para cada Gene gen_g do ind_t faça {iniciando com g ← 0 e incrementando a cada passagem}
 6:       se a subárvore de no indicada pelo gene g é válida então
 7:          no ← subárvore de no indicada pelo gene g
 8:          desce 1 nível na árvore
 9:       senão
10:          Procurar por subárvore válida de no
11:          se achar subárvore válida de no então
12:             modifica o valor do gene inválido para representar subárvore encontrada
13:             no ← subárvore válida de no encontrada
14:             desce 1 nível na árvore
15:          senão
16:             se no é a raiz da árvore então
17:                FIM {possibilidades de busca esgotadas}
18:             senão
19:                Decrementa(g) {voltando ao gene na posição anterior}
20:                marca no como inválido
21:                sobe 1 nível na árvore
       {no aponta um nó-folha}
22:    para cada Objeto s_i ∈ no faça
23:       ind_t.dist ← d(s_rep, s_q))
24:       se ind_t.dist > D_max então
25:          D_max ← ind_t.dist
26:       se s_i é qualificado então {para compor o conjunto-resposta}
27:          Adicionar s_i ao conjunto-resposta
28:          Incrementa(ind_t.seletividade)
29: para cada Indivíduo ind_t ∈ Pop faça
30:    ind_t.fitness ← (D_max − ind_t.dist) × (1 + Ws × ind_t.seletividade)
```

5. Experimentos realizados

Os algoritmos desenvolvidos para consultas por similaridade aproximadas foram testados para validação e avaliação de desempenho com uma grande variedade de conjuntos de dados, tanto reais quanto sintéticos, que diferem quanto ao número de dimensões, o tamanho dos objetos e do conjunto de dados, distribuição dos dados no espaço e função de distância utilizada. Os conjuntos de dados utilizados são descritos na Tabela 1.

Tabela 1: Descrição dos conjuntos de dados utilizados nos experimentos.

Nome	Nro. Objs.	Dimensão	Métrica	Descrição
Random128	10000	128	L_2	Dados sintéticos aleatoriamente distribuídos
Clusters128	20000	128	L_2	Dados sintéticos com 8 nuvens de agrupamentos
Palavras em Inglês	25143	Adimensional	L_{Edit}	Conjunto de palavras extraídas do dicionário da língua inglesa
MRTHistogramas	39740	2000	L_2	Histogramas de imagens de várias partes do corpo obtidas por tomografia.

Os conjuntos “Clusters128” e “Random128” foram gerados pela ferramenta para geração de dados *DBGen* [Bueno et al., 2004]. A métrica L_{Edit}, utilizada com o conjunto “Palavras em Inglês”, retorna o número de substituições, inserções e remoções necessárias para se obter uma palavra a partir de outra. Os resultados foram gerados pelos algoritmos

Algoritmo 4.3 *$Avalia(Pop)$ com aplicação de penalidades no cálculo da função objetivo*

Entrada: População Pop de indivíduos ind_t.
Saída: Conjunto-resposta com possível inserção de novos objetos.

1: $D_{max} \leftarrow 0$
2: $Nivel_{max} \leftarrow$ altura da árvore -1
3: $Wp \leftarrow$ peso dado à penalidade
4: $no \leftarrow$ nó raiz da árvore
5: **para** cada Indivíduo $ind_t \in Pop$ **faça**
6: $ind_t.seletividade \leftarrow 0$
7: **para** cada Gene gen_g do ind_t **faça** {iniciando com $g \leftarrow 0$ e incrementando a cada passagem}
8: **se** a subárvore de no indicada pelo gene g é válida **então**
9: $no \leftarrow$ subárvore de no indicada pelo gene g
10: desce 1 nível na árvore
11: **senão**
12: Procurar por subárvore válida de no
13: **se** achar subárvore válida de no **então**
14: modifica o valor do gene inválido para representar subárvore encontrada
15: $no \leftarrow$ subárvore válida de no encontrada
16: desce 1 nível na árvore
17: **senão**
18: **se** no é a raiz da árvore **então**
19: FIM {possibilidades de busca esgotadas}
20: **senão**
21: Decrementa(g) {voltando ao gene na posição anterior}
22: marca no como inválido
23: sobe 1 nível na árvore
24: $ind_t.dist \leftarrow d(s_{rep}, s_q))$ {s_{rep} é o representativo do nó alcançado.}
25: **se** $ind_t.dist > D_{max}$ **então**
26: $D_{max} \leftarrow ind_t.dist$
27: $ind_t.nivel \leftarrow g$ {nível alcançado na incursão}
28: $BREAK$ {Interrompe incursão}
29: **se** no aponta para um nó folha **então**
30: **para** cada Objeto $s_i \in no$ **faça**
31: $ind_t.dist \leftarrow d(s_{rep}, s_q))$
32: **se** $ind_t.dist > D_{max}$ **então**
33: $D_{max} \leftarrow ind_t.dist$
34: **se** s_i é qualificado **então** {para compor o conjunto-resposta}
35: Adicionar s_i ao conjunto-resposta
36: Incrementa($ind_t.seletividade$)
37: **para** cada Indivíduo $ind_t \in Pop$ **faça**
38: $ind_t.fitness \leftarrow (D_{max} - ind_t.dist) \times (1 + Ws \times ind_t.seletividade)$
39: **se** Não alcançou um nó folha **então**
40: $ind_t.fitness \leftarrow \frac{ind_t.fitness}{(Nivel_{max} - ind_t.nivel) * Wp}$ {Aplica penalidade}

implementados em C++ (*Borland C++ Builder*), executados em um computador *AMD Athlon XP* 2.6GHz, com 512Mb de *RAM* e 40Gb de disco rígido, com o sistema operacional *Windows 2000 Professional*.

Em todos os resultados apresentados, os experimentos foram realizados com a população do algoritmo genético com tamanho 10, peso da penalidade (Wp) definido com valor 1, a taxa de probabilidade de cruzamento igual a 75% e a taxa de mutação de 0,5%, valores que mostraram-se adequados para obtenção dos melhores resultados. Para realização dos testes, os objetos de cada conjunto de dados foram aleatoriamente "embaralhados", simulando inserções aleatórias, comuns em atividades reais em bancos de dados. De cada um dos conjuntos de dados da Tabela 1 foram selecionados aleatoriamente 500 objetos para compor o conjunto de objetos centrais para consultas. Dos objetos selecionados, 250 foram removidos do conjunto de dados originais. Portanto, metade dos objetos do conjunto de busca pertencem ao conjunto indexado e metade não pertence.

Foram realizadas duas classes de experimentos. Na primeira classe o objetivo é avaliar a precisão das respostas obtidas obtidas com um tempo limitado de processamento. Na segunda classe, são avaliados os custos da obtenção das respostas exatas com os algoritmos desenvolvidos, em comparação com os algoritmos convencionais.

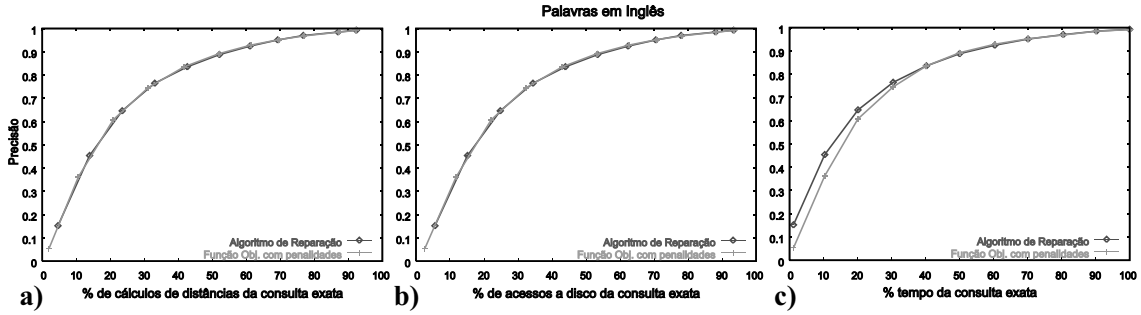

Figura 4: a) *Precisão* × *Porcentagem de cálculos de distâncias da consulta exata*, b) *Precisão* × *Porcentagem de acessos a disco da consulta exata* e c) *Precisão* × *Porcentagem do tempo de execução da consulta exata*, conjunto Palavras em Inglês.

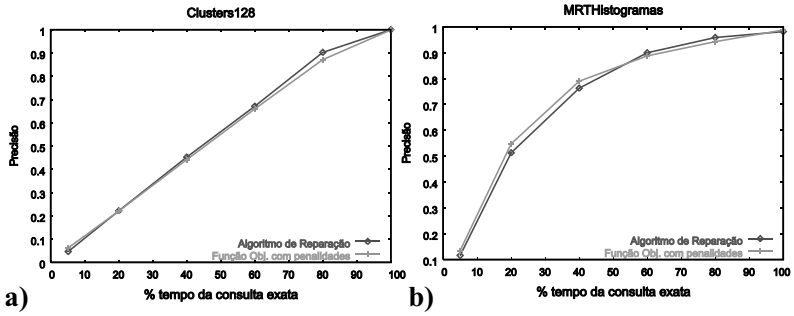

Figura 5: *Precisão* × *Porcentagem do tempo de execução da consulta exata*, conjuntos Clusters128 e MRTHistogramas.

5.1. Precisão das respostas aproximadas

As respostas obtidas pela execução do algoritmo convencional de consulta exata são consideradas como corretas. A precisão das respostas dos algoritmos de consultas aproximadas é dada pela proporção dos objetos recuperados que são considerados corretos. Os gráficos a seguir apresentam as precisões obtidas limitando a execução dos algoritmos à frações do tempo exigido para execução do algoritmo de consulta exata. Em todos os experimentos são realizadas consultas aos 50 vizinhos mais próximos com o cálculo da média de 500 consultas com diferentes objetos centrais de consulta, exceto no experimento (b) da Figura 5 ,onde foram realizadas 50 consultas. O algoritmo de consulta exata é executado com os diferentes objetos de referência para consulta, para a obtenção do tempo médio de execução. Depois, com os mesmos objetos de consulta, os algoritmos para consultas aproximadas são executados, limitados por diferentes frações do tempo médio da consulta exata.

O experimento que deu origem aos gráficos da Figura 4 foram realizados com o conjunto de dados de "Palavras em Inglês". Analizando o gráfico *Precisão* × *Porcentagem do tempo de execução da consulta exata*, pode-se ver que com 20% do tempo requerido para uma consulta exata obtém-se aproximadamente uma precisão de 65% na formação do conjunto-resposta, e com 50% do tempo da consulta exata alcança-se aproximadamente 90% de precisão para ambos os algoritmos. Como pode ser visto, os gráficos relativos a cálculos de distância e acessos a disco apresentam comportamento muito parecido, de maneira que a análise do gráfico de *Precisão* × *Porcentagem do tempo de execução da consulta exata* se mostra eficiente na avaliação dos algoritmos.

Na Figura 5, são apresentados os resultados de experimentos com o conjunto "Clusters128" no gráfico (a) e "MRTHistogramas" no gráfico (b). Apesar dos resultados alcançados no experimento (a) não terem sido tão bons como os da Figura 4, pode-

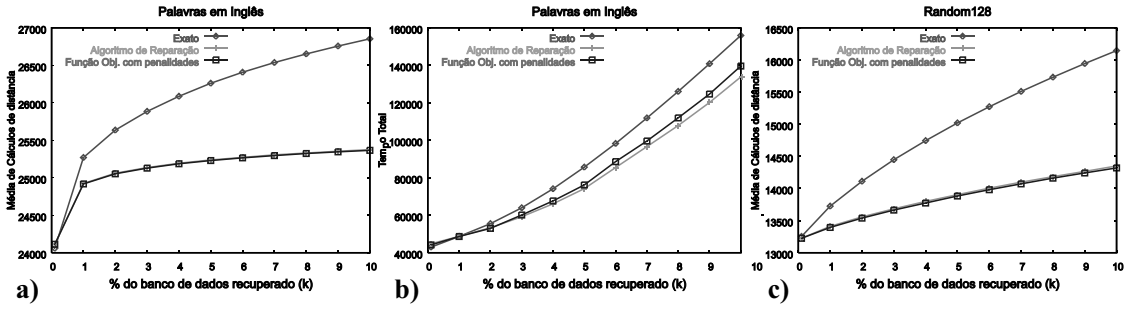

Figura 6: a) *Média de cálculos de distância*, conjunto Palavras em Inglês, b) *Tempo de execução*, conjunto Palavras em Inglês, e *(c) Média de cálculos de distância*, conjunto Random128. A abscissa dos gráficos é a porcentagem de dados recuperados pela consulta k-NN.

se observar que a média de tempo necessário para a obtenção da resposta exata pelos algoritmos de consultas aproximadas não ultrapassa a média de tempo da consulta exata. Analizando o gráfico do experimento (b), verifica-se que com 20% do tempo médio de uma consulta exata obtém-se uma resposta aproximada com precisão de 55% utilizando-se a estratégia de função objetivo com aplicação de penalidades (e aproximadamente 52% utilizando o algoritmo de correção), e com 60% do tempo da consulta exata alcança-se aproximadamente 90% de precisão.

5.2. Consultas exatas com os algoritmos desenvolvidos

Por fim, existe a possibilidade de obter-se a resposta exata com os algoritmos de consulta aproximada, não limitando-se o tempo de execução dos mesmos. Na Figura 6 são mostrados gráficos comparativos de desempenho entre o algoritmo de consulta exata convencional e os algoritmos desenvolvidos para consultas aproximadas executados sem limitação de tempo. Nestes experimentos, variou-se o número de vizinhos (k) requeridos pela consulta. Para a construção dos gráficos foi considerada a porcentagem dos objetos do conjunto de dados recuperada pelas consultas.

Na Figura 6, os experimentos (a) e (b) foram realizados com o conjunto “Palavras em Inglês” e o experimento (c) com o conjunto “Random128”, e as medidas também consideram a média dos cálculos de distância de 500 consultas, realizadas com objetos centrais de consulta diferentes. São apresentados gráficos que avaliam o custo das consultas considerando o tempo no experimento (b) e o número de cálculos de distâncias nos experimentos (a) e (c). Como pode ser visto nessa figura, apesar de não ser o principal objetivo dos algoritmos desenvolvidos, eles foram sempre capazes de obter a resposta exata com custos inferiores aos algoritmos convencionais de consulta exata, seja considerando-se o tempo de execução ou o número de cálculos de distância. No experimento (b) o algoritmo convencional de consulta exata precisou de aproximadamente 15% a mais de tempo do que o algoritmo de consultas aproximadas para obter a mesma resposta, com a consulta recuperando 5% do conjunto de dados. Já no experimento (c), o algoritmo convencional realizou 8% a mais de cálculos de distância do que o algoritmo de consultas aproximadas para consultas recuperando 5% do conjunto de dados, e 12,5% a mais com consultas recuperando 10% do conjunto de dados.

6. Conclusões

Em domínios de dados complexos, quase sempre são extraídas características dos elementos de dados para realizar comparações com outros elementos, ao invés de utilizar os próprios elementos de dados. Portanto, consultas exatas realizadas sobre dados

nesse tipo de domínio obtêm respostas exatas segundo as características utilizadas nas comparações, mas frequentemente isso não implica que as respostas obtidas sejam as melhores, ou então exatas, segundo a percepção do usuário. Dessa forma, os resultados obtidos por algoritmos para consultas aproximadas podem ser tão bons quanto aqueles obtidos pelos algoritmos de consultas exatas, muito mais custosos.

Este trabalho apresenta como contribuição inédita o desenvolvimento de algoritmos para consultas por similaridade aproximadas em domínios métricos, empregando algoritmos genéticos para acelerar a obtenção de respostas aproximadas. Tanto quanto é do conhecimento dos autores, não existe na literatura até agora outra descrição de algum algoritmo génetico para essa finalidade. Os algoritmos desenvolvidos têm sua execução limitada a um tempo pré-determinado, fornecido pelo usuário da consulta. Se o tempo for suficiente, a resposta exata é obtida. Durante a execução dos algoritmos, é explorada a navegação das subárvores buscando melhores caminhos de incursão na árvore, visando refinar o conjunto-resposta. Uma boa aproximação do conjunto-resposta exato pode ser obtida com um subconjunto de subárvores percorridas bem menor do que o conjunto de todas as subárvores válidas. Os resultados dos experimentos realizados mostram que os algoritmos desenvolvidos são capazes de obter respostas aproximadas com boa precisão utilizando apenas uma fração do tempo exigido pela consulta exata. Respostas com precisão superior a 90% foram obtidas limitando o tempo de execução dos algoritmos a apenas metade do tempo da consulta exata com o algoritmo convencional, e respostas com até 65% de precisão foram obtidas com 20% do tempo da consulta exata.

Além disso, as estratégias de busca baseadas em algoritmos genéticos também podem ser utilizadas para a obtenção das mesmas respostas exatas que os algoritmos tradicionais. Experimentos realizados mostraram que a resposta exata pode ser obtida com um custo menor do que com os algoritmos tradicionais para consulta exata.

Por fim, vale ressaltar que os experimentos apresentados neste trabalho comparam as respostas obtidas pelos algoritmos desenvolvidos com aquelas dos algoritmos tradicionais de consultas exatas. Mas como já foi discutido antes neste trabalho, respostas que não são consideradas exatas considerando comparações sobre as características extraídas podem ser consideradas corretas por um usuário humano. Assim, considerando a percepção humana, os algoritmos desenvolvidos podem recuperar respostas tão boas quanto aquelas obtidas pelo algoritmo para consultas exatas em uma pequena fração do tempo requerido.

Referências

[Berchtold et al., 1998] Berchtold, S., Böhm, C., and Kriegel, H.-P. (1998). The Pyramid-tree: Breaking the curse of dimensionality. In *ACM Int'l Conference on Data Management (SIGMOD)*, pages 142–153, Seattle, WA.

[Bueno et al., 2004] Bueno, R., Traina, Ferreira, M. R. P., and Traina, Caetano, J. (2004). Dbgen, manual da ferramenta. Relatório técnico 246, ICMC/USP.

[Bueno et al., 2005] Bueno, R., Traina Jr., C., and Traina, A. J. M. (2005). Accelerating approximate similarity queries using genetic algorithms. In *Proceedings of the 20th Annual ACM Symposium on Applied Computing (SAC)*, pages 621–626, Santa Fe, New Mexico, USA. ACM Press.

[Burkhard and Keller, 1973] Burkhard, W. A. and Keller, R. M. (1973). Some approaches to best-match file searching. *Communications of the ACM*, 16(4):230–236.

[Bustos and Navarro, 2004] Bustos, B. and Navarro, G. (2004). Probabilistic proximity search algorithms based on compact partitions. *Journal of Discrete Algorithms*, 2(1):115–134.

[Chávez and Navarro, 2001] Chávez, E. and Navarro, G. (2001). Towards measuring the searching complexity of metric spaces. In *Proc. Mexican Computing Meeting*, volume II, pages 969–978, Aguascalientes, México. Sociedad Mexicana de Ciencias de la Computación.

[Ciaccia and Patella, 2000] Ciaccia, P. and Patella, M. (2000). PAC nearest neighbor queries: Approximate and controlled search in high-dimensional and metric spaces. In *ICDE*, pages 244–255, San Diego, CA. IEEE Computer Society.

[Ciaccia et al., 1997] Ciaccia, P., Patella, M., and Zezula, P. (1997). M-tree: An efficient access method for similarity search in metric spaces. In *VLDB*, pages 426–435, Athens, Greece.

[Faloutsos, 1997] Faloutsos, C. (1997). Indexing of multimedia data. In *Multimedia Databases in Perspective*, pages 219–245. Springer Verlag.

[Faloutsos and Lin, 1995] Faloutsos, C. and Lin, K.-I. D. (1995). FastMap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. In *ACM SIGMOD International Conference on Management of Data*, pages 163–174, San Jose, CA. ACM Press.

[Gionis et al., 1999] Gionis, A., Indyk, P., and Motwani, R. (1999). Similarity search in high dimensions via hashing. In *VLDB*, volume 1, pages 518–529, Edinburgh, Scotland, UK. Morgan Kaufmann.

[Goldberg, 1989] Goldberg, D. E. (1989). *Genetic algorithms in search, optimization, and machine learning.* Addison-Wesley, 1 edition.

[Goldstein and Ramakrishnan, 2000] Goldstein, J. and Ramakrishnan, R. (2000). Contrast plots and P-sphere trees: space vs. time in nearest neighbour searches. In *VLDB*, pages 429–440.

[Hjaltason and Samet, 2000] Hjaltason, G. R. and Samet, H. (2000). Incremental similarity search in multimedia databases. Computer Science Department TR-4199, University of Maryland, College Park, MD.

[Hjaltason and Samet, 2003] Hjaltason, G. R. and Samet, H. (2003). Index-driven similarity search in metric spaces. *ACM Trans. Database Syst.*, 28(4):517–580.

[Roussopoulos et al., 1995] Roussopoulos, N., Kelley, S., and Vincent, F. (1995). Nearest neighbor queries. In *ACM SIGMOD International Conference on Management of Data*, pages 80–91, San Jose, CA. ACM Press.

[Rubner and Tomasi, 2001] Rubner, Y. and Tomasi, C. (2001). *Perceptual Metrics for Image Database Navigation*. Kluwer Academic Publishers.

[Traina et al., 2000] Traina, Caetano, J., Traina, A. J. M., Seeger, B., and Faloutsos, C. (2000). Slim-trees: High performance metric trees minimizing overlap between nodes. In *Intl. Conf. on Extending Database Technology*, volume 1777 of *Lecture Notes in Computer Science*, pages 51–65, Konstanz, Germany. Springer.

[Zezula et al., 1998] Zezula, P., Savino, P., Amato, G., and Rabitti, F. (1998). Approximate similarity retrieval with M-trees. *VLDB J.*, 7(4):275–293.