
Tratamento do tempo e dinamicidade em
dados representados em espaços métricos

Renato Bueno

SERVIÇO DE PÓS-GRADUAÇÃO DO ICMC-USP

Data de Depósito:

Assinatura:

Tratamento do tempo e dinamicidade em dados representados em espaços métricos¹

Renato Bueno

Orientador: *Prof. Dr. Caetano Traina Júnior*

Tese apresentada ao Instituto de Ciências Matemáticas e de Computação - ICMC-USP, como parte dos requisitos para obtenção do título de Doutor em Ciências - Ciências de Computação e Matemática Computacional.

**USP – São Carlos
Novembro de 2009**

¹ Apoio financeiro FAPESP (processo 05/02760-6)

Agradecimentos

A Deus, por tudo.

À minha família e minha noiva, por sempre estarem ao meu lado.

Ao meu orientador Prof. Dr. Caetano Traina Júnior e à Profa. Dra. Agma J. Machado Traina, pelo apoio, confiança e amizade.

Ao amigo Daniel dos Santos Kaster, representando todos os companheiros do Grupo de Bases de Dados e Imagens.

A todos os professores e funcionários do ICMC, sempre dispostos a ajudar.

Aos meus amigos e a todos aqueles que de alguma forma contribuíram para a realização deste trabalho.

À FAPESP, CAPES E CNPq, pelo apoio financeiro.

Resumo

Os Sistemas de Gerenciamento de Bases de Dados devem atualmente ser capazes de gerenciar dados complexos, como dados multimídia, sequências genéticas, séries temporais, além dos dados tradicionais. Em consultas em grandes coleções de dados complexos, a similaridade entre os dados é o fator mais importante, e pode ser adequadamente expressada quando esses dados são representados em espaços métricos. Independentemente do domínio de um tipo de dados, existem aplicações que devem acompanhar a evolução temporal dos elementos de dados. Porém, os Métodos de Acesso Métrico existentes consideram que os dados são imutáveis com o decorrer do tempo.

Visando o tratamento do tempo e dinamicidade em dados representados em espaços métricos, o trabalho apresentado nesta tese foi desenvolvido em duas frentes principais de atividades. A primeira frente tratou da inclusão das operações de remoção e atualização em métodos de acesso métrico, e visa atender às necessidades de domínios de aplicação em que dados em espaços métricos sofram atualização frequente, independentemente de necessitarem de tratamento temporal. Desta frente de atividades também resultou um novo método de otimização de árvores métricas, baseado no algoritmo de remoção desenvolvido.

A segunda frente de atividades aborda a inclusão do conceito de evolução temporal em dados representados em espaços métricos. Para isso foi proposto o Espaço Métrico-temporal, um modelo de representação de dados que permite a comparação de elementos métricos associado a informações temporais. O modelo conta com um método para identificar as contribuições relativas das componentes métrica e temporal no cálculo da similaridade. Também foram apresentadas estratégias para análise de trajetórias de dados métricos com o decorrer do tempo, através da imersão de espaços métrico-temporais em espaços dimensionais. Por fim, foi apresentado um novo método de balanceamento de múltiplos descritores para representação de imagens, fruto de modificações no método proposto para identificar as contribuições das componentes que podem formar um espaço métrico-temporal.

Abstract

Nowadays, the Database Management Systems (DBMS) must be able to manage complex data, such as multimedia data, genetic sequences, temporal series, besides the traditional data. For queries on large collections of complex data, the similarity among elements is the most relevant concept, and it can be adequately expressed when data are represented in metric spaces. Regardless of the data domain, there are applications that must track the evolution of data over time. However, the existing Metric Access Methods assume that the data elements are immutable.

Aiming at both treating time and allowing changes in metric data, the work presented in this thesis consisted of two main parts. The first part addresses the inclusion of the operations for element remotion and updating in metric access methods. These operations are meant to application domains that work with metric data that changes over time, regardless of the need to manage temporal information. A new method for metric trees optimization was also developed in this part of the work. It was based on the proposed remotion algorithm.

The second part of the thesis addresses including the temporal evolution concept in data represented in metric spaces. The Metric-Temporal Space was proposed, a representation model to allow comparing elements consisting of metric data with temporal information associated. The model includes a method to identify the relative contributions of the temporal and the metric components in the final similarity calculation. Strategies for trajectory analysis of metric data over time was also presented, through the immersion of metric-temporal spaced in dimensional spaces. Finally, a new method for weighting multiple image descriptors was presented. It was derived from changes in the proposed method to identify the contributions of the components of the metric-temporal space.

Sumário

Resumo	ii
Abstract	ii
Sumário	vi
Lista de Figuras	ix
Lista de Tabelas	xiv
1 Introdução	1
1.1 Considerações Iniciais	1
1.2 Motivação	3
1.3 Objetivos e Contribuições	4
1.4 Organização do Trabalho	5
2 Recuperação por conteúdo e consultas por similaridade	7
2.1 Considerações Iniciais	7
2.2 Recuperação por conteúdo em dados multimídia	8
2.3 Combinação de múltiplos descritores	11
2.4 Espaços métricos	13
2.4.1 Métricas	14
2.5 Consultas por similaridade	16
2.5.1 Consulta por abrangência	16
2.5.2 Consulta aos vizinhos mais próximos	16
2.5.3 Algoritmos para consultas por similaridade	17
2.6 Teoria dos fractais aplicada a bases de dados	20
2.7 Considerações Finais	22
3 Métodos de acesso métrico	23
3.1 Considerações Iniciais	23
3.2 Métodos de acesso métrico	23
3.3 <i>Slim-tree</i>	29

3.3.1	Organização da <i>Slim-tree</i>	29
3.3.2	Construção de uma <i>Slim-tree</i>	31
3.3.3	Consultas por similaridade na <i>Slim-tree</i>	33
3.3.4	Sobreposição em árvores métricas	35
3.4	Considerações Finais	37
4	Tempo em Bases de Dados	39
4.1	Considerações Iniciais	39
4.2	Dados temporais	40
4.3	Objetos móveis	43
4.4	Considerações Finais	47
5	Dinamicidade em Métodos de Acesso Métrico	49
5.1	Considerações Iniciais	49
5.2	Remoção em MAM	50
5.2.1	Experimentos	54
5.3	Uma técnica para redução de sobreposição e otimização em MAM dinâmicos	59
5.3.1	Push Pull	60
5.3.2	Smart Push Pull	62
5.3.3	Experimentos	63
5.4	Atualização em MAM dinâmicos	68
5.4.1	Experimentos	70
5.5	Considerações Finais	72
6	Evolução temporal em Dados Métricos	73
6.1	Considerações Iniciais	73
6.2	Espaço Métrico-Temporal	75
6.2.1	Similaridade Métrico-Temporal	76
6.2.2	Fator de escala para a similaridade métrico-temporal	79
6.2.3	Experimentos	82
6.3	Trajetórias de dados métrico-temporais	90
6.3.1	Mapeamento do espaço métrico-temporal	90
6.3.2	Consultas aproximadas no espaço mapeado	91
6.3.3	Experimentos	92
6.3.4	Visualização de dados métrico-temporais	97
6.4	Fator de escala para similaridade aplicado ao balanceamento de múltiplos descritores de imagens médicas	101
6.4.1	Experimentos	103
6.5	Considerações Finais	110

7 Conclusão	111
7.1 Principais contribuições	112
7.2 Trabalhos Futuros	113
Referências Bibliográficas	115

Lista de Figuras

2.1	Exemplo de extração de características: histograma normalizado com 256 níveis de cinza.	9
2.2	Áreas de cobertura de algumas funções de distância da família das normas L_p	14
2.3	Consulta por abrangência.	17
2.4	Consulta pelos vizinhos mais próximos	17
2.5	Coordenadas geográficas de intersecções de vias em <i>Montgomery County, MD, EUA</i> .(a) <i>Distance plot</i> .(b) Distribuição espacial dos dados.	21
3.1	Representação gráfica da estrutura lógica de nós-índice e nós-folha da <i>Slim-tree</i> . [Traina Jr. et al., 2002a]	30
3.2	Exemplo de indexação de sete palavras usando a função de distância <i>LEdit</i>	30
3.3	Representação de uma <i>Slim-tree</i> em (a) e sua estrutura lógica em (b).	31
3.4	Descarte de elementos com o uso da desigualdade triangular.	34
3.5	Exemplos de otimização realizada pelo método <i>Slim-down</i>	37
5.1	Desempenho de consultas realizadas após remoções, comparando o algoritmo proposto (com dois valores diferentes de TOM) com o algoritmo de remoção apenas marcando os representantes removidos. Número médio de acessos a disco (primeira coluna), número médio de cálculos de distâncias (segunda coluna) e tempo médio em milisegundos (terceira coluna) em consultas k-NN variando k.	56
5.2	Desempenho de consultas realizadas após remoções seguidas de inserções, comparando algoritmo proposto (com dois valores diferentes de TOM) com o algoritmo de remoção apenas marcando os representantes removidos. Número médio de acessos a disco (primeira coluna), número médio de cálculos de distâncias (segunda coluna) e tempo médio em milisegundos (terceira coluna) em consultas k-NN variando k.	58

5.3	Comparação do desempenho de consultas realizadas sobre a estrutura original e sobre a estrutura resultante do processo de remoção e reinserção de 500 elementos, com o conjunto <i>Cidades</i> (TOM = 60%) e <i>Letras</i> (TOM = 45%). Número médio de acessos a disco (primeira coluna), número médio de cálculos de distâncias (segunda coluna) e tempo médio em milisegundos (terceira coluna) em consultas k-NN variando k.	59
5.4	Execução da técnica <i>Push Pull</i> removendo 2 elementos por nó-folha.	62
5.5	Comparação entre as otimizações obtidas pela Técnica proposta e pelo <i>Slim-Down</i>	63
5.6	Comparação do tempo total de processamento (primeira linha), número médio de acessos a disco (segunda linha) e número médio de cálculos de distância (terceira linha) para processamento de 500 consultas 10-NN, e <i>relative fat-factor</i> (quarta linha) de uma <i>Slim-tree</i> não otimizada, uma <i>Slim-tree</i> otimizada pelo <i>Slim-down</i> e <i>Slim-trees</i> otimizadas com a técnica <i>Push-pull</i> (variando o número de elementos removidos por nó) e <i>Smart Push-pull</i>	65
5.7	Comparação do tempo total de processamento (primeira linha), número médio de acessos a disco (segunda linha) e número médio de cálculos de distância (terceira linha) para processamento de 500 consultas variando o valor de <i>k</i> entre 5 e 50, sobre uma <i>Slim-tree</i> não otimizada, uma <i>Slim-tree</i> otimizada pelo <i>Slim-down</i> e <i>Slim-tree</i> otimizada com o método <i>Smart Push-pull</i>	66
5.8	Comparação do tempo total de processamento, número médio de acessos a disco e número médio de cálculos de distância para processamento de 500 consultas 10-NN realizadas sobre uma <i>Slim-tree</i> não otimizada, uma <i>Slim-tree</i> otimizada pelo <i>Slim-down</i> e uma <i>Slim-tree</i> otimizada com a técnica <i>Smart Push-pull</i> , variando o tamanho do conjunto de dados	67
5.9	Comparação do tempo total de processamento para otimização com o <i>Slim-down</i> e <i>Smart Push-pull</i> , comparado ao tempo para construção da árvore inicial, variando o tamanho do conjunto.	67
5.10	Comparação dos custos das atualizações realizadas removendo e reinserindo os elementos e realizadas pelo algoritmo de atualização proposto, variando o número de atualizações realizados antes das consultas.	70
5.11	Comparação dos desempenho em consultas após a atualizações realizadas removendo e reinserindo os elementos e pelo algoritmo de atualização proposto. Tempo total (em ms) (primeira coluna), número médio de acessos a disco (segunda coluna) e número médio de cálculos de distâncias (terceira coluna) em consultas 10-NN, variando o número de atualizações realizados antes das consultas.	71

6.1	Métrica para instantes: $d_{ti}(t_1, t_2) = t_1 - t_2 $	76
6.2	Métrica para intervalos : $d_{tp}(t_1, t_2) = M(t_1) - M(t_2) + I(t_1) - I(t_2) $. . .	77
6.3	Espaço métrico mapeado em um espaço mutidimensional \mathbb{R}^3 com uma função de distância L_2 . O tamanho do lado do cubo que cobre o conjunto de dados é dado por $\ell_s = \frac{1}{\sqrt{3}} \cdot \delta_{s_max}$	81
6.4	Exemplos de imagens da biblioteca de imagens ALOI.	83
6.5	Elementos relevantes para diferentes elementos centrais de consulta. Resultados para 5-NNq (a) e 7-NNq (b).	85
6.6	<i>Distance plot</i> da componente temporal T	85
6.7	<i>Distance plots</i> das componentes métricas S dos conjuntos: (a) <i>Histogramas</i> (b) <i>Zernike</i> (c) <i>Histogramas métricos</i>	87
6.8	<i>Precisões médias</i> de consultas k -NN variando o valor de p_s para os três conjuntos de dados. Os pontos mostram as <i>precisões médias</i> relativas às consultas realizadas sobre o espaço métrico-temporal, e a linha tracejada mostra as <i>precisões médias</i> relativas às consultas realizadas somente sobre a componente métrica.	87
6.9	Comparação dos resultados de consultas 10-NN utilizando o espaço métrico-temporal com os pesos estimados com resultados obtidos com a utilização apenas da componente métrica.	88
6.10	<i>Distance plots</i> da componente métrica do conjunto <i>Histogramas métricos</i> calculados com diferentes taxas de amostragem sobre o conjunto original. .	89
6.11	Exemplos de consultas no espaço mapeado: (a) estimativa do estado do paciente P_A com 15 meses de tratamento, (b) estimativa do estado do paciente P_A com 6 meses de tratamento.	93
6.12	Tipos de consultas realizadas nos experimentos.	94
6.13	Consultas 10- NN realizadas sobre o espaço mapeado utilizando o conjunto <i>Histogramas</i>	95
6.14	Consultas 10- NN realizadas sobre o espaço mapeado utilizando o conjunto <i>Zernike</i>	96
6.15	Avaliação da qualidade dos mapeamentos.	97
6.16	Exemplos de visualização: a)objetos visualizados; b) visualização bidimensional em planos paralelos; c) visualização utilizando o modelo métrico-temporal.	99
6.17	Exemplos de imagens do conjunto MRI.	103
6.18	Conjunto de imagens MRI: <i>Distance plots</i> de <i>Histogramas Métricos_MRI</i> , <i>Haralick_MRI</i> e <i>Zernike_MRI</i>	105
6.19	Conjunto de imagens MRI: <i>precisão média</i> dos descritores individuais e da combinação entre pares de descritores com a variação do fator de escala entre eles.	106

6.20 Conjunto de imagens MRI: Combinação dos descritores com o método FPM comparada com a utilização de descritores individualmente.	106
6.21 Conjunto de imagens CT_Pulmão_ROIs: <i>Box counting plots</i> dos conjuntos <i>Haralick Pulmão</i> e <i>Zernike Pulmão</i>	107
6.22 Conjunto de imagens CT_Pulmão_ROIs: <i>precisão média</i> dos descritores individuais e da combinação entre eles com a variação do fator de escala. . .	108
6.23 Conjunto de imagens CT_Pulmão_ROIs: Combinação dos descritores com o método FPM comparada com a utilização dos descritores individualmente.	108
6.24 Aplicação desenvolvida para avaliação do método FPM: exemplo de uma consulta aos 5 vizinhos mais próximos utilizando o conjunto de imagens MRI.	109

Lista de Tabelas

5.1	Conjuntos de dados utilizados nos experimentos.	54
5.2	Comparação entre os custos dos algoritmos de remoção apenas marcando os representantes removidos e o algoritmo de remoção proposto.	55
5.3	Conjuntos de dados utilizados nos experimentos.	64
6.1	Conjuntos de dados utilizados nos experimentos.	83
6.2	Tempo de processamento para obtenção de p_s e p_t	89
6.3	Conjuntos de dados utilizados nos experimentos.	93
6.4	Conjuntos de dados utilizados nos experimentos.	104

Introdução

1.1 Considerações Iniciais

Os Sistemas de Gerenciamento de Bancos de Dados (SGBD) foram projetados para suportar dados de tipos numéricos e pequenas cadeias de caracteres. Sobre esses dados existem fundamentalmente dois tipos de operadores de comparação que são amplamente utilizados em SGBD: operadores para comparação por igualdade e operadores relacionais. Os operadores de comparação por igualdade ($=$ e \neq) podem ser universalmente aplicados a quaisquer tipo de dados, pois é sempre possível decidir se dois elementos são iguais ou não. Os operadores relacionais ($<$, \leq , $>$ e \geq) necessitam que os dados comparados estejam em domínios que atendam à chamada “Relação de Ordem Total” (ROT). Esta propriedade permite comparar quaisquer pares de elementos de dados e decidir qual deles precede/sucede ao outro. Juntos, os operadores relacionais e por igualdade compõem os operadores mais comuns encontrados em SGBD tradicionais.

No entanto, os requisitos impostos por muitas das aplicações mais novas têm levado à necessidade de que estes os SGBD suportem tanto outros tipos de dados quanto outros tipos de consultas que sejam adequadas a eles. Um exemplo de tipo de dados que requer tipos de consultas específicas são os chamados tipos de dados espaciais, como por exemplo Sistemas de Informações Geográficas (SIG) e Sistemas de Apoio à Projetos em Engenharia. Nesses sistemas são necessários operadores de consultas específicos que envolvem a noção

de dimensões espaciais, tais como as consultas topológicas (intercepta, adjacente a, etc.) e as consultas cardinais, baseadas em ângulos (ao norte, a sudeste, acima, a esquerda, etc.) [Gaede & Günther, 1998].

Já os domínios de dados complexos, como por exemplo dados multimídia (imagens, áudio, textos longos), séries temporais, sequências genéticas, etc., geralmente não apresentam a ROT, e muitos deles tampouco apresentam a noção de dimensões, impedindo também consultas topológicas ou cardinais. De fato, os operadores relacionais não são aplicáveis para dados de tipos complexos - por exemplo, genericamente, não é possível ordenar imagens, a menos que elas sejam associadas a algum atributo extra não complexo (por exemplo: nome, data, etc). Em domínios de dados complexos, mesmo as operações de comparação por igualdade têm pouca utilidade: continuando com o exemplo de imagens, a possibilidade duas imagens médicas serem exatamente iguais é muito pequena, mesmo que sejam de exames da mesma pessoa e com o mesmo aparelho. Para dados complexos, consultas por similaridade tornam-se a solução mais adequada, e o grau de similaridade entre os dados é o fator mais importante [Faloutsos, 1997].

Os operadores de consulta por similaridade se aplicam a muitos dos tipos de dados complexos, incluindo os dados espaciais e diversos outros. Com a necessidade crescente do suporte a dados multimídia em SGBD, os operadores por similaridade vêm despertando muito interesse, principalmente para a recuperação por conteúdo de dados complexos.

Para os operadores de consulta por similaridade serem aplicáveis a um determinado domínio de dados, é necessário que esteja definida no domínio uma função de similaridade, também chamada de função de distância, que atenda às propriedades de simetria, não-negatividade e desigualdade triangular. Funções que atendam a essas três propriedades associadas a um tipo de dados criam o que se denomina um domínio métrico. A definição formal para domínios métricos é apresentada no Capítulo 2.

Uma função de distância quantifica quão similar são dois elementos, e habilita a representação de consultas de seleção baseadas na similaridade dos elementos. Usualmente, a função de distância pode ser considerada como uma “*caixa preta*”, geralmente definida por um especialista no domínio da aplicação. Uma consulta

1.2 Motivação

por similaridade recupera os elementos que atendem a um determinado critério de similaridade, expresso com referência a um elemento do domínio de dados $s_q \in \mathbb{D}$, chamado “elemento central da consulta”.

Para conjuntos de dados em espaços métricos, onde não são consideradas relações geométricas e somente existem os elementos e as distâncias entre eles, diversas pesquisas levaram ao desenvolvimento das estruturas de indexação chamadas genericamente de Métodos de Acesso Métrico (MAM), sendo a criação de MAM eficientes um dos grandes problemas que vêm sendo pesquisado no suporte à consultas por similaridade em SGBD.

1.2 Motivação

Independentemente do domínio de um tipo de dados, sempre existe a possibilidade de um elemento de dados evoluir no tempo. Existem diversos modelos para representar o tempo em SGBD, sendo que o tratamento em domínios que suportam a ROT e em domínios espaciais apresentam requisitos distintos, e vêm despertando interesse da comunidade. Os modelos de tempo para domínios que suportam a ROT costumam considerar uma ou mais dimensões temporais, em particular a dimensão de tempo de transação e a de tempo de validade semântica do elemento no mundo real [Snodgrass, 1995]. Domínios espaciais têm recebido bastante atenção no contexto de aplicações com dados geo-referenciados, nos chamados modelos espaço-temporais [Sellis, 1999, Erwig & Schneider, 2002].

No entanto, não existem trabalhos associando tempo a dados em domínios métricos, embora muitas aplicações tratem de dados em domínios métricos que evoluem no tempo, como por exemplo em:

casos clínicos - acompanhamento de pacientes através do monitoramento de resultados de exames clínicos;

imagens médicas - imagens de exames baseados em imagens que acompanham a evolução de determinado caso clínico;

sensoreamento de edificações civis - acompanhamento do estado de edificações, como pontes e torres, baseadas em sensores colocados em estruturas;

sensoreamento industrial - manutenção de equipamentos industriais, como fornos e dutos em refinarias;

bolsa de valores - acompanhamento do mercado de ações pelos índices financeiros associados às diversas empresas do mercado.

monitoramento ambiental - acompanhamento de indicadores ambientais para avaliação da qualidade da água, ar ou solo em ecossistemas.

Os MAM existentes consideram que cada elemento de dado representa um objeto imutável no tempo. A grande maioria dos MAM publicados não tem descrita sequer a operação de remoção de dados, as quais são realizadas restringindo-se a marcar o dado como removido, sem efetivamente removê-lo da estrutura. Os MAM existentes também não dispõem de suporte para a manutenção da história de evolução de um elemento ou a possibilidade de previsão de estados futuros.

1.3 Objetivos e Contribuições

O objetivo deste trabalho foi abordar a dinamicidade e a inclusão do conceito de evolução temporal de dados em espaços métricos armazenados em MAM, criando um suporte consistente para o tratamento do tempo em operações de atualização e busca por similaridade em MAM. O trabalho foi realizado em duas frentes de atividades. A primeira abordou a indexação de dados em espaços métricos que sofram atualização com frequência, sem que seja necessário manter o histórico das atualizações. Foram incluídas as operações de remoção e atualização de elementos, aumentando o conjunto de operações suportadas pelos MAM, que restringiam-se as operações de inclusão de dados e buscas em dados estáticos. Esta frente de atividades foi genérica, e os resultados são universalmente aplicáveis a domínios de aplicação que trabalhem com dados em espaços métricos, independente de haver a necessidade de tratamento temporal aos dados.

A segunda frente de atividades correspondeu a incorporar de fato o suporte temporal a dados em espaços métricos, com suporte a evolução histórica e predição de estados futuros dos elementos de dados métricos. Um modelo de dados métrico-temporal foi

1.4 Organização do Trabalho

proposto para o caso em que o tempo deve ser levado em consideração durante o cálculo da similaridade entre os elementos de dados métricos, e foram propostos métodos para análise das trajetórias desses dados no decorrer do tempo. Também foi desenvolvido um método de balanceamento de múltiplos descritores de imagens, criado a partir do algoritmo de balanceamento entre componentes do espaço métrico-temporal.

1.4 Organização do Trabalho

Neste capítulo foram apresentados o contexto do trabalho, a motivação para o desenvolvimento da pesquisa, os objetivos e os resultados obtidos.

No Capítulo 2 são apresentados conceitos fundamentais para recuperação por conteúdo e consultas por similaridade. Um histórico da evolução das estruturas de indexação para os diversos domínios de dados, com ênfase em domínios métricos é apresentado Capítulo 3. No capítulo 4 são apresentados os conceitos básicos no desenvolvimento de bases de dados temporais. São apresentados vários modelos de dados temporais e são também discutidos o caso específico de objetos móveis. No Capítulo 5 são apresentados os resultados da primeira frente de atividades deste trabalho de doutorado: algoritmos de remoção e atualização de dados para MAM, e um novo método de otimização para MAM dinâmicos.

No Capítulo 6 é apresentado um modelo de representação para dados métrico-temporais, e são discutidas formas de analisar trajetórias desses dados. Também nesse capítulo é apresentado um método de balanceamento para integrar múltiplos descritores de imagens, criado a partir de uma modificação do algoritmo de balanceamento do espaço métrico-temporal. Por fim, no Capítulo 7 são apresentadas as considerações finais, com as principais contribuições e possibilidades de trabalhos futuros.

Recuperação por conteúdo e consultas por similaridade

2.1 Considerações Iniciais

Durante todo o desenvolvimento dos sistemas relacionais e objeto-relacionais até hoje, os dados suportados são majoritariamente os de tipo numérico ou textual curto. Os SGBD atuais aproveitam a propriedade de ROT existente entre os elementos desses domínios de dados elementares para executar as operações de consulta e atualização dos dados. Essa propriedade garante que, dados dois elementos distintos quaisquer do mesmo domínio, sempre se pode dizer qual elemento precede o outro. Mesmo quando uma operação de busca envolve apenas operações de comparação por igualdade, as estruturas de indexação usadas para agilizar a consulta dependem de que a propriedade de ROT seja atendida pelo respectivo domínio de dados.

No entanto, dados mais complexos em geral não possuem ROT - genericamente, não é possível ordenar imagens, por exemplo, sem a utilização de um atributo adicional. Nesses domínios de dados, apesar de aplicáveis, os operadores de comparação por igualdade têm pouca utilidade, uma vez que a existência de dois elementos exatamente iguais é rara [Faloutsos, 1996], e são praticamente inúteis para operações de recuperação em bases de dados. Nesses casos, o grau de similaridade entre os dados é o fator mais importante.

Uma classe de operadores mais adequada para manipular esses dados são os operadores por similaridade [Faloutsos, 1997]. Para que possam ser empregados, o domínio de dados em questão deve dispor de uma função de dissimilaridade entre os elementos, também chamada função de distância, que deve atender à algumas propriedades para ser considerada métrica. Domínios de dados que dispõem de uma função de distância métrica são chamados domínios (ou espaços) métricos.

Para conjuntos de dados em espaços métricos somente existem os elementos e as distâncias entre eles. As estruturas de indexação aplicáveis a esse domínio de dados são chamadas genericamente MAM.

Na Seção 2.2 são apresentados os mecanismos para recuperação baseada em conteúdo em dados multimídia. Na Seção 2.3 discute-se a utilização de vários conjuntos de características na representação de dados complexos. Em seguida, nas Seções 2.4 e 2.5 são apresentados conceitos fundamentais para o processo de recuperação por conteúdo, como a definição de espaços métricos e os principais operadores de consulta por similaridade. Por fim, na Seção 2.6 são apresentados alguns conceitos sobre a aplicação da teoria dos fractais em bases de dados, necessários para o desenvolvimento desse trabalho.

2.2 Recuperação por conteúdo em dados multimídia

Muitas aplicações que envolvem a comparação de dados multimídia, como imagens, extraem características dos objetos multimídia, criando um ‘vetor de características’ para representar cada objeto. A comparação é feita utilizando seu vetor de características no lugar do objeto propriamente dito. Esse processo de comparação permite que se realize a chamada “Recuperação de dados por conteúdo” (Content-based retrieval - CBR), em contraposição à recuperação dos dados por descrição, onde se associa um texto a cada objeto multimídia (por exemplo, o laudo de uma imagem médica) e se realiza a busca comparando palavras-chaves dos textos [Amato et al., 1997, Ashwin et al., 2002].

A escolha das características ou propriedades mais importantes, que melhor representem a informação no processo de CBR, normalmente é feita por um especialista no domínio da aplicação[Traina & Traina Jr., 2003]. Escolhidas essas características, elas

são extraídas de cada um dos objetos que fazem parte da base de dados. Cada objeto da base de dados passa a ser representado então por seu ‘vetor de características’, que pode ser indexado. Na realização de consultas sobre os dados indexados, os objetos de referência utilizados na consulta têm suas características extraídas, as quais são a seguir utilizadas para realizar-se a busca. Portanto, as operações de busca são efetuadas sobre as características extraídas dos dados, as quais os descrevem [Faloutsos, 1996] [Smeulders et al., 2000].

Tomando por exemplo o domínio de imagens, características básicas usadas com frequência para compor os vetores de características [Torres & Falcão, 2006] são as distribuições de cor [Traina et al., 2003, Stehling et al., 2002], forma [Zhang & Lu, 2004, Guliato et al., 2008, Andaló et al., 2010] ou textura [Howarth & Rüger, 2004, Montoya-Zegarra et al., 2008]. Na Figura 2.1 é apresentado um exemplo de extração de características utilizando imagens com 256 níveis de cinza, tendo como vetores de características o histograma de cor normalizado.

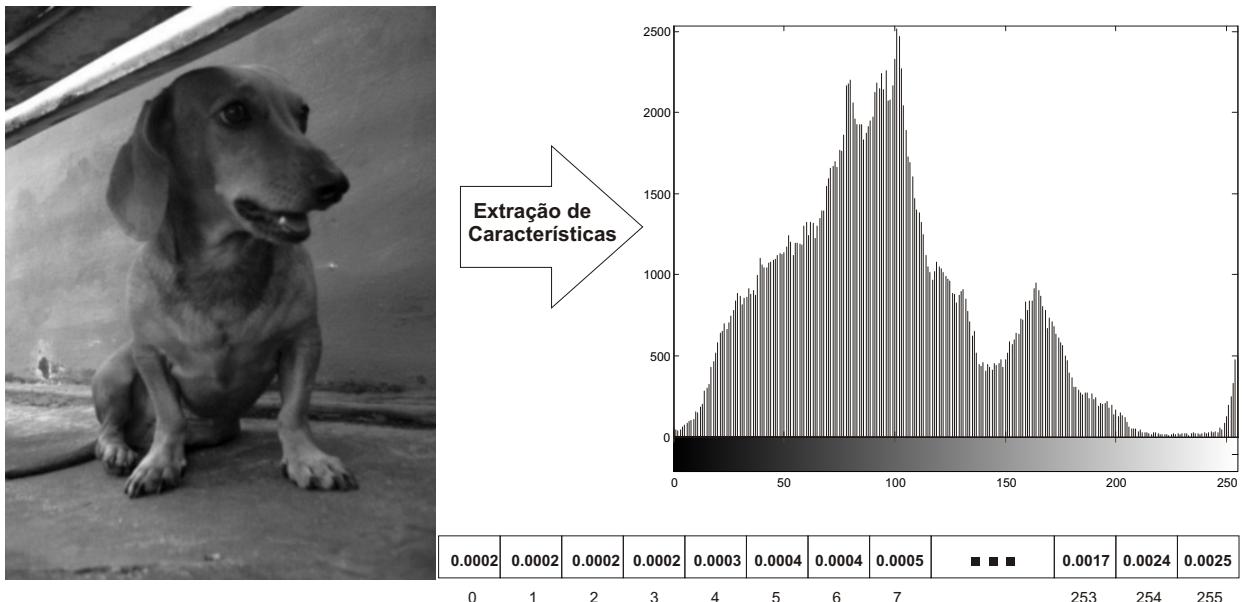


Figura 2.1: Exemplo de extração de características: histograma normalizado com 256 níveis de cinza.

Os momentos de Zernike [Khotanzad & Hong, 1990] são capazes de representar formas complexas contidas nas imagens. Já os descritores de Haralick [Haralick et al., 1973] representam as texturas da imagem, e são extraídos da matriz de co-ocorrência.

Em [Torres & Falcão, 2006] são apresentados os principais descritores de imagens para recuperação por conteúdo.

Embora o processamento de uma consulta usando CBR seja mais trabalhoso, e em alguns casos sujeitos a erros devido à uma baixa discriminação dos dados pelas características extraídas, ele é considerado na maioria das vezes superior à busca utilizando palavras chaves, pois independe da intervenção humana para criar as descrições, e da habilidade do analista em gerá-las [Müller et al., 2004, Datta et al., 2008].

Dessa maneira, a busca em um conjunto de dados multimídia pode ser feita utilizando estruturas de indexação espaciais ou métricas [Petrakis & Faloutsos, 1997] [Petrakis et al., 2002]. Infelizmente, a quantidade de características extraídas por exemplo, no tratamento de imagens, ultrapassa facilmente a casa das centenas, e a grande maioria dos métodos de acesso multidimensional, tais como a R-tree [Guttman, 1984] e suas derivadas R⁺-tree [Sellis et al., 1987] e R*-tree [Beckmann et al., 1990] (veja-se por exemplo [Gaede & Günther, 1998] para uma descrição de métodos de acesso multidimensionais), degradam rapidamente com o aumento da dimensionalidade dos dados, tornando-se inviáveis para dimensionalidades maiores que uma dezena. Isso ocorre devido ao fato que conforme a dimensionalidade aumenta, os espaços multidimensionais tendem a ser muito mais esparsos, num fenômeno conhecido como a “maldição da dimensionalidade” (*dimensionality curse*) [Talavera, 1999, Korn et al., 2001, Jeong et al., 2007, Volnyansky & Pestov, 2009]. Em altas dimensões as distâncias entre os elementos tendem a se homogenizar, fazendo com que praticamente toda a estrutura de indexação tenha quer ser percorrida para a obtenção da resposta exata. Um comportamento equivalente pode ser observado também em espaços métricos [Chávez & Navarro, 2001], apesar da inexistência de coordenadas não permitir a análise de complexidade em termos de dimensões.

Diversos métodos de acesso vêm sendo criados especificamente para dar suporte à recuperação por conteúdo em dados multimídia [Yoshitaka & Ichikawa, 1999], incluindo buscas aproximadas [Papadias et al., 2000, Jiang et al., 2000, Bueno et al., 2005b], buscas baseadas em conteúdo semântico [Roddick & Spiliopoulou, 1999,

2.3 Combinação de múltiplos descritores

Megalou & Hadzilacos, 2003] e buscas especializadas em algum tipo de dados, tais como imagem [Atnafu et al., 2004, Wang et al., 2003, Huang & Dai, 2003] vídeo [Adjerooh et al., 1999, Liu & Chen, 2002], áudio [Tseng, 1999, Downie & Nelson, 2000] dados espaciais [Kollios et al., 2005, Mokbel et al., 2003] dados de bio-informática [Lane et al., 2000] e séries temporais [Povinelli & Feng, 2003].

2.3 Combinação de múltiplos descritores

A utilização de vários conjuntos de características para representar dados complexos, como imagens, tende a melhorar significativamente os resultados obtidos na recuperação por conteúdo. Devido ao aumento da variedade e do número de imagens, a utilização de apenas um descritor torna-se insuficiente [Stejić et al., 2003].

Uma maneira trivial para combinar vários conjuntos de características é concatenar todos os valores em um “supervetor”, e então usar uma função de distância para compará-los. Porém, os conjuntos podem apresentar atributos com valores bem diferentes, e torna-se necessário um processo de normalização para reduzir os efeitos dessas diferenças, evitando que características com valores mais altos dominem o resultado final [Aksoy & Haralick, 2001]. Depois de normalizados, os supervetores são comparados usando uma única função de distância definida sobre o espaço dimensional gerado.

Porém, com o aumento da dimensionalidade (“maldição da dimensionalidade”) o desempenho das estruturas de indexação degradam rapidamente. Resultados apresentados em [Bustos et al., 2004] mostram que o aumento indiscriminado do número de características combinadas (estratégia discutida abaixo) também prejudica a precisão dos resultados.

Além disso, existe um estreito relacionamento entre um conjunto de características e a métrica usada para comparar os elementos desse conjunto, e que uma melhor integração desse binômio é um requisito para melhorar a qualidade dos resultados [Bugatti et al., 2008, Torres et al., 2009]. Portanto, provavelmente não serão alcançados os melhores resultados se sempre for usada a mesma métrica para comparar diferentes conjuntos de características. Em [Silva et al., 2009a], sugere-se a utilização

de informações adicionais para uma melhor definição desse binômio. Em um caso de estudo com exames de pulmão, realizado com o auxílio de especialistas do HCFMRP-USP, as imagens foram divididas em classes de acordo com a patologia apresentada. Foi constatado que cada classe de imagens é melhor evidenciada utilizando-se um par específico de descritor/métrica. Ou seja, a partir da hipótese de diagnóstico do paciente, pode-se identificar o que o usuário procura, ou espera achar, nas imagens, e a partir disso definir o melhor binômio descritor/métrica. O estudo mencionado apresenta um exemplo de como a integração de conhecimento de especialistas associado a informações relacionadas às imagens podem ser utilizadas para melhorar a recuperação por conteúdo, definindo condições de contorno para as consultas por similaridade.

Em uma abordagem mais flexível para combinar múltiplos descritores, são calculadas as similaridades de cada conjunto de características separadamente, e então essas similaridades parciais são agregadas usando uma função de composição. Várias estratégias têm sido propostas para definir o balanceamento da combinação de vários descritores. Em alguns trabalhos [Heesch & Rüger, 2002], [Caicedo et al., 2007] são realizadas buscas exaustivas pela melhor combinação. Grande parte das técnicas que combinam múltiplos descritores permitem apenas a combinação linear entre eles. Em [Torres et al., 2009] foi proposto um modelo que pode definir funções mais complexas utilizando programação genética, e assim proporcionar maior flexibilidade e melhor expressar a percepção visual dos usuários.

Em [Bustos et al., 2004] é proposto um método que calcula o balanceamento entre os descritores dinamicamente para cada consulta, utilizando para isso um conjunto de treinamento com imagens pré-classificadas. Visando aumentar a flexibilidade dos sistemas de recuperação por conteúdo, vários trabalhos [Rui et al., 1998, Rui & Huang, 2000, Stejić et al., 2003, Ferreira et al., 2008] utilizam técnicas de realimentação por relevância visando captar as preferências dos usuários e utilizá-las para combinar múltiplos descritores. Em [Ferreira et al., 2008], utiliza-se o mesmo modelo de [Torres et al., 2009] em conjunto com técnicas de realimentação por relevância. Com isso, a função de similaridade pode ser mais complexa que a combinação linear entre os descritores, e é

definida dinamicamente nas consultas, refletindo as preferências dos usuários.

2.4 Espaços métricos

Um espaço métrico é definido como $\langle \mathbb{S}, d \rangle$, onde \mathbb{S} é o conjunto de todos os elementos que atendem às propriedades do domínio e d é uma função de distância, ou métrica, entre esses elementos, definida como $d : \mathbb{S} \times \mathbb{S} \rightarrow \mathbb{R}^+$. A função de distância é dependente do tipo dos elementos e da aplicação, e quanto menor o valor resultante da aplicação da função de distância entre dois elementos, mais semelhantes eles são.

Dados s_1, s_2 e $s_3 \in \mathbb{S}$, uma função de distância (métrica) deve satisfazer as seguintes propriedades [Traina Jr. et al., 2002a]:

1. *Simetria*: $d(s_1, s_2) = d(s_2, s_1)$;
2. *Não Negatividade*: $0 \leq d(s_1, s_2) \leq \infty$, $d(s_1, s_1) = 0$;
3. *Desigualdade Triangular*: $d(s_1, s_2) \leq d(s_1, s_3) + d(s_3, s_2)$;

Um conjunto de dados S é dito estar num espaço métrico se $S \subset \mathbb{S}$.

Se os elementos de um domínio são vetores numéricos de tamanho fixo, tal domínio é denominado Espaço Multidimensional. Em dados multidimensionais, além da função de distância entre os elementos, podem ser exploradas informações geométricas durante o processo de consulta, o que não ocorre com dados em domínios puramente métricos, onde apenas o elemento de referência e a função de distância estão presentes.

Dados em espaços multidimensionais podem ser considerados como estando também em um domínio métrico se for definida uma função de distância (métrica).

Para medir a distância entre dois elementos em um domínio de dados complexo, devido a complexidade dos elementos, o mais comum é comparar vetores de características extraídas dos elementos. Desta maneira, cada elemento passa a ser representado e indexado por seu vetor de características.

Espaços métricos distintos podem ser agregados em um novo espaço métrico, com a composição de suas respectivas métricas em uma nova função que satisfaça as propriedades

de uma métrica. Essas agregações métricas correspondem a produtos de espaços métricos, e a métrica resultante é comumente chamada de *métrica produto* [Searcoid, 2006].

2.4.1 Métricas

Para dados multidimensionais, as funções de distância mais comuns são as da família das normas L_p (ou Minkowski), definidas em 2.1 por:

$$L_p(x, y) = \sqrt[p]{\sum_{i=0}^{d-1} |x_i - y_i|^p} \quad (2.1)$$

onde d é a dimensão do espaço. x e $y \in \mathbb{S}$ são elementos de dimensão d . As funções distância mais utilizadas da família L_p são:

- L_0 - conhecida como *Infinity* ou *Chebychev*;
- L_1 - conhecida como *City block* ou *Manhattan*;
- L_2 - conhecida como Euclidiana;

A Figura 2.2 mostra as diferentes regiões de cobertura relativas às funções de distância citadas acima, para um dado elemento representativo e um dado raio r .

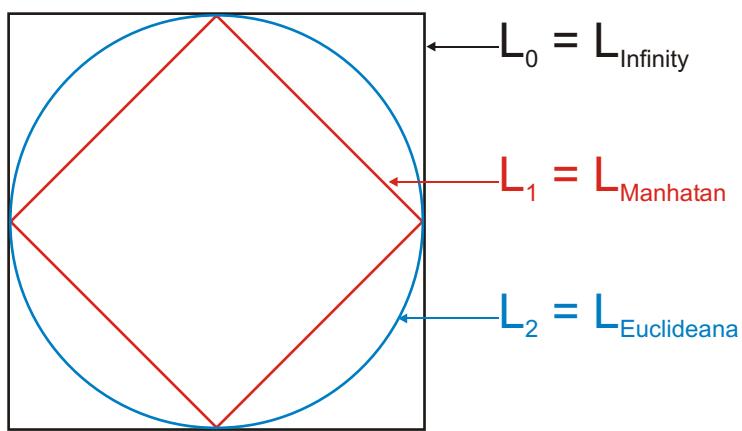


Figura 2.2: Áreas de cobertura de algumas funções de distância da família das normas L_p .

A métrica L_2 , conhecida como distância Euclideana, é a função normalmente utilizada para calcular a distância entre objetos no espaço tridimensional.

Um conjunto de histogramas de cores imagens, onde todos os histogramas tem a mesma dimensão, ou seja, o mesmo número de características, é um exemplo de conjunto de dados em espaço multidimensional. As funções de distância são dependentes do domínio da aplicação, e da maneira como os elementos do domínio são representados.

Outro exemplo é a distância de Canberra, sensível a pequenas variações [Bugatti et al., 2008]:

$$Canberra(x, y) = \sum_{i=0}^{d-1} \frac{|x_i - y_i|}{|x_i| + |y_i|} \quad (2.2)$$

Focando histogramas de imagens, em [Traina et al., 2003] foram propostos os Histogramas Métricos, que além de serem invariantes a transformações geométricas nas imagens, são também invariantes com relação a transformações lineares de brilho. Representadas por histogramas métricos, as imagens não podem ser representadas como pontos no espaço multidimensional, pois a semântica e a quantidade de valores no vetor de características pode ser diferentes para imagens distintas. No entanto, foi definida no mesmo trabalho uma nova função de distância para histogramas métricos, chamada *MHD* - *metric histogram distance*, que propicia uma comparação mais rápida entre imagens.

Conjuntos de palavras de uma linguagem também são dados que não podem ser representados diretamente em espaços multidimensionais. Em conjuntos de sequências de caracteres, a função de distância *Levenshtein*, conhecida como $L_{edit}(x, y)$ [Levenshtein, 1966], pode ser usada para medir a similaridade entre duas palavras. A $L_{edit}(x, y)$ indica a quantidade mínima de símbolos que devem ser substituídos, inseridos ou removidos para transformar uma palavra x em outra y . Por exemplo, $L_{Edit}(\text{"casa"}, \text{"asia"}) = 2$, uma remoção (a letra "c") e uma inserção (a letra "i").

Desde que garantidas as propriedades de uma função de distância métrica, a função de dissimilaridade pode ser considerada como uma “caixa preta”, geralmente definida por um especialista na aplicação.

2.5 Consultas por similaridade

Em domínios métricos, existem dois operadores principais de consulta por similaridade: a consulta por abrangência (“*range queries*”, RQ), que considera os elementos que estejam até uma distância limite do elemento de referência; e a consulta aos vizinhos mais próximos (“*k-nearest neighbor queries*”, $k - NNQ$) [Yianilos, 1993] [Korn et al., 1996] [Braunmüller et al., 2000], que limita os elementos do conjunto resposta a um número máximo.

Considerando um conjunto de elementos $S = \{s_1, s_2, \dots, s_n\}$ de um domínio \mathbb{S} , $S \subset \mathbb{S}$, uma função de distância (métrica) d entre esses elementos, são apresentados a seguir os principais operadores de seleção por similaridade.

2.5.1 Consulta por abrangência

Uma consulta por abrangência recebe como parâmetros um elemento do domínio de dados $s_q \in \mathbb{S}$ (chamado de elemento central da consulta ou elemento de referência) e um grau de dissimilaridade $r_q \geq 0$, e obtém todos os elementos da base de dados S que diferem do elemento da consulta s_q por no máximo a dissimilaridade indicada r_q .

Formalmente, tem-se:

$$\text{range}(s_q, r_q) = \{s_i | s_i \in S, d(s_i, s_q) \leq r_q\}$$

Um exemplo de consulta por abrangência em uma base com dados geográficos seria “selecione as cidades que estejam a uma distância de até 100 quilometros da cidade apresentada como referência”. Na Figura 2.3 pode-se ver uma ilustração desse exemplo.

2.5.2 Consulta aos vizinhos mais próximos

Uma consulta aos vizinhos mais próximos recebe como parâmetros um elemento do domínio de dados $s_q \in \mathbb{S}$ (o elemento de referência, também chamado de elemento central da consulta) e uma quantidade $k > 0$, e obtém os k-elementos da base de dados mais próximos do elemento da consulta.

Formalmente, tem-se:

$$k\text{-}NN(s_q, k) = \{s_i | s_i \in A, A \subseteq S, |A| = k, \forall s_i \in A, s_j \in S - A, d(s_q, s_i) \leq d(s_q, s_j)\}$$

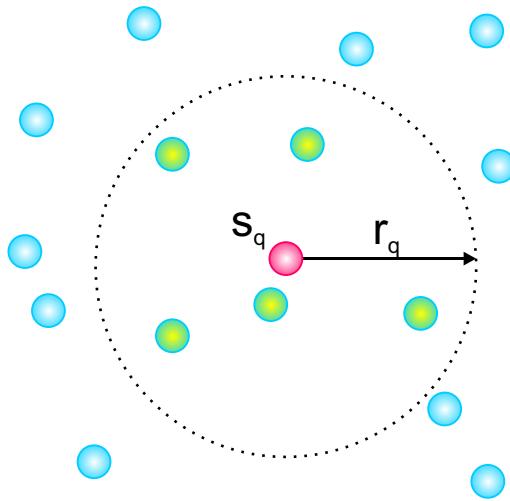


Figura 2.3: Consulta por abrangência.

Um exemplo de consulta de vizinhos mais próximos em uma base de dados com imagens seria “selecione as 4 imagens mais similares à imagem apresentada como referência para a consulta”, como ilustrado na Figura 2.4

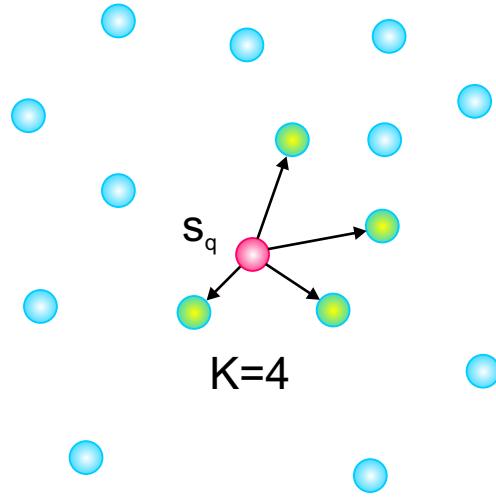


Figura 2.4: Consulta pelos vizinhos mais próximos

2.5.3 Algoritmos para consultas por similaridade

Os algoritmos de consulta por abrangência e aos k-vizinhos mais próximos são aplicáveis em todas as árvores métricas e espaciais [Roussopoulos et al., 1995]. Em ambos os algoritmos, a resposta é ordenada pela distância dos elementos encontrados para o elemento de referência da consulta.

Algoritmos de consultas por abrangência $range(s_q, r_q)$ têm o raio limitante r_q conhecido durante todo o processo de busca. Assim, o algoritmo de consulta por abrangência percorre a estrutura e calcula a distância entre o elemento de referência s_q com os elementos armazenados s_i , e inclui na resposta todos aqueles que estão a uma distância inferior ou igual ao raio de consulta r_q .

Já na consulta por vizinhos mais próximos $k\text{-}NN(s_q, k)$, o raio limitante final da resposta da consulta não é conhecido desde o início da busca. Portanto, o raio limitante é dinâmico, definido inicialmente com valor infinito. Da mesma maneira que o algoritmo de consultas por abrangência, o algoritmo de consulta percorre a estrutura e calcula a distância entre o elemento de referência s_q com os elementos armazenados s_i . Se é encontrado um elemento com distância inferior ao raio limitante, este elemento é inserido na resposta. No caso de já haver k elementos na resposta, o elemento encontrado é inserido e então o elemento mais distante da resposta anterior é cortado. Ao se preencher a resposta com k elementos, o raio limitante passa a ser atualizado a cada inserção com a distância do k -ésimo elemento do conjunto resposta.

Em estruturas baseadas em árvores, uma subárvore somente é percorrida se seus elementos de controle (representantes) e o elemento central da consulta atenderem à propriedade de desigualdade triangular da maneira determinada pelo algoritmo de busca de cada estrutura em particular [Traina Jr. et al., 2002b].

A ordem em que a estrutura de indexação é percorrida não influencia o desempenho das consultas por abrangência, mas pode influenciar muito o desempenho dos algoritmos de consultas aos vizinhos mais próximos: encontrando os elementos mais próximos no início da execução da consulta, o raio limitante dinâmico será reduzido mais rapidamente, aumentando as possibilidades de poda.

Muitos trabalhos foram propostos com o objetivo de acelerar o processo de consultas por similaridade, principalmente $k\text{-}NNQ$ [Roussopoulos et al., 1995, Berchtold et al., 1998, Hjaltason & Samet, 1999, Samet, 2003, Chen et al., 2007, Tao et al., 2009, Bustos & Navarro, 2009].

Um algoritmo de $k\text{-}NNQ$ que utiliza a dimensão fractal do conjunto de dados indexados

para estimar o raio da resposta da consulta foi proposto em [Arantes et al., 2003]. Em [Bueno et al., 2005a] são utilizados algoritmos genéticos para encontrar respostas aproximadas para k -NNQ e RQ. Em [Patella & Ciaccia, 2009] é apresentada uma revisão bibliográfica das várias propostas para consultas por similaridade aproximadas.

Algoritmos para buscas incrementais aos vizinhos mais próximos são discutidas em [Hjaltason & Samet, 1999]. Tais algoritmos baseiam-se no fato que após a realização de uma consulta aos k vizinhos mais próximos, o vizinho $k + 1$ pode ser obtido sem a necessidade de reiniciar a busca, tornando-os mais eficientes. Em [Park & Kim, 2003] é apresentada uma nova versão do algoritmo incremental para consultas com atributos não-espaciais em seu predicado, utilizando-os para podas.

Vários trabalhos propuseram algoritmos k -NN para aplicações específicas. Em [Koudas et al., 2004] são apresentadas consultas aproximadas de k -NN para aplicações de “data-stream”, em que os dados chegam continuamente e podem ser acessados somente uma vez. Em [Papadias et al., 2003] são apresentados algoritmos para consultas em redes espaciais (*spatial network databases*). Já em [Ku et al., 2006] é apresentado o protótipo de um sistema baseado em informações de tráfego com algoritmos de k -NN desenvolvidos para “travel time networks” (TTN), que utilizam o tempo de viagem ao invés da distância, utilizada nas “spatial networks”.

Outros algoritmos baseados nas consultas ao vizinhos mais próximos são chamados de consultas k -NN contínuas [Song & Roussopoulos, 2001, Huang et al., 2009]. Em [Tao et al., 2002] é proposta a consulta “continuous nearest-neighbor” (CNN), que recupera os elementos mais próximos de todos os pontos de um segmento de linha. Em [Hu & Lee, 2006] é proposta a consulta “range nearest-neighbor” (RNN), que dado um conjunto de dados de dimensão d , recupera os elementos mais próximos de todos os pontos de um hyper-retângulo de dimensão d .

Variações que levam a algoritmos bem mais custosos são as consultas aos vizinhos mais próximos reversos (“reverse nearest neighbor queries” [Tao et al., 2006, Lee et al., 2008, Achtert et al., 2009, Tran et al., 2009]). Essas consultas retornam quais são os elementos do conjunto de dados que têm o elemento central da consulta como o vizinho mais

próximo, com as correspondentes variações que permitem retornar os elementos que têm o elemento central da consulta como um dos seus até k elementos mais próximos [Tao et al., 2006, Xia et al., 2005]. Embora essas consultas sejam operações de seleção, as consultas por similaridade reversas têm complexidade de execução quadrática, semelhante aos operadores de junção por similaridade.

2.6 Teoria dos fractais aplicada a bases de dados

Um fractal é definido pela propriedade de auto-similaridade, independentemente da escala ou tamanho. Dessa forma, partes de um objeto fractal são direta ou estatisticamente similares ao fractal como um todo [Schroeder, 1991]. A teoria dos fractais vem sendo aplicada com sucesso na modelagem de conjuntos de dados reais. Experimentos mostraram que a distribuição das distâncias entre pares de elementos na maioria dos conjuntos de dados reais apresenta auto-similaridade. Dessa maneira, estes conjuntos podem ser considerados como conjuntos fractais [Faloutsos & Kamel, 1994].

Um resultado interessante da teoria dos fractais é que qualquer fractal apresenta uma dimensão intrínseca, independente do espaço onde o objeto está imerso, e que pode ser medida pela sua dimensão fractal. A teoria dos fractais define várias medidas para a dimensão fractal. A *dimensão fractal de correlação* D_2 é uma das mais utilizadas em aplicações relacionadas a bases de dados. Conhecendo D_2 de um conjunto de dados métrico, por exemplo, pode-se estimar suas propriedades como sendo similar a um conjunto de dados dimensional, com o mesmo número de dimensões.

O método *box-counting* [Belussi & Faloutsos, 1995, Traina Jr. et al., 2000c, Attikos & Doumpos, 2009] é comumente aplicado para estimar a dimensão fractal de dados multidimensionais. Dado um conjunto de dados imerso em um espaço E -dimensional, divide-se recursivamente o hipercubo que envolve o conjunto de dados em células de lado r , e conta-se o número de elementos em cada célula, até que r tenda a zero. O gráfico em escala log-log gerado por esse processo é chamado de *box-counting plot*. Para a dimensão de correlação fractal D_2 , os valores representados no eixo vertical correspondem à somatória ao quadrado do número de pontos que incidem a células de

lado r , e o eixo horizontal apresenta os valores de r . Os gráficos gerados por conjunto de dados perfeitamente fractais são linhas retas, e a maioria dos conjunto de dados reais resultam em curvas que também podem ser ajustadas por linhas retas. A inclinação da linha é uma boa estimativa da dimensão intrínseca do conjunto de dados.

Para dados métricos, a dimensão intrínseca pode ser estimada pelo método proposto em [Traina Jr. et al., 2000a]. Dado um conjunto de elementos em um conjunto de dados com uma métrica d , o número médio k de vizinhos dentro de uma distância r é proporcional a r elevado a um valor \mathcal{D} . Portanto, a contagem de pares $PC(r)$ de elementos dentro da distância r segue a seguinte lei de formação:

$$PC(r) = K_p \cdot r^{\mathcal{D}} \quad (2.3)$$

onde K_p é uma constante de proporcionalidade.

O gráfico obtido pelo cálculo da Equação 2.3 é chamado de *distance plot*. Utilizando escalas log-log, os *distance plot* também podem ser ajustados por uma linha reta, e pode-se estimar o expoente \mathcal{D} da Equação 2.3, chamado *distance exponent*, pela inclinação dessa linha. O valor do *distance exponent* é muito próximo à dimensão fractal de correlação D_2 .

Na Figura 2.5 é mostrada a distribuição espacial e o *distance plot* de um conjunto de coordenadas geográficas de intersecções de vias em *Montgomery County, MD, EUA*.

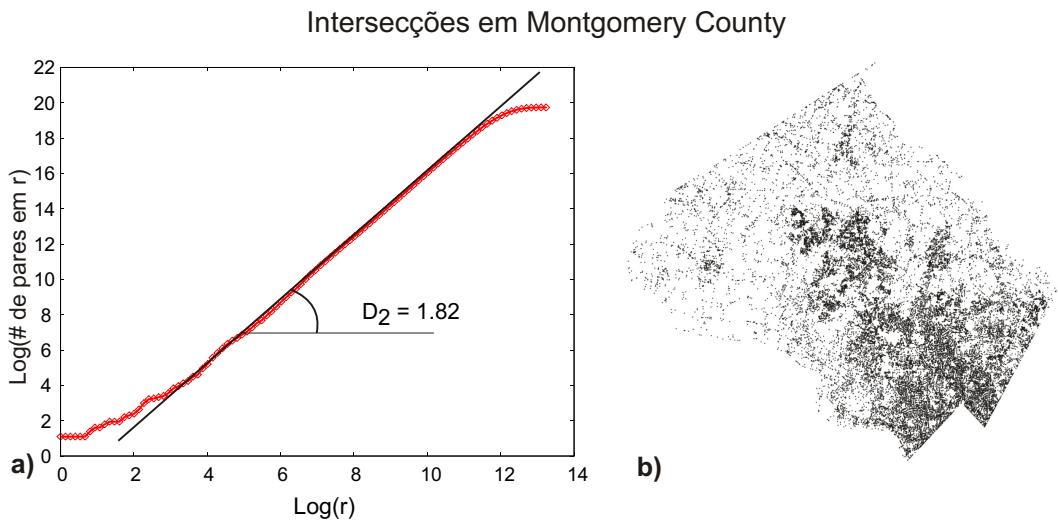


Figura 2.5: Coordenadas geográficas de intersecções de vias em *Montgomery County, MD, EUA*. (a) *Distance plot*. (b) Distribuição espacial dos dados.

Os valores de D_2 não são sensíveis à cardinalidade dos dados. Portanto, se a distribuição dos dados for mantida, o valor de D_2 não precisa ser recalculado após modificações no conjunto de dados. A dimensão fractal de correlação pode ser calculada utilizando-se apenas uma pequena amostra do conjunto de dados original, pois é invariante à amostragem imparcial (*unbiased sampling*).

2.7 Considerações Finais

Domínios de dados complexos, como dados multimídia ou sequências genéticas, não apresentam a propriedade de ROT. Domínios de dados que dispõem de uma função de distância métrica formam os espaços métricos. Para conjuntos de dados em espaços métricos somente existem os elementos e as distâncias entre eles. Para dados em espaços métricos, os operadores de comparação de ordem não são aplicáveis, sendo os operadores por similaridade os mais adequados para a manipulação desses dados.

Os principais operadores de seleção por similaridade são: consulta por abrangência, que seleciona todos os elementos do conjunto que estejam até uma dada distância do elemento de referência; e consulta aos k -vizinhos mais próximos, que seleciona os k elementos mais próximos do elemento de referência.

Em domínios de dados complexos, a comparação pode ser feita entre os próprios elementos do conjunto. Porém, comumente são comparados vetores de características extraídas dos elementos indexados. Em [Traina & Traina Jr., 2003] podem ser encontrados muitos dos conceitos apresentados neste capítulo, e um estudo de casos com indexação de imagens.

Para indexar dados em espaços métricos e utilizar os operadores por similaridade apresentados neste capítulo são necessárias estruturas de indexação mais complexas do que as tradicionalmente utilizadas para dados elementares, que serão discutidas no próximo capítulo.

Métodos de acesso métrico

3.1 Considerações Iniciais

As estruturas de indexação são dependentes dos domínios dos dados que se pretende manipular. Desde os métodos de acesso tradicionais aplicáveis em domínios de dados que apresentam ROT, muitos trabalhos foram realizados até o desenvolvimento dos Métodos de Acesso Métrico, para indexar dados em domínios que só dispõem dos próprios elementos e de uma função de distância entre eles. Esses são os métodos mais adequados para as consultas por similaridade em dados multimídia.

Neste capítulo é apresentado um levantamento bibliográfico do desenvolvimentos dos MAM.

3.2 Métodos de acesso métrico

Para dados que possuem a ROT foram desenvolvidos operadores de seleção que permitem executar operações de busca com complexidade sublinear para o número de elementos da base de dados, ou seja, existem algoritmos de busca com complexidade $O(N)$ ou menos, onde N é o número de elementos na base de dados. Esses algoritmos correspondem aos métodos de acesso tradicionais, tais como a amplamente utilizada *B-tree* [Johnson & Shasha, 1993] e suas variantes *B*-tree* e *B+-tree* [Zisman, 1993]. Por exemplo, a seleção de números ou textos curtos utilizando árvores de indexação pode

ser executada com algoritmos que exibem complexidade $O(\log n)$. Invariavelmente esses algoritmos utilizam a possibilidade de comparar os elementos dois a dois, e decidir pela verdade ou falsidade de determinada relação de ordem (menor que, igual, maior ou igual, etc.) entre cada par de elementos, o que permite evitar (podar) a comparação com ramos inteiros da árvore.

Uma outra classe de domínios interessantes são os chamados espaços multidimensionais, também chamados domínios em espaços dimensionais. Exemplos comuns desses espaços são: dados que representam informações geográficas [Güting, 1994]; a disposição espacial de estrelas e galáxias no espaço sideral; e conjuntos de características (“*features*”) extraídas de dados complexos, tais como imagens e dados de sensoreamento científico, que são utilizados para a realização de busca por conteúdo baseada nessas características [Faloutsos, 1996]. Em espaços multidimensionais, além da função de distância (em geral funções de distância do tipo L_p , tal como Euclidiana - L_2 , “*Manhattan*” - L_1 ou *Infinity* - L_0), são definidas relações de direção, permitindo a definição de ângulo entre triplas de elementos do conjunto, e a representação de hiper-volumes [Papadias et al., 1995]. A projeção de elementos em dimensões menores também é possível em espaços multidimensionais.

Esse conjunto (relativamente) grande de relações, chamadas de relações geométricas, permitem a construção de algoritmos eficientes para a indexação e suporte à execução de consultas em espaços multidimensionais. Para dados nesses espaços foram desenvolvidas diversas estruturas de indexação, conhecidas como “Métodos de Acesso Espacial - MAE” (ou *Spatial Access Methods - SAM*) a partir do trabalho pioneiro sobre as *R-trees* [Guttman, 1984] e suas derivadas *R*-tree* [Beckmann et al., 1990] e *R+-tree* [Sellis et al., 1987]. Um tutorial mostrando a evolução dos MAE pode ser encontrado em [Gaede & Günther, 1998], e um outro tutorial, mostrando seu uso em ambientes de altas dimensões é apresentado em [Böhm et al., 2001].

Para conjuntos de dados que não apresentam relações geométricas, mas nos quais somente existem os elementos e as distâncias entre eles, diversas pesquisas têm sido efetuadas, levando-se ao desenvolvimento dos “Métodos de Acesso Métrico - MAM”.

3.2 Métodos de acesso métrico

As técnicas de divisão de um espaço métrico propostas por Burkhard e Keller [Burkhard & Keller, 1973] foram o ponto de partida para o desenvolvimento desses métodos, apresentando as técnicas de particionamentos recursivos que permitem a construção de MAM. A primeira técnica proposta divide um conjunto de dados escolhendo um elemento como o representante do conjunto e agrupando os demais de acordo com suas distâncias para o representante. A segunda técnica divide o conjunto original em uma quantidade pré-determinada de subconjuntos e seleciona um representante para cada subconjunto. Cada representante e a maior distância dele para qualquer elemento do subconjunto são mantidas na estrutura, para facilitar as consultas por similaridade posteriores.

A árvore de indexação métrica proposta por Uhlmann [Uhlmann, 1991], referenciada como *Metric tree* e a *Vantage-Point tree* (*VP-tree*) proposta por Yianilos [Yianilos, 1993] são exemplos de MAM baseados na primeira técnica proposta em [Burkhard & Keller, 1973]. Em ambos os trabalhos, uma árvore binária é construída recursivamente. Em cada nível, é escolhido um elemento para ser o centro de uma “esfera métrica”, e então define-se um raio para essa esfera, de forma que metade dos elementos indexados estejam dentro desta esfera e metade fora. Esses dois grupos de elementos são colocados nos ramos esquerdos e direitos da árvore, respectivamente. Em cada nó tem-se um representante, chamado em [Yianilos, 1993] de *Vantage-Point*.

Em [Baeza-Yates et al., 1994] foi apresentada a *FQ-tree* (*Fixed Queries tree*), outro exemplo de MAM baseado na primeira técnica proposta em [Burkhard & Keller, 1973]. A diferença principal com relação à *VP-tree* é que apenas um representante é usado para todos os nós no mesmo nível da árvore.

A *MVP-tree* (*Multi-Vantage-Point tree*) [Bozkaya & Özsoyoglu, 1997, Bozkaya & Özsoyoglu, 1999] é uma variante da *VP-tree* em que são utilizados mais de um representante (*Vantage-Point*) para cada nó. Outra característica da *MVP-tree* é o armazenamento das distâncias já computadas entre os elementos nos nós-folha e seus ancestrais, juntamente com os elementos nas folhas.

Outros trabalhos posteriores que foram baseados na *VP-tree* e suas

variantes podem ser encontrados em [Yianilos, 1999],[Gennaro et al., 2001] [Fu et al., 2000],[Dohnal et al., 2003] e [Sahinalp et al.,].

A *GH-tree*(*Generalized Hyperplane Decomposition tree*), também proposta por Uhlmann [Uhlmann, 1991], é uma estrutura de indexação baseada na segunda técnica proposta em [Burkhard & Keller, 1973]. A *GH-tree* particiona recursivamente o conjunto de dados em dois, selecionando dois representantes e associando os elementos restantes com o representante mais próximo. Desta forma, no ramo esquerdo da árvore ficam os elementos que estão mais próximos ao primeiro representante do que ao segundo (ou equidistantes), e no ramo direito os elementos que estão mais próximos ao segundo representante do que ao primeiro [Hjaltason & Samet, 2003].

A *GNAT* (*Geometric Near-neighbor Access tree*) [Brin, 1995] é uma extensão da *GH-tree*, permitindo a escolha de mais do que dois representantes por nó. Além disso, durante a construção da estrutura, para cada representante são armazenados também as distâncias mínimas e máximas para elementos de todos os outros representantes, visando aumentar a capacidade de poda pela desigualdade triangular durante as buscas.

Os principais MAM apresentados neste capítulo até aqui [Uhlmann, 1991][Yianilos, 1993][Baeza-Yates et al., 1994] [Bozkaya & Özsoyoglu, 1997, Bozkaya & Özsoyoglu, 1999][Brin, 1995] constroem a estrutura de indexação utilizando todo o conjunto de dados disponível numa única operação, e não permitem operações posteriores de inserção e remoção de elementos, sendo por isso denominados métodos estáticos.

A *M-tree* [Ciaccia et al., 1997] foi o primeiro MAM dinâmico, apresentado na literatura em 1997, sendo baseado na segunda técnica de [Burkhard & Keller, 1973], com estrutura similar à *R-tree* [Guttmann, 1984]. Trata-se de uma estrutura balanceada pela altura que armazena os nós em registros de disco de tamanho fixo.

Em Ciaccia2002a é proposta uma extensão da *M-tree*, a *QIC-M-tree*, que possibilita a utilização de várias funções de distância. O MAM *M⁺-tree* [Zhou et al., 2003] é outra extensão da *M-tree* que também tira proveito das idéias dos vários representantes por nó da *MVP-tree* [Bozkaya & Özsoyoglu, 1997, Bozkaya & Özsoyoglu, 1999]. Os nós da árvore

são divididos em “nós gêmeos” sem sobreposição entre si, aumentando a capacidade de poda nas operações de busca. A *BM⁺-tree* [Zhou et al., 2005] é uma variação da *M⁺-tree* que utiliza outro mecanismo para o particionamento do espaço dos nós.

Na *MB⁺-tree*[Ishikawa et al., 2000], os elementos são armazenados em *B⁺-trees* e é utilizada uma estrutura adicional com informações de particionamento do espaço para auxiliar nas operações sobre os dados indexados.

A *M*-tree* [Skopal & Hoksza, 2007] é outra extensão da *M-tree* que tem associada a cada nó uma estrutura para armazenar referências para o vizinhos mais próximo de cada entrada do nó, visando aumentar a capacidade de poda da estrutura. Em [Skopal & Lokoč, 2009] são propostas duas novas técnicas de inserção de dados na *M-tree*, ambas com o objetivo de produzir estruturas mais compactas.

A *Slim-tree* [Traina Jr. et al., 2000b, Traina Jr. et al., 2002a] aperfeiçoou a *M-tree*, estabelecendo a primeira técnica operando em espaços métricos capaz de medir e reduzir a sobreposição entre subárvores. A sobreposição de nós é um efeito indesejável, pois obriga a busca em profundidade em diversas subárvores para a localização dos elementos solicitados pelas consultas [Traina Jr. et al., 2002a]. A *Slim-tree* (detalhada na Seção 3.3) também é uma estrutura de indexação dinâmica, e possui o algoritmo *Slim-Down*, que reorganiza a estrutura para minimizar a sobreposição entre as subárvores em cada nó.

Um novo algoritmo de particionamento (divisão) para o caso em que os nós têm sua capacidade máxima excedida foi proposto com a *Slim-tree*, chamado *Minimum Spanning Tree* (MST). Outro algoritmo de particionamento para a *M-tree* foi proposto em [Lim et al., 2006], visando diminuir a sobreposição entre os nós. Porém esse novo algoritmo de particionamento utiliza pontos centrais virtuais (objetos artificiais), restringindo sua utilização para espaços multidimensionais.

Um desenvolvimento posterior foi a proposta de utilização de múltiplos representantes chamados “*omni-focos*” para atuar como geradores de coordenadas globais para todos os elementos de um conjunto [Santos Filho et al., 2001, Traina et al., 2007]. Essa abordagem foi a base para a criação de uma família de MAM, entre eles o *Omni-Sequential*, o *Omni-R-tree*, e o *Omni-B-Forest* [Traina et al., 2007]. Ela foi utilizada juntamente com a *Slim-tree*,

resultando no MAM *DF-tree* [Traina Jr. et al., 2002b].

Em [Vieira et al., 2004] foi proposta a *DBM-tree*, na qual é possível diminuir a sobreposição entre os nós através da flexibilização do balanceamento da estrutura.

A *Antipole tree* [Cantone et al., 2005], que segundo os autores combina as idéias da *FQ-tree*, *M-tree* e *MVP-tree*, procura pelos elementos mais distantes entre si (*Antipole pair*), utilizando-os para particionar o conjunto de dados. Trata-se de uma árvore binária que armazena os elementos indexados agrupados nas folhas, que também permite consultas aproximadas.

Existem alguns trabalhos interessantes que apresentam extensas revisões bibliográficas sobre MAM, como [Hjaltason & Samet, 2003] e [Chávez et al., 2001].

Com exceção da *DBM-tree* [Vieira et al., 2004], nenhum MAM publicado até agora permite a remoção completa de elementos já inseridos. Mesmo assim, a *DBM-tree* o faz abrindo mão do balanceamento da estrutura. Embora seja concebível que elementos armazenados apenas nas folhas, que não tenham sido usados como representantes em nós acima das folhas, possam ser removidos, os elementos que porventura estejam sendo utilizados como representantes não podem ser removidos, a não ser que a estrutura tenha as subárvores abaixo do elemento removido completamente refeitas, o que pode acarretar a recriação de toda a árvore, quando elementos presentes na raiz são removidos. Assim, em todos eles, sugere-se que a operação de remoção de elementos utilizados como representantes seja feita apenas marcando-se os mesmos como removidos, sem eliminá-los de fato. Essa alternativa é aceitável quando existem poucas operações de remoção de elementos. No entanto, quando a quantidade de operações de remoção não é pequena (e isso ocorre particularmente quando os elementos de dados podem evoluir no tempo), essa alternativa torna-se ruim, pois além de aumentar o consumo de memória em disco e consequentemente a quantidade de acessos que devem ser feitos aos discos, ela aumenta a quantidade de cálculos de distância, que passam a ser feitos com elementos que não existem mais no conjunto de dados. Como parte deste trabalho de doutorado, foi desenvolvido um algoritmo de remoção efetiva de dados, implementado sobre o MAM *Slim-tree* Seção 5.2. Com isso, a *Slim-tree* passa também a permitir a remoção efetiva de qualquer elemento

indexado, e não somente a marcação dos elementos removidos.

3.3 *Slim-tree*

A *Slim-tree* [Traina Jr. et al., 2000b] [Traina Jr. et al., 2002a] é uma estrutura balanceada e dinâmica, que tem crescimento *bottom-up* (das folhas para a raiz) e permite inserções de dados de um domínio métrico.

Na *Slim-tree* pode ocorrer sobreposição das áreas de cobertura de nós no mesmo nível da árvore. Ou seja, a divisão do espaço métrico feita pelos nós no mesmo nível da árvore não gera regiões necessariamente disjuntas. A *Slim-tree* dispõe de um algoritmo chamado *Slim-Down*, utilizado posteriormente à construção da árvore, para minimizar o problema da sobreposição entre os nós. A estrutura também tem a capacidade de avaliar o grau de sobreposição de seus nós (*fat-factor*).

3.3.1 *Organização da Slim-tree*

Em uma *Slim-tree*, os elementos são agrupados no disco em páginas de tamanho fixo, cada página correspondendo a um nó da árvore. Os elementos são armazenados nas folhas, organizados numa estrutura hierárquica que utiliza um elemento representante como centro de uma região de cobertura dos elementos em uma subárvore, delimitada por um raio máximo de cobertura.

Existem dois tipos de nós na *Slim-tree*, os nós-folha (*data-nodes* ou *leaf-nodes*) e os nós-índice (*index-nodes*). Cada página armazena um número máximo de elementos. A estrutura de um nó-folha, onde são armazenados todos os elementos da *Slim-tree*, é a seguinte:

$$\text{nó-folha} \left[\text{vetor de } <OID_i, d(s_i, s_{rep}), s_i>\right]$$

onde OID_i é o identificador do elemento s_i e $d(s_i, s_{rep})$ é a distância entre o elemento s_i e s_{rep} , que é o representante desse *nó-folha*.

A estrutura de um nó-índice é:

$$\text{nó-índice} \left[\text{vetor de } <s_i, r_i, d(s_i, s_{rep}), \text{Ptr}(Ts_i), \#\text{Ent}(\text{Ptr}(Ts_i))>\right]$$

onde s_i armazena o elemento representante da subárvore apontada por $Ptr(Ts_i)$, r_i é o raio de cobertura da região, $d(s_i, s_{rep})$ é a distância entre s_i e o representante deste nó, e $\#Ent(Ptr(Ts_i))$ armazena o número de entradas na subárvore apontada por $Ptr(Ts_i)$.

Na Figura 3.1, tem-se uma representação gráfica da estrutura lógica de uma *Slim-tree*, com a distinção entre nós-índice e nós-folha, e na Figura 3.2 um exemplo de sua utilização para a indexação de um conjunto de sete palavras, utilizando a função de distância *LEdit*.

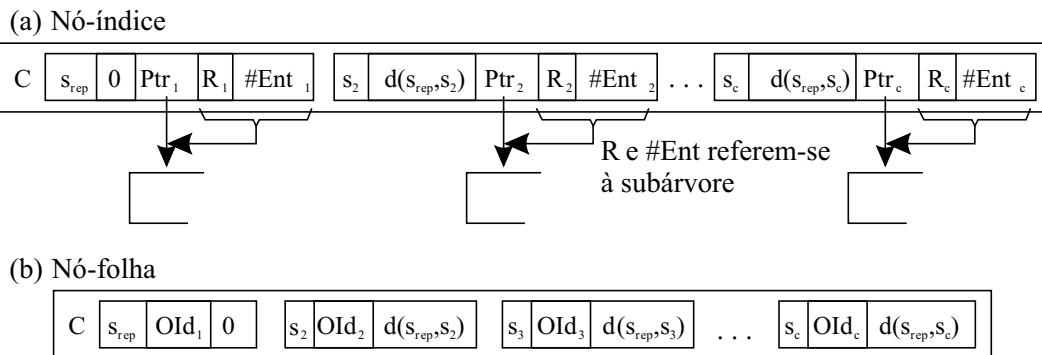


Figura 3.1: Representação gráfica da estrutura lógica de nós-índice e nós-folha da *Slim-tree*. [Traina Jr. et al., 2002a]

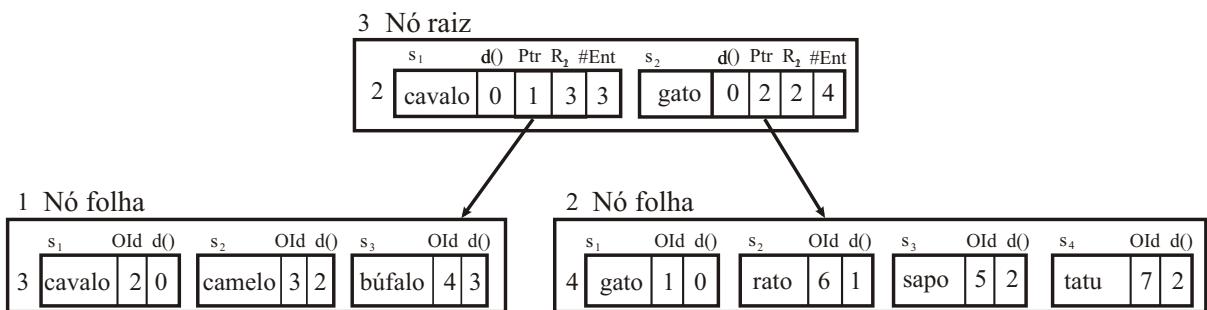


Figura 3.2: Exemplo de indexação de sete palavras usando a função de distância *LEdit*.

A Figura 3.3(a) apresenta a disposição de 17 elementos (s_1, \dots, s_{17}) indexados por uma *Slim-tree* com capacidade máxima de 3 elementos por nó.

Os círculos brancos da Figura 3.3(b) representam nós-folha, e o de cor cinza os nós-índice. Os elementos representantes de cada nó são mostrados em preto, e os demais elementos indexados são representados por pontos cinzas. Pode-se ver na figura várias sobreposições entre os nós, indicadas pelas intersecções entre as áreas de cobertura. Neste exemplo, o elemento s_{13} está localizado na região de intersecção de dois nós-folha, com

representantes s_6 e s_8 . Portanto, o elemento s_{13} poderia estar inserido em qualquer um dos dois nós-folha.

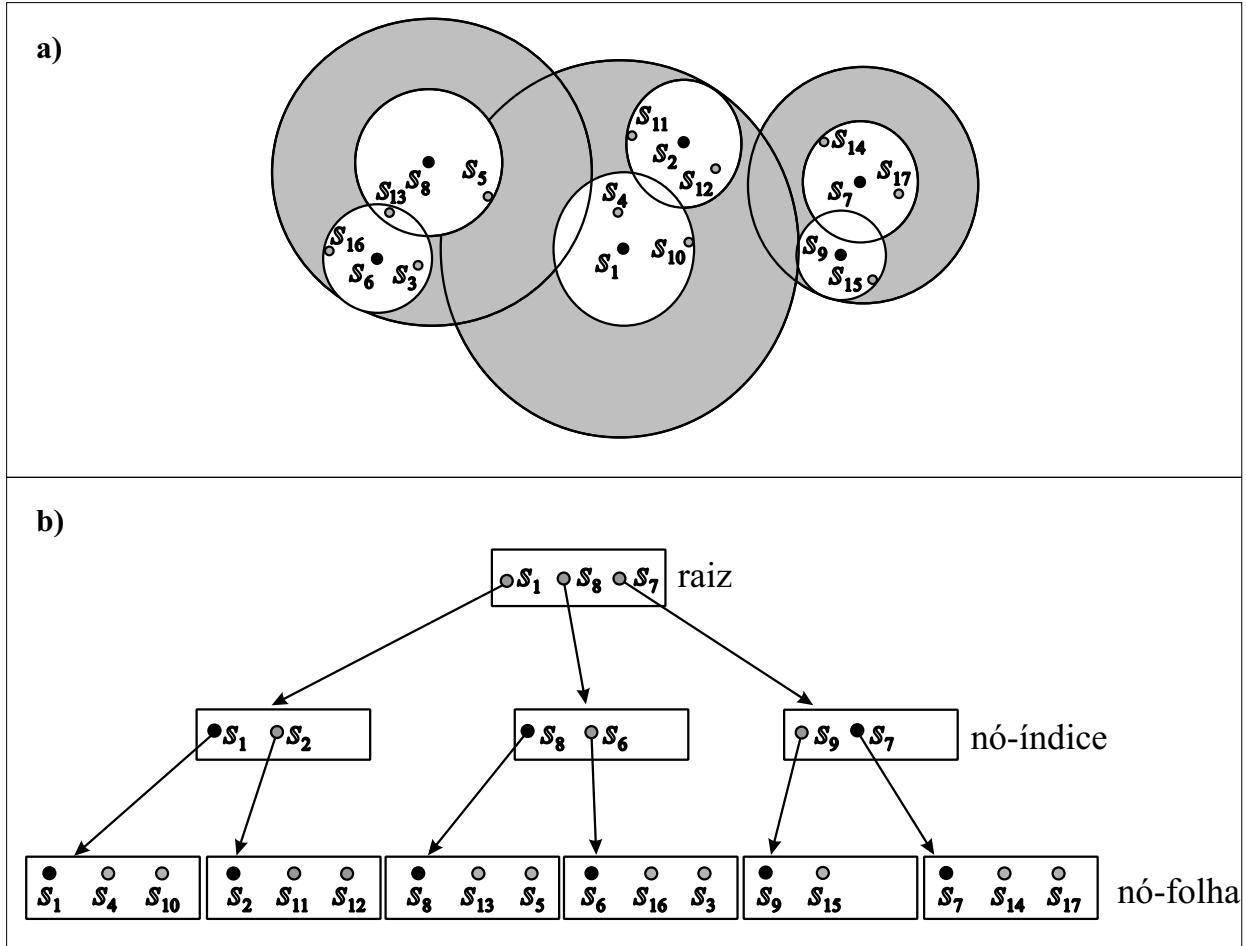


Figura 3.3: Representação de uma *Slim-tree* em (a) e sua estrutura lógica em (b).

3.3.2 Construção de uma Slim-tree

Na inserção de um novo elemento em uma *Slim-tree*, começando pela raiz, o algoritmo tenta encontrar um nó em que o raio de cobertura cubra o novo elemento. Se nenhum nó for qualificado, escolhe-se o nó que tenha o elemento representante mais próximo do novo elemento, aumentando-se o raio desse nó para que cubra o novo elemento. Se dois ou mais nós forem qualificados, é executado um algoritmo para escolher qual é o nó onde será inserido o novo elemento. Existem três opções para a escolha do nó onde será inserido o elemento:

Aleatória (*Random*): o nó é escolhido aleatoriamente entre aqueles que foram

qualificados para inserção sem o aumento do raio de cobertura;

Mínima Distância *MinDist*: é escolhido o nó cuja distância entre seu elemento representante e o novo elemento seja a menor, dentre os nós qualificados.

Mínima Ocupação (*MinOccup*): seleciona-se dentre os nós previamente qualificados aquele que apresenta o menor número de elementos já armazenados.

Esse processo é sucessivamente aplicado para cada nível da *Slim-tree*.

Na ocorrência de uma nova inserção, pode ocorrer do nó escolhido ter sua capacidade máxima excedida. Quando isso ocorre, um novo nó é alocado no mesmo nível. A *Slim-tree* possui três opções de algoritmos para a escolha de quais serão os elementos representantes dos novos nós, quando ocorre a divisão de nós (*splitting*):

Random: são escolhidos aleatoriamente dois novos representantes para os dois nós;

Minmax: todos os possíveis pares de elementos são considerados potenciais representantes. Serão escolhidos como representantes os dois elementos que minimizarem o raio de cobertura.

MST (*Minimal Spanning Tree*): é criada uma árvore de menor caminho com os elementos e um dos arcos mais longos é removido, tendo-se então dois agrupamentos.

Depois de escolhidos os representantes, os elementos restantes são distribuídos pelos 2 nós, alocando cada elemento no nó cujo representante está mais próximo do respectivo elemento. Os elementos escolhidos como representantes dos dois nós depois da divisão devem ser inseridos no nó pai, o que pode gerar a necessidade de nova divisão. O processo de divisão, se necessário, pode ser repetido nos níveis superiores, aumentando a altura da árvore quando propagado até o nó raiz. As informações referentes ao número de elementos nas subárvore e raios de cobertura devem ser atualizados nos nós dos níveis superiores.

Em [Vespa et al., 2007] foram propostas técnicas de carga rápida (*bulk-loading*) baseadas em amostragem para MAM hierárquicos e dinâmicos. Os algoritmos foram implementados e avaliados com o MAM *Slim-tree*.

3.3.3 Consultas por similaridade na *Slim-tree*

Na *Slim-tree*, assim como outros MAM, pode ocorrer a sobreposição da área de cobertura de nós distintos. A *Slim-tree* utiliza a desigualdade triangular para “poder” cálculos de distâncias.

Como as distâncias entre todos os elementos armazenados em um nó e o representante de tal nó são conhecidas, a propriedade de desigualdade triangular pode ser usada durante uma consulta por similaridade para descartar elementos que, não fazem parte do conjunto resposta. A “poda” pode ser feita sem a necessidade de calcular as distâncias entre cada elemento armazenado e o elemento de referência para consulta, reduzindo assim o número de cálculos de distâncias necessários para a realização da consulta.

Dado o espaço métrico $\langle \mathbb{S}, d \rangle$ o conjunto de elementos $S \subseteq \mathbb{S}$, o elemento de consulta $s_q \in \mathbb{S}$, o raio de consulta r_q e um elemento representante $s_{rep} \in S$, um elemento $s_i \in S$ poderá ser descartado pela propriedade de desigualdade triangular se uma das duas condições a seguir forem satisfeitas:

$$d(s_{rep}, s_i) < d(s_{rep}, s_q) - r_q \quad (3.1)$$

$$d(s_{rep}, s_i) > d(s_{rep}, s_q) + r_q \quad (3.2)$$

Um exemplo da utilização da desigualdade triangular para podar cálculos de distância na árvore é mostrado na Figura 3.4, em que s_{rep} é o elemento representante do nó da árvore, s_q é o elemento de referência para busca e r_q é o raio da consulta. Os elementos s_1, s_2, s_3 e s_4 pertencem ao conjunto de dados em domínios métricos $S \subset \mathbb{S}$. Os elementos s_2 e s_3 podem ser descartados pela propriedade de desigualdade triangular; o elemento s_1 não pode ser descartado, mas não faz parte do conjunto resposta; e s_4 não pode ser descartado e faz parte do conjunto resposta.

Na Figura 3.4, a região *A* equivale à condição 3.1 e a região *C* à condição 3.2. Como a distância de qualquer elemento ao seu representante é armazenada na árvore, qualquer

elemento que estiver nas regiões A ou C pode ser descartado através da propriedade de desigualdade triangular, sem calcular sua distância com o elemento da consulta. Já na região B, há necessidade de se calcular a distância entre o elemento de consulta e os elementos pertencentes à região, para verificar se esses elementos devem ser incluídos como resultados da consulta.

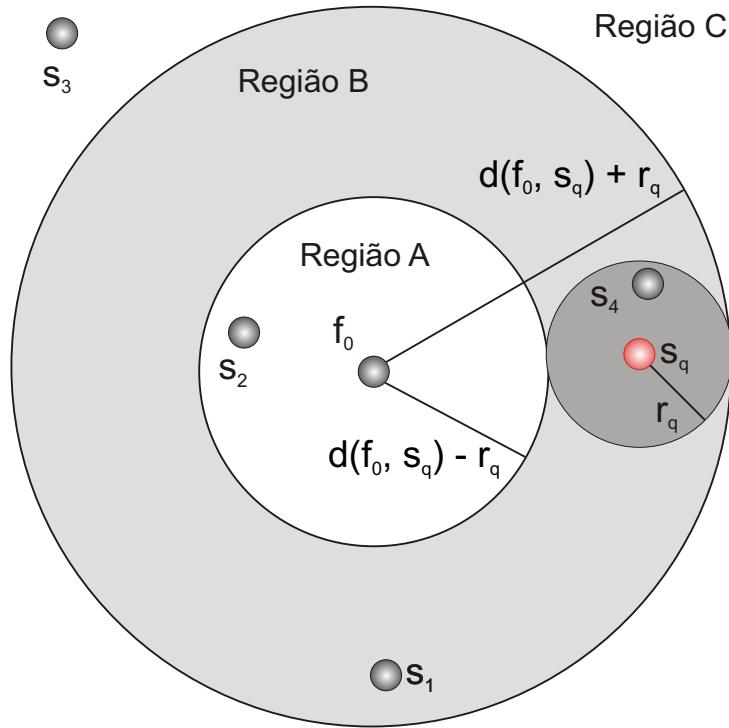


Figura 3.4: Descarte de elementos com o uso da desigualdade triangular.

Na execução do algoritmo de consulta por abrangência, iniciando pelo nó raiz, percorre-se a árvore e calcula-se a distância do elemento s_q com os elementos armazenados no conjunto de dados que não podem ser podados pela propriedade de desigualdade triangular. Inclui-se no conjunto resposta todos aqueles que estão a uma distância inferior ou igual a r_q .

Já na consulta por vizinhos mais próximos é utilizada uma lista global de páginas válidas, que armazena ponteiros para as páginas (subárvores) que podem conter elementos qualificados para a resposta, mas ainda não percorridas. O raio da consulta é dinâmico, podendo ser considerado inicialmente como um valor infinito.

A ordem em que os nós da árvore são acessados pode influenciar no desempenho da

consulta por vizinhos mais próximos. Visando aumentar a quantidade de subárvores que podem ser podadas, a próxima página escolhida para ser percorrida é aquela que tem seu elemento representante mais próximo do elemento de consulta, na lista global de páginas válidas.

Quando são encontrados elementos melhores que aqueles presentes no vetor de respostas, esses elementos são inseridos nesse vetor de maneira a mantê-lo ordenado. O critério utilizado para “podar” subárvores é dinâmico, sendo baseado na distância entre o elemento de consulta e o elemento mais distante armazenado no vetor de respostas.

3.3.4 Sobreposição em árvores métricas

A divisão dos espaços métricos de quase todos os MAM dinâmicos não garante regiões disjuntas, produzindo sobreposição entre os nós de um mesmo nível da árvore, que reduz a capacidade de podar subárvores.

Em [Traina Jr. et al., 2002a], juntamente com o MAM *Slim-tree*, foi proposto o *fat-factor* para avaliar o grau de sobreposição entre os nós da estrutura. A sobreposição entre dois nós foi definida como o número de elementos em suas subárvores que são cobertos por ambos os nós, dividido pelo número total de elementos nas subárvores.

O *fat-factor absoluto* ($Fat(T)$) de uma *Slim-tree* T com altura H , armazenando N elementos em M páginas de disco é dado por:

$$Fat(T) = \frac{I_C - H * N}{N} * \frac{1}{(M - H)} \quad (3.3)$$

onde I_C representa o total de nós acessados necessários para responder consultas pontuais para cada um dos N elementos armazenados na árvore métrica.

Os valores do *fat-factor absoluto* de uma estrutura podem variar no intervalo $[0, 1]$, e valores maiores representam estruturas com maior grau de sobreposição, e $Fat(T) = 0$ indica uma árvore ideal onde não existe sobreposição entre os nós.

O *fat-factor absoluto* é uma medida da quantidade de elementos que ocupam regiões de intersecção de nós em um mesmo nível de um MAM. Porém, se duas árvores armazenando o mesmo conjunto de dados têm diferentes números de nós, a comparação direta dos

valores de *fat-factor absoluto* não pode ser feita. Para permitir a comparação das duas árvores diferentes que armazenam o mesmo conjunto de dados, foi proposto o *fat-factor relativo*, que “penaliza” árvores que usam mais do que o número mínimo de nós necessários para armazenar os elementos do conjunto de dados. No cálculo do *fat-factor relativo* são considerados os número de nós e a altura da árvore ideal, e não da árvore real.

$$rFat(T) = \frac{I_C - H_{min} * N}{N} * \frac{1}{(M_{min} - H_{min})} \quad (3.4)$$

onde $H_{min} = \lceil \log_2 N \rceil$ é a altura mínima da árvore, e o número mínimo de nós para um conjunto de dados é dado por $M_{min} = \sum_{i=1}^{H_{min}} \lceil N/C^i \rceil$, onde C é a capacidade dos nós.

O valor de $rFat(T)$ varia entre 0 e um número real positivo, sendo que quanto menor o valor, menor o número de acessos a disco necessários para responder uma consulta.

Em [Traina Jr. et al., 2002a] também foi proposta uma técnica para minimizar a sobreposição entre nós em árvores métricas, chamado *Slim-down*. Esse algoritmo é executado sobre a árvore já construída. Os valores do *fat-factor* podem ser monitorados para indicar a necessidade do processo de otimização.

Quando existe sobreposição entre nós-folha, o *Slim-down* realiza a “migração” do elemento mais distante do representante do nó para um nó irmão que também já cubra o elemento. Com essa migração, o raio de cobertura do nó que “exporta” o elemento pode ser reduzido, sem que seja necessário aumentar o raio de cobertura do nó que recebe o elemento. Com isso, a sobreposição entre os nós tende a diminuir. Este procedimento é repetido até que não existam mais migrações de elementos entre os nós irmãos. Um exemplo da aplicação do método *Slim-down* pode ser visto na Figura 3.5.

O *Slim-down* realiza a redução dos raios de cobertura comparando apenas os nós-folha irmãos, ou seja, aqueles que estão ligados a um mesmo nó-índice. Assim, nenhuma operação é realizada no caso de sobreposição de dois nós-folhas que não sejam irmãos. Em [Skopal et al., 2003] foi proposto o algoritmo *Generalized Slim-down*, que percorre a árvore otimizando os nós, não se restringindo apenas aos nós-folha. Procura-se uma melhor localização para cada uma das entradas dos nós percorridos, redistribuindo-as no mesmo nível da árvore. Porém, como pode-se inferir pelos experimentos do autor, essa

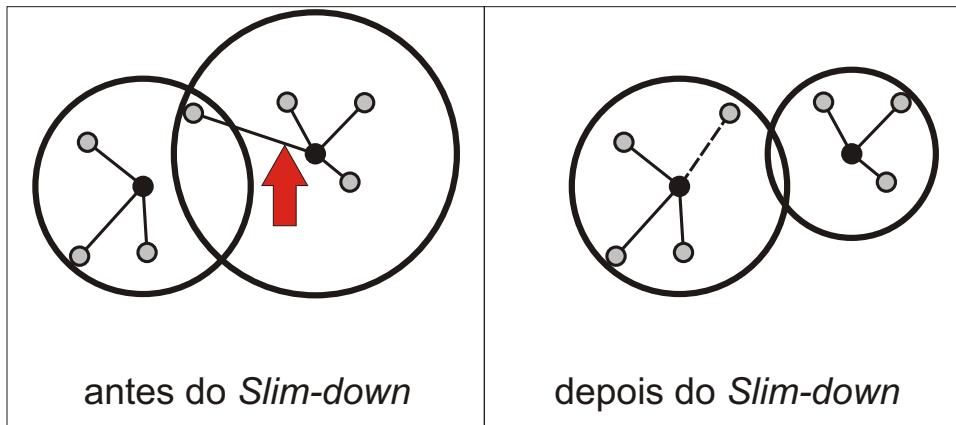


Figura 3.5: Exemplos de otimização realizada pelo método *Slim-down*.

otimização chega a ser até duzentas vezes mais custosa do que a construção inicial da árvore.

Na Seção 5.3 é apresentada uma nova técnica de otimização [Bueno et al., 2008a] que permite a movimentação de elementos entre subárvores, sendo menos custoso que o *Generalized Slim-down*.

3.4 Considerações Finais

Neste capítulo foi apresentado um levantamento bibliográfico do desenvolvimentos dos MAM, apresentando com mais detalhes alguns trabalhos de maior interesse para este doutorado. Muitas estruturas de indexação foram propostas com o objetivo de agilizar as buscas por similaridade, especialmente em domínios métricos.

Nos MAM, normalmente o espaço de busca é partitionado de acordo com técnicas específicas de partitionamento, e são criadas estruturas hierárquicas para o armazenamento e manipulação dos elementos, geralmente árvores.

As primeiras estruturas propostas eram estáticas, sendo a *M-Tree* [Ciaccia et al., 1997] o primeiro MAM dinâmico proposto na literatura. No presente capítulo foi apresentada mais detalhadamente a *Slim-tree* [Traina Jr. et al., 2000b, Traina Jr. et al., 2002a], que foi base para a implementação dos algoritmos de remoção, atualização e otimização desenvolvidos durante este doutorado, apresentados no Capítulo 5.

No próximo capítulo são apresentados alguns conceitos e trabalhos sobre o tratamento de informações temporais em bases de dados.

Capítulo
4

Tempo em Bases de Dados

4.1 Considerações Iniciais

A maioria das aplicações às quais os SGBD atualmente devem dar suporte manipulam, de alguma maneira, dados com características de tempo, seja para informações históricas, atuais ou futuras.

De uma forma geral, o termo ‘bases de dados temporais’ engloba toda base de dados que utiliza algum aspecto de tempo na organização de dados [Elmasri & Navathe, 2006]. As Bases de dados temporais permitem o armazenamento de dados do passado, presente e futuro da aplicação, mantendo registrada a sua evolução temporal [Edelweiss, 1998, Tansel et al., 1993].

Em bases de dados temporais, a representação do tempo pode ser discreta ou contínua. No caso de representação contínua, especialmente em espaços de baixa dimensionalidade, os objetos são frequentemente chamados de “objetos móveis”.

Na seção 4.2 são apresentados os conceitos básicos no desenvolvimento de bases de dados temporais. Em seguida, na seção 4.3 são discutidos objetos móveis, que podem ser vistos como uma especialização de dados espaço-temporais.

4.2 Dados temporais

O suporte à ideia de evolução temporal em bases de dados apresenta diversos aspectos conceituais no que se refere à maneira como o tempo deve ser tratado. Dois dos aspectos mais importantes são a maneira como o tempo deve ser modelado e a possibilidade de se registrar o tempo sob diversas dimensões. Esses aspectos afetam tanto a maneira como os dados devem ser modelados e armazenados, quanto os aspectos de representação de consultas envolvendo tempo, que devem expressar a semântica das consultas.

Apesar de o tempo ser naturalmente contínuo, sua visão em bases de dados temporais pode ser discreta ou contínua, sendo a interpretação discreta mais comumente adotada em bases de dados temporais devido a simplicidade e facilidade de implementação [Tansel et al., 1993]. Na interpretação discreta, o tempo é visto como uma sequência de intervalos consecutivos. Esses intervalos são chamados de *Chronon* e são a menor porção de tempo representada (que não pode ser decomposta) e é dependente da aplicação.

Um fator importante a ser analisado na representação de dados temporais refere-se a ordem do tempo. A forma mais comumente adotada é a ordem linear. Dessa forma, dois pontos no tempo podem sempre ser ordenados. Outro exemplo de ordem que pode ser adotada na representação do tempo trata-se da ordem circular, que é utilizada quando pretende-se representar períodos de tempo que se repetem periodicamente, como por exemplo meses do ano, ou dias da semana. Outra diferenciação que pode ser feita nos modelos de representação de tempo refere-se à maneira como o tempo é representado, seja na forma de instantes de tempo ou intervalos de tempo associados aos dados.

A associação de tempo aos dados de uma base pode ser interpretada de diferentes maneiras. O ‘tempo da transação’ registra o momento em que a informação é manipulada pelo SGBD. Já o ‘tempo válido’ representa o período em que a informação é considerada correta no domínio da aplicação. Essas duas interpretações de tempo são as mais comuns, e são chamadas de dimensões de tempo [Elmasri & Navathe, 2006].

As bases de dados podem ser classificadas pela forma que as dimensões de tempo são utilizadas [Edelweiss, 1998]. Vale ressaltar aqui que muitos dos termos e classificações referentes a bases de dados temporais foram atualizados e modificados no decorrer do

tempo [Jensen et al., 1997].

As bases de dados chamadas de ‘bases de dados instantâneas’ não utilizam nenhuma dimensão de tempo, armazenando apenas os dados correntes. Nas ‘bases de dados de tempo de transação’ (anteriormente chamadas de ‘rollback’), os dados são rotulados apenas com seus tempos de transação. Quando algum dado da base é alterado, o valor anterior não é destruído, sendo armazenados todos os estados passados. As ‘bases de dados de tempo válido’ (anteriormente chamadas de ‘históricas’) armazenam somente o tempo válido juntamente com os dados, não se tendo associada ao dado a informação do tempo de transação.

A partir da Linguagem TSQL2 [Snodgrass, 1995], introduzida como uma extensão ao padrão SQL-92, passou a ser aceita a representação do tempo em um modelo conceitual Bi-temporal (‘base de dados bi-temporais’), que incorpora tanto a representação do ‘tempo da transação’ quanto a representação do ‘tempo válido’. Além dessas duas dimensões de tempo, uma dimensão de tempo adicional pode ser definida caso a aplicação necessite, chamada de ‘tempo definido pelo usuário’. Em uma base de dados bi-temporal existe a possibilidade do acesso a todos os estados passados da base, seja pelo histórico de transações ou de validade, e valores futuros podem ser obtidos pelo tempo de validade [Snodgrass, 1995].

Em TSQL2 o tempo sempre é representado através de períodos de tempo com início e fim, em oposição à representação de instantes de tempo. Nessa linguagem não existe a necessidade de se representar o tempo explicitamente, em alguma coluna ‘tempo’ nas relações, pois o suporte a tempo é estabelecido pelo próprio gerenciador em todas as relações declaradas como dependentes do tempo [Chen & Zaniolo, 1999]. Dessa maneira, um SGBD pode utilizar estruturas apropriadas ao gerenciamento dos dados considerando sua variação temporal, incluindo métodos de acesso que considerem a evolução dos dados indexados evitando, na maioria das situações, a necessidade de realizar a cara operação de intersecção de resultados intermediários para conhecer dados oriundos de instantes de tempo diversos [Sellis, 1999]. Em [Manica et al., 2009b, Manica et al., 2009a] foi apresentada uma ferramenta que possibilita a execução de consultas na linguagem

TSQL2 sobre SGBD convencionais.

Diversas estruturas de indexação têm sido criadas para permitir representar e recuperar dados com vínculo temporal. Um levantamento interessante das estruturas de indexação para dados espaço-temporais pode ser encontrado em [Mokbel et al., 2003], classificadas de acordo com o tipo e o tempo (passado, corrente e futuro) da consulta. Isso inclui a necessidade de indexar tanto a dimensão tempo de transação quanto o tempo de validade, o que é exemplificado pelo método *BT-tree*, que permite a navegação alternando as dimensões temporais [Jiang et al., 2000].

O gerenciamento eficiente de conjuntos de intervalos de tempo é um requisito fundamental para essas estruturas. Uma estrutura voltada especialmente para tratar esse problema é a *RI-tree* [Kriegel et al., 2004, Kriegel et al., 2002], que processa a interseção dos resultados de consultas, codificados como sequências de intervalos representados pelas tuplas de uma relação em um SGBD objeto-relacional [Kriegel et al., 2000]. A *RI-tree* é interessante especialmente para aplicações que trabalham com dados espaço-temporais, uma vez que os intervalos podem ser tanto temporais quanto faixas de valores para as dimensões espaciais.

Outro ponto importante é que a taxa de atualização nas estruturas de indexação para dados que evoluem pode ser alta. Visando tratar desse problema, em [Kwon et al., 2002] é apresentada a *LUR-tree*, baseada na *R-tree*. Essa técnica atualiza a estrutura de indexação somente se um objeto se move para fora de seu MBR (*minimum bounding rectangle*) original. Se a nova posição do objeto ainda é coberta por seu MBR, apenas é alterada sua posição no nó-folha. Com isso a operação de atualização torna-se muito mais rápida. Em [Lee et al., 2003] é apresentada uma estratégia de atualização *bottom-up* para *R-trees*. Um índice compacto que permite o acesso direto aos nós-índice da *R-Tree* é mantido em memória, o qual é usado em conjunto com algoritmos de navegação *bottom-up*. Em [Xiong & Aref, 2006, Silva et al., 2009b] foi proposta a *RUM-tree*, que utiliza anotações para distinguir dados atuais dos obsoletos (não removidos imediatamente da estrutura, mas marcados para remoção posterior), visando agilizar o processo de atualização.

O uso de estruturas multidimensionais para consultas a dados que variam com o tempo,

além dos problemas oriundos da maldição da dimensionalidade, têm um complicador no fato de que dados em evolução freqüentemente não tem seu extremo superior do limite de validade definido. Com isso, todos os objetos têm uma intersecção no infinito, todos compartilhando um ponto comum na dimensão tempo, o que causa degradação adicional nos métodos de busca [Kollios et al., 2005].

Outros trabalhos voltados para o modelo relacional podem ser encontrados em [Clifford & Croker, 1987][Navathe & Ahmed, 1989][Gadia & Yeung, 1988]. Em [Skjellaug, 1997] é apresentada uma revisão bibliográfica sobre extensões do modelo relacional.

Em [Faria et al., 1998] é apresentado um *framework* para extensão de um sistema gerenciador de bases de dados orientado a objetos, incluindo o arcabouço necessário para o desenvolvimento de aplicações espaço-temporais. O modelo orientado a objetos TF-ORM (*Temporal Functionality in Objects With Role Model*)[Edelweiss et al., 1993, de Oliveira et al., 1995] utiliza o conceito de papéis para representação dos comportamentos dos objetos [Edelweiss, 1998], e suporta tanto o tempo de transação como de validade. Outros trabalhos também voltado para modelos orientados a objetos podem ser encontrados em [Rose & Segev, 1991][Elmasri et al., 1993][Bertino et al., 1996][Fauvet et al., 1997].

Em [Skjellaug, 1997] é apresentada uma revisão bibliográfica sobre extensões do modelo de bases de dados orientadas a objetos. Outros trabalhos que trazem revisões bibliográficas sobre modelos de dados temporais são apresentados em [Ozsoyoglu & Snodgrass, 1995] e [Theodoulidis et al., 1996].

4.3 Objetos móveis

SGBD para dados espaço-temporais tratam de mudanças geométricas nos dados indexados no decorrer do tempo, que podem ser discretas ou contínuas. Nesse último caso, especialmente em espaços de baixa dimensionalidade (usualmente com duas ou três dimensões) os objetos são frequentemente chamados de “objetos móveis”. Um banco de dados de objetos móveis pode ser visto como uma especialização de bases de dados espaço-

temporais [Yi & Medeiros, 2002].

Bases de dados de aplicações que tratam de muitos objetos móveis têm que ser continuamente atualizadas, ou as respostas para as consultas rapidamente ficarão obsoletas [Agarwal et al., 2000].

Em geral, a localização futura de um objeto móvel é modelada como uma função linear do tempo. Nesse caso, a trajetória torna-se um segmento de linha no domínio espaço/tempo. Assim, a localização futura de um objeto móvel pode ser determinada sem aferir a localização do objeto em intervalos regulares [Chon et al., 2003]. Com isso, o comportamento dinâmico dos objetos não necessariamente requer atualizações constantes na base de dados, que são necessárias apenas quando os parâmetros das funções de trajetória mudam.

Em [Wolfson et al., 1998] e [Sistla et al., 1997] é apresentado o modelo MOST (*Moving Objects Spatio-Temporal*). É proposta a construção de uma camada superior ao SGBD que fornece recursos para o suporte a objetos móveis, que inclui atributos dinâmicos cujos valores mudam com o tempo, e uma linguagem própria para consultas temporais em objetos móveis, a qual inclui maneiras de indexar atributos dinâmicos e tratamento da imprecisão da localização dos objetos. Neste modelo, atributos dinâmicos se alteram continuamente com o passar do tempo sem serem explicitamente atualizados, sendo representados por funções de tempo.

A maioria dos métodos de acesso direcionados para objetos móveis foram desenvolvidos a partir da *R-Tree*. Por exemplo, em [Pfoser et al., 2000] e [Pfoser, 2002] são apresentadas modificações da *R-Tree* específicas para indexar trajetórias de objetos móveis. Além de consultas por abrangência e por vizinhos mais próximos, nesses trabalhos são discutidos também alguns tipos de consultas baseadas nas trajetórias dos objetos.

Em [Kollios et al., 1999] e [Agarwal et al., 2000] foram propostos esquemas de indexação de pontos em baixas dimensões (1 ou 2), movendo-se em trajetória linear. Em ambos os trabalhos considera-se que os objetos movem-se em trajetória linear com velocidade constante. Já em [Aggarwal & Agrawal, 2003] são apresentadas estruturas para indexação de objetos de trajetória não-linear, em dimensões arbitrárias, tendo como

foco principal as consultas por vizinhos mais próximos.

Em [Güting et al., 2000] e em [Forlizzi et al., 2000] é proposto um *framework* que serve como base para representação e consultas de dados com transformações geométricas dependentes do tempo.

Partindo do princípio que um objeto móvel também é um objeto espaço-temporal, o modelo proposto em [Yi, 2004] modifica um modelo de dados geográfico-temporal([Faria, 1998]¹ *apud* [Yi, 2004]) estendendo-o para armazenar trajetórias de objetos móveis.

Em [Tao & Papadias, 2003] e [Tao & Papadias, 2002] é proposto um *framework* para transformação de “consultas espaciais” em “consultas parametrizadas pelo tempo”, adicionando ao resultado da consulta o momento em que o resultado deverá expirar e a mudança nos dados que será responsável pela expiração.

O método de acesso *TPR-Tree* (*time-parameterized R-tree*) [Saltenis et al., 2000] é uma extensão da *R*-Tree* [Beckmann et al., 1990] para indexação de objetos móveis. Utilizando uma função linear de tempo, o método é capaz de indexar a posição corrente e antecipar posições futuras de objetos em espaços de até três dimensões. Em [Jensen & Saltenis, 2002a], [Tao et al., 2003] e [Pelanis et al., 2006] são apresentadas extensões e modificações desse método, chamadas, respectivamente, *R^{EXP}-Tree*, *TPR*-Tree* e *R^{PPF}-Tree*.

Em [Iwerks et al., 2003] apresenta-se um método para coletar as respostas que permitem a manutenção de consultas por similaridade contínuas sobre os objetos móveis, quando estes são atualizados, permitindo tratar as informações de atualização de posição como um fluxo de dados (*data streams*). Para isso, são declarados alguns eventos adicionais no momento em que ocorrem as atualizações, visando filtrar e limitar o número de objetos processados durante as consultas. Além de variações para responder consultas contínuas [Mokbel et al., 2004, Mouratidis et al., 2005, Papadopoulos et al., 2007], novos tipos de consulta também tem sido propostos, como operações de junção [Zhang et al., 2008a, U et al., 2007, Corral et al., 2008] e

¹[Faria, 1998] Faria, G. (1998). Um banco de dados espaço-temporal para desenvolvimento de aplicações em sistemas de informação geográfica. Dissertação de mestrado, Universidade Estadual de Campinas.

agrupamentos [Jensen et al., 2007, Zhang et al., 2008b].

Os altos custos das operações individuais de atualização em índices baseados em “minimum bounding rectangles” (MBRs), tais como a *R-tree* e suas derivadas, motivaram o desenvolvimento de outras estruturas que adotam outras propriedades dos dados para promover sua indexação. Em [Jensen et al., 2004] utilizam-se estruturas baseadas em *B+-trees* para gerenciar objetos móveis. Já em [Patel et al., 2004] foi proposta a estrutura chamada *STRIPES*, que é baseada na *Quadtree*. Outra estrutura baseada na *Quadtree* pode ser encontrada em [Tayeb et al., 1998].

A modelagem de objetos móveis pode ser dividida em duas grandes áreas. A primeira é direcionada à aplicações altamente dinâmicas, e as consultas são relativas à posição atual e futura dos objetos indexados. Já na segunda, o enfoque das consultas é relativo ao histórico dos objetos[Yi, 2004]. Porém, novos trabalhos estão sendo propostos combinando ambas as funcionalidades [Sun et al., 2004, Lin et al., 2005, Pelanis et al., 2006, Praing & Schneider, 2007, Li et al., 2008].

Apesar de utilizar recursos como a definição de trajetórias para evitar constantes atualizações, em bases de objetos móveis o número de atualização tende a ser bastante maior em comparação às atualizações em bases de dados convencionais. Em [Ooi et al., 2002] e [Jensen & Saltenis, 2002b] são apresentadas várias estratégias para aliviar a freqüência e melhorar a performance das atualizações.

Devido à abundância de trabalhos em técnicas de indexação e busca em dados móveis e trajetórias, vários autores têm se preocupado em acompanhar o estado da arte na área. Em [Inam & Matin, 2003] é apresentado um levantamento bibliográfico de técnicas de indexação de trajetórias de objetos móveis. Em [Agarwal & Procopiuc, 2002] tem-se outro survey, focado nas consultas por abrangência e vizinhos mais próximos. Em [Nascimento et al., 1999], são comparadas várias estruturas baseadas na *R-tree* e voltadas para o suporte de “pontos móveis discretos” (que mudam de posição “instantaneamente”, e não continuamente).

4.4 Considerações Finais

A maioria das aplicações às quais os SGBD atualmente devem dar suporte manipulam, de alguma maneira, dados com características de tempo, seja para informações históricas, atuais ou futuras. Neste capítulo foram discutidos vários conceitos que envolvem a área de pesquisas sobre bases de dados temporais, como as definições de ‘tempo de transação’, ‘tempo válido’ e o modelo conceitual bi-temporal. Foram apresentados diferentes modelos de dados temporais e estruturas de indexação para dados espaço-temporais.

Os objetos móveis, que podem ser vistos como uma especialização de dados espaço-temporais, também foram discutidos neste capítulo. Foram apresentados trabalhos que possibilitam consultas à posições dos objetos no passado, no presente e no futuro.

Dinamicidade em Métodos de Acesso Métrico

5.1 Considerações Iniciais

Os MAM dinâmicos, tais como a *M-tree* e a *Slim-tree*, consideram que cada elemento de dado representa um objeto imutável com o passar do tempo. A denominação ‘dinâmicos’ contrapõe-se aos MAM inicialmente desenvolvidos, que constroem a estrutura de indexação utilizando todo o conjunto de dados numa única operação, não permitindo operações posteriores de inserção. Esses MAM são denominados estáticos na literatura. Note-se que apesar da denominação “dinâmicos”, apenas a operação de inserção é permitida, uma vez que a grande maioria dos MAM não tem sequer descrita a operação de remoção.

Neste capítulo são apresentados os resultados da primeira frente de atividades realizadas durante o desenvolvimento deste trabalho de doutorado, que abordou a indexação em espaços métricos de dados que sofrem atualização com frequência, sem que seja necessário manter o histórico dos dados. Esta frente de atividades é genérica, e deve ser universalmente aplicável a domínios de aplicação que trabalhem com dados em espaços métricos, independente de haver a necessidade de tratamento temporal aos dados.

Foram desenvolvidos e implementados algoritmos de remoção e atualização de dados, apresentados respectivamente nas seções 5.2 e 5.4, tornando os MAM realmente dinâmicos. Além disso, foi desenvolvido um novo método de otimização de MAM dinâmicos,

apresentado na seção 5.3, baseado no algoritmo de remoção proposto.

5.2 Remoção em MAM

A remoção de elementos em árvores métricas pode causar grandes reorganização das subárvores. Embora em geral não haja problemas em se remover elementos armazenados apenas nas folhas, ou seja, que não tenham sido usados como representantes em nós de níveis superiores da árvore, os elementos que estejam sendo utilizados como representantes não podem ser removidos, a não ser que a estrutura tenha as subárvores abaixo do elemento removido completamente refeitas, o que pode acarretar até a recriação de toda a árvore, quando elementos presentes na raiz são removidos.

Assim, em praticamente todos os MAM em formato de árvore, principalmente árvores balanceadas, sugere-se que a operação de remoção de elementos utilizados como representantes seja feita apenas marcando-se os mesmos como removidos, sem eliminá-los de fato. Essa alternativa é aceitável quando existem poucas operações de remoção de elementos.

No entanto, em aplicações em que a quantidade de operações de remoção (ou atualização) não é pequena (em que os elementos indexados podem evoluir no tempo, por exemplo), a alternativa de apenas marcar-se os elementos apagados torna-se inaceitável. Além de aumentar a quantidade de memória em disco necessária para armazenar os elementos indexado e, consequentemente, a quantidade de acessos que devem ser feitos aos discos, esta alternativa aumenta também a quantidade de cálculos de distância durante as operações de busca, pois elementos que não existem mais no conjunto de dados continuam tendo suas distâncias calculadas.

Outra característica que deve ser levada em consideração quanto à remoção de elementos é a Taxa de Ocupação Mínima (TOM) dos nós. O valor padrão da TOM dos nós para a *Slim-tree* é 25%. Dessa maneira, além da reorganização necessária no caso de apagar representantes, a operação de remoção pode também comprometer a TOM.

No algoritmo para remoção desenvolvido procurou-se respeitar a TOM dos nós da *Slim-tree*, mas há casos, apresentados a seguir, em que são permitidas taxas inferiores

para evitar um processo maior e mais custoso de reorganização da árvore.

Nos Algoritmos 5.1 e 5.2 são apresentados os pseudo-algoritmos para remoção de qualquer elemento indexado pela *Slim-tree*.

Algorithm 5.1: Algoritmo de remoção efetiva

```

Delete(DelObj)
  INPUT: O elemento DelObj a ser removido
  OUTPUT: TRUE se DelObj foi removido, e caso contrário FALSE

1   IF TracePointQuery(DelObj, root) retorna NULL
2     return FALSE
3   RecursiveDelete(DelObj, root)
4   IF ElementsToReinsert não está vazio
5     reinsere os elementos em ElementsToReinsert
6   return TRUE

```

Para remover o elemento *DelObj*, o Algoritmo 5.1 inicia sua execução chamando o procedimento *TracePointQuery*(*DelObj*, *root*), para localizar o elemento a ser apagado. O procedimento *TracePointQuery*(*DelObj*, *root*) realiza um busca pontual no MAM pelo elemento *DelObj*. Caso esse elemento seja encontrado, o caminho de descida na árvore métrica até o nó-folha que contém *DelObj* é armazenado, para que o algoritmo de remoção acesse o elemento mais rapidamente.

Caso o elemento não seja encontrado, o procedimento retorna NULL, e o *Delete* retorna FALSE. Caso contrário, a remoção é executada recursivamente, chamando-se o procedimento *RecursiveDelete* (Algoritmo 5.2).

Em *RecursiveDelete*(Algoritmo 5.2), além do elemento a ser removido e da página do nível corrente que leva ao elemento, também é passado como parâmetro o “Caminho”, utilizado para determinar *NóFilho* em cada um dos níveis da árvore. São também retornados como variáveis os parâmetros para atualização dos nós superiores, como raio de cobertura, número de elementos, novos representantes escolhidos, etc. Tais argumentos foram omitidos no peseudo-algoritmo para facilitar a visualização dos passos importantes.

O procedimento *RecursiveDelete* é chamado recursivamente em cada um dos níveis da árvore até chegar ao nó-folha onde *DelObj* está armazenado. Após cada chamada é retornada a ação que deve ser realizada de acordo com a situação propagada pelos níveis inferiores. NO_ACTION indica que nada precisa ser alterado na estrutura de representantes dos níveis superiores; CHANGED REP indica que foi trocado o

 Algorithm 5.2: Algoritmo recursivo de remoção

RecursiveDelete(*DelObj*, *Node*)

 INPUT: O elemento *DelObj* a ser removido e o nó corrente *Node* no caminho
 OUTPUT: NO_ACTION, CHANGED REP or REMOVED_NODE

```

1 IF Node é nó-índice
2     SWITCH (RecursiveDelete(DelObj, ChildNode))
3         CASE NO_ACTION
4             return NO_ACTION
5         CASE CHANGED REP
6             atualiza o representante da entrada ChildNode
7             IF o representante de Node mudou
8                 escolher um novo representante
9                 return CHANGED REP
10            return NO_ACTION
11        CASE REMOVED_NODE
12            definir RemoveEntry como TRUE
13            IF ExportSubtrees não está vazio // subárvores de nó filho
14                IF é possível exportar entradas em ExportSubtrees para outro nó
15                    filho de Node
16                    delete ChildNode
17                    definir RemoveEntry como FALSE
18                IF RemoveEntry é TRUE
19                    remover a entrada ChildNode de Node
20                IF Node viola TOM //TOM do nó raiz é 2
21                    IF Node é o nó raiz
22                        definir o nó apontado pela entrada como nova raiz
23                        remover Node
24                        return NO_ACTION
25                    Tentar importar um elemento
26                    IF não foi possível importar
27                        IF Node tem menos entradas que MIN_CONCESSION OR Node
28                            está na metade de baixo da árvore
29                            armazenar entradas de Node em ExportSubtrees
30                            return REMOVED_NODE
31                        IF o representante de Node foi alterado
32                            escolher um novo representante
33                            return CHANGED REP
34                        return NO_ACTION
35                    IF Node é nó-folha
36                        delete DelObj
37                        IF Node viola TOM
38                            armazema as entradas de Node em ElementsToReinsert
39                            remove Node
40                            return REMOVED_NODE
41                        IF DelObj é o representante de Node
42                            escolher um novo representante
43                            return CHANGED REP
44                        return NO_ACTION

```

representante da subárvore do nível inferior e é preciso alterar a entrada para ela, substituindo o representante antigo pelo novo; e REMOVED_NODE indica a necessidade de remover a entrada para a subárvore no nível inferior, pois ela foi removida.

Os procedimentos realizados caso a taxa de ocupação em um nó seja inferior à TOM

5.2 Remoção em MAM

dependem do tipo de nó. Em nós-folha onde o elemento está sendo removido, no caso de violação da TOM, todos os outros elementos são também removidos e armazenados em uma lista para reinserção *ElementsToReinsert*, e o nó-folha é removido da estrutura (linhas 36-38).

Já no caso de um nó-índice violar a TOM (linha 19), procura-se importar uma entrada de um dos nós irmãos (linha 24). Na importação, uma entrada para ser importada é procurada nos nós irmãos que possam exportá-la sem violar a TOM. A busca da entrada a ser importada pode ser dividida em duas alternativas possíveis. Primeiramente, entre os nós irmãos candidatos a exportadores, procura-se por entradas que já sejam cobertas pelo raio de cobertura do nó importador, ou seja, uma entrada que possa ser importada sem a necessidade de aumentar o raio de cobertura do nó importador. Se nenhuma entrada for encontrado com essa característica, a segunda alternativa é escolher a entrada que aumente minimamente o raio de cobertura do nó importador.

Se nenhum dos nós irmãos puder exportar uma entrada sem que viole a TOM, a alternativa é exportar as entradas restantes e apagar o nó que viola a TOM. A exportação dos nós restantes sempre é possível pois, se nenhum dos nós irmãos puderam exportar um elemento, então estão todos com baixa taxa de ocupação e podem receber os elementos exportados do nó a ser apagado. Note que um caso especial acontece quando o nó tem apenas 1 subárvore e $MINCONCESSION = 1$ (limite de concessão para violação da TOM, explicado a seguir). Nesse caso a subárvore não é exportada.

Para reduzir a reorganização da árvore, é feita uma concessão: se o nó-índice está na metade superior da árvore e todos os seus nós irmãos estão com suas ocupações mínimas (e não podem exportar), então é permitida a violação da TOM, até o limite definido pelo parâmetro $MINCONCESSION$ (linha 26). A aceitação da violação da TOM na metade superior da árvore não é crítica e causa pouco impacto na ocupação média da árvore, pois a proporção de nós nos níveis superiores da árvore é pequena. É importante também notar que o algoritmo tradicional que apenas marca os representantes como removido, também permite a violação da TOM até mesmo nas folhas.

Quando um representante é removido, um novo representante é escolhido (linhas 8,

30 e 40) visando minimizar o raio de cobertura do nó. O representante antigo é então substituído pelo novo na entrada do nível superior da árvore. Se, em decorrência do processo de remoção, a raiz ficar com apenas uma entrada (violando a TOM da raiz, que deve ser maior ou igual a 2), esta entrada passa a ser a nova raiz e a raiz antiga é removida (linhas 21 e 22), diminuindo a altura da árvore. A seguir são apresentados os resultados de experimentos realizados com o algoritmo de remoção efetiva proposto, comparando-o com a solução tradicional: um algoritmo de remoção que apenas marca os elementos como removidos no caso de serem representantes.

5.2.1 Experimentos

Os conjuntos de dados usados nos experimentos são apresentados na Tabela 5.1. Cada um dos conjuntos *Cidades* e *Letras* foi dividido em 2 partes, a primeira com 80% dos elementos e a outra com 20%. Para cada conjunto foi construída uma árvore com a primeira parte. Em seguida foram removidos de maneira aleatória aproximadamente metade dos elementos indexados.

Tabela 5.1: Conjuntos de dados utilizados nos experimentos.

Nome	Nr Elems.	Dim.	Capacidade dos nós	Descrição
<i>Cidades</i>	5507	2	26	Latitudes e longitudes de cidades brasileiras (www.ibge.gov.br)
<i>Letras</i>	20000	16	56	atributos extraídos de imagens de caracteres UCI Machine Learning Archive (mlearn.ics.uci.edu/MLRepository.html)

Custo das operações de remoção

A intenção deste primeiro experimento é comparar o custo da remoção efetiva ao custo da remoção com marcação, além da performance das consultas realizadas após as remoções. Além disso, outra comparação foi feita usando o algoritmo de remoção efetiva com dois valores diferentes para TOM: o valor padrão de 25% e um valor um pouco superior, de 45%.

Na Tabela 5.2 são mostrados os custos das execuções dos algoritmos de remoção. A coluna *Tempo Total* mostra o total de tempo utilizado para a realização das remoções com o conjunto *Letras* e *Cidades*. Já nas colunas *Média de acessos a disco* e *Média de cálculos*

de distâncias os valores referem-se às médias dos valores durante às mesmas remoções.

Tabela 5.2: Comparação entre os custos dos algoritmos de remoção apenas marcando os representantes removidos e o algoritmo de remoção proposto.

Conjunto/Algoritmo	# de páginas após remoções	# de páginas após remoções e inserções	Tempo Total (ms)	média de acessos a disco	média de cálculos de distâncias
<i>Letras</i>					
Remoção com marcação	599	639	6296	43.1	997.4
Remoção efetiva (TOM = 25%)	363	442	6328	44.9	1058.9
Remoção efetiva (TOM = 45%)	276	431	7109	45.2	1170
<i>Cidades</i>					
Remoção com marcação	382	408	203	13.9	46.4
Remoção efetiva (TOM = 25%)	248	309	218	14.9	76.5
Remoção efetiva (TOM = 45%)	232	293	349	20.9	128.4

Comparando-se inicialmente o algoritmo de remoção efetiva com valor de TOM de 25% com o algoritmo de remoção apenas marcando os representantes removidos, na Tabela 5.2 pode-se ver que o número de acessos a disco foi bem parecido para ambos os conjuntos. Considerando os cálculos de distâncias, o algoritmo de marcação realizou 6% menos cálculos de distância para o conjunto *Letras* e 40% menos cálculos de distância para o conjunto *Cidades*. Com o conjunto *Letras*, o tempo total para a execução das remoções foi praticamente o mesmo para os dois algoritmos, e aproximadamente 7% mais rápido com o conjunto *Cidades*. Porém, o número de páginas na árvore após a execução do algoritmo de marcação foi 54% maior para o conjunto *Cidades* e 65% para o conjunto *Letras* (*# de páginas após remoções*, na Tabela 5.2).

A execução do algoritmo de remoção efetiva com TOM de 45% elevou o custo da operação de remoção. Comparando-se com o algoritmo de remoção com TOM de 25%, no conjunto *Letras* ocorreu um aumento de aproximadamente 12% no tempo de execução, e no conjunto *Cidades* esse aumento foi de aproximadamente 60%. Entretanto, além do aumento na performance nas consultas posteriores, que será mostrado a seguir, o número de páginas na árvore após a execução do algoritmo de marcação foi 65% maior para o conjunto *Cidades* e 117% para o conjunto *Letras*, em comparação com o algoritmo de remoção efetiva com TOM de 45%. Os experimentos mostraram que o custo da remoção efetiva é sempre maior do que meramente marcar os elementos como removidos, como é previsível. No entanto, as árvores resultante da remoção efetiva sempre foram melhores

para responder consultas posteriores.

Consultas após remoções

Primeiramente foi avaliada a performance das consultas sobre a árvores resultante dos processos de remoções, comparando as estruturas resultantes das remoções conforme a Tabela 5.2. Os resultados apresentados na Figura 5.1 referem-se ao desempenho de 500 consultas k-NN com k variando entre 5 e 50.

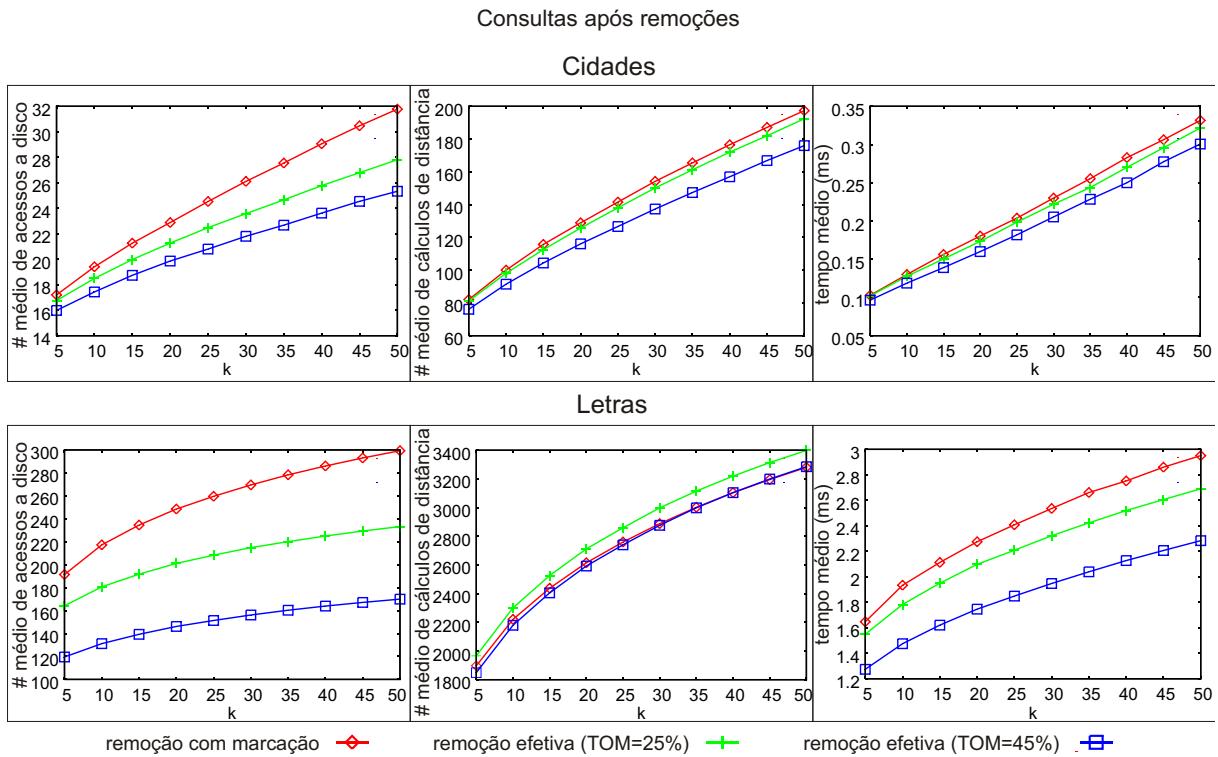


Figura 5.1: Desempenho de consultas realizadas após remoções, comparando o algoritmo proposto (com dois valores diferentes de TOM) com o algoritmo de remoção apenas marcando os representantes removidos. Número médio de acessos a disco (primeira coluna), número médio de cálculos de distâncias (segunda coluna) e tempo médio em milisegundos (terceira coluna) em consultas k-NN variando k .

Como pode ser visto na Figura 5.1, nos experimentos com o conjunto *Letras*, as consultas realizadas sobre as árvores resultantes do processo de remoção utilizando o algoritmo de remoção efetiva foram sempre mais rápidas. As consultas foram até 9% mais rápidas na árvore resultante das remoções com TOM de 25%, e até 24% mais rápidas na estrutura resultante do processo utilizando TOM de 45%. O número médio de acessos a disco nas consultas também foi significativamente reduzido após a utilização da remoção

efetiva. A redução chegou a 22% com Tom de 25%, e a 43% com TOM de 45%.

Assim como aconteceu com o conjunto *Letras*, com o conjunto *Cidades*, as consultas realizadas sobre as árvores resultantes da remoção efetiva dos elementos apresentaram sempre melhor performance. As consultas realizadas após as remoções e antes das inserções foram até 5% mais rápidas na árvore resultante das remoções com TOM de 25%, e até 12% com TOM de 45%. A redução no número médio de acessos a disco chegou a 20%, e a redução no número médio de cálculos de distância atingiu 11%, ambos com Tom de 45%.

Consultas após remoções e inserções

Visando avaliar a performance das consultas após as remoções seguidas de novas inserções, foram inseridos os elementos das segundas partes dos conjuntos nas árvores resultantes das remoções realizadas no experimento anterior. Na Tabela 5.2 também é mostrada a quantidade de páginas na árvore após as inserções (*# de páginas após remoções e inserções*). Na Figura 5.2 são mostrados gráficos referentes ao desempenho de 500 consultas k-NN com k variando entre 5 e 50.

Após a inserção dos elementos restantes, com o conjunto *Letras* as consultas continuaram a ser mais rápidas nas árvores resultantes das remoções utilizando o algoritmo proposto: até 5 e 15%, respectivamente para TOM de 25 e 45%, respectivamente. Também nos acessos a disco, as consultas após os algoritmos de remoção efetiva apresentaram resultados melhores, alcançando reduções superiores a 25% com TOM de 45%. Analisando-se os cálculos de distâncias, verifica-se que com o conjunto *Letras* houve até um pequeno aumento (máximo de 4.6%) nesse quesito nas consultas realizadas após a execução da remoção com TOM de 25%. Apesar de ser um resultado não apresentado em outros experimentos, isto pode ser explicado pela redução do número de páginas na árvore com a utilização do algoritmo de remoção efetiva (e consequentemente redução dos acessos a disco), que leva a um aumento no número de entradas nos nós, aumentando portanto o número de cálculos de distâncias necessários ao acesso a cada nó. Apesar disso, como já foi dito, o tempo de execução das consultas foi sempre menor após a utilização do algoritmo proposto.

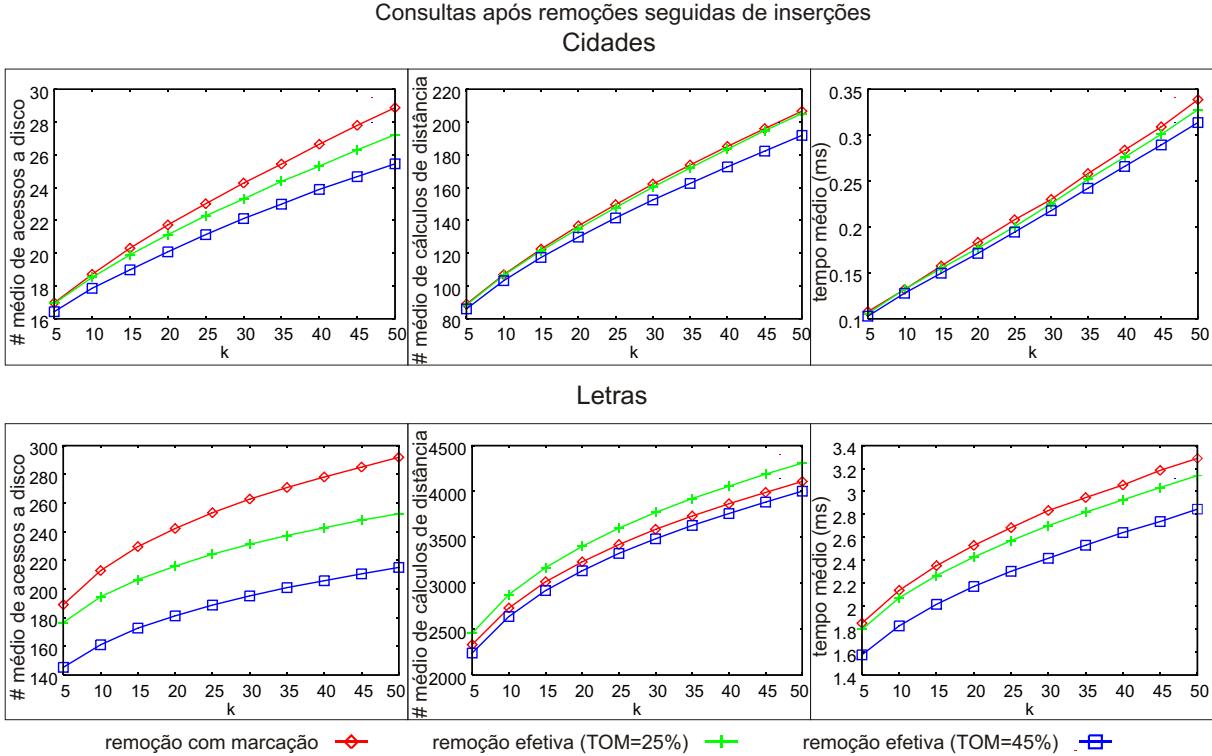


Figura 5.2: Desempenho de consultas realizadas após remoções seguidas de inserções, comparando algoritmo proposto (com dois valores diferentes de TOM) com o algoritmo de remoção apenas marcando dos representantes removidos. Número médio de acessos a disco (primeira coluna), número médio de cálculos de distância (segunda coluna) e tempo médio em milisegundos (terceira coluna) em consultas k-NN variando k. .

Também depois da inserção da segunda parte do conjunto *Cidades*, as consultas continuaram a ser sempre mais rápidas após a execução de remoções efetivas (até 7,4%), exigindo menos acessos a disco (redução de até 12%) e cálculos de distância (redução de até 7%).

Além dos resultados apresentados acima é importante salientar que com a continuidade deste processo de remoção/inserção dos dados indexados (comportamento esperado conforme a atualização de uma estrutura dinâmica com o correr do tempo), os ganhos de desempenho do algoritmo proposto sobre o algoritmo que apenas marca os representantes removidos tendem a aumentar cada vez mais, pois espera-se que o número de elementos apagados mas que permanecem na árvore tende a crescer com a continuação das remoções apenas marcando os representantes removidos.

5.3 Uma técnica para redução de sobreposição e otimização em MAM dinâmicos

Durante os experimentos com o algoritmo de remoção proposto na seção anterior com diversos conjuntos de dados, um resultado interessante notado foi a melhora na performance das consultas após a remoção seguida da reinserção dos mesmos elementos. Para exemplificar este comportamento, foi realizado o seguinte experimento: o conjunto todo foi indexado, dando origem a estrutura que na Figura 5.3 é denominada *estrutura original*. Em seguida, foram removidos 500 elementos dessa estrutura, aleatoriamente, e depois esses mesmos elementos removidos foram reinseridos. O mesmo procedimento foi realizado para os conjuntos *Letras* e *Cidades*, utilizando o algoritmo de remoção efetiva com TOM de 45% e 60%, respectivamente.

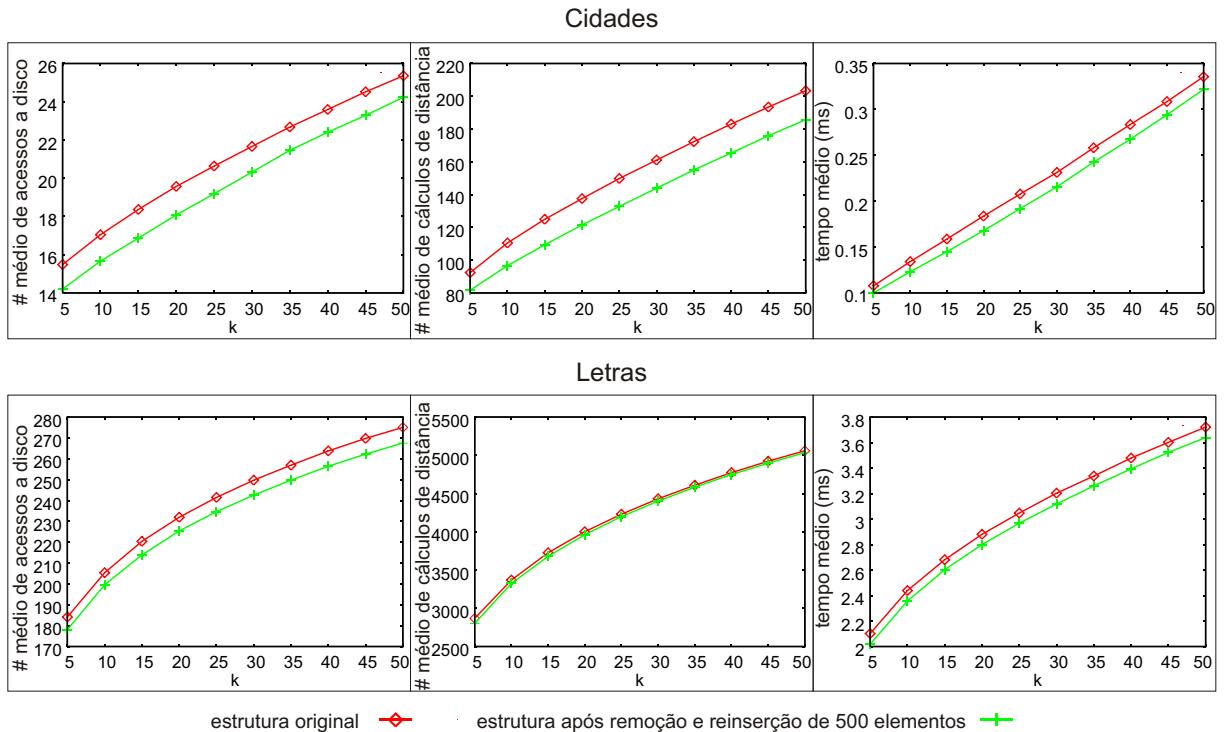


Figura 5.3: Comparação do desempenho de consultas realizadas sobre a estrutura original e sobre a estrutura resultante do processo de remoção e reinserção de 500 elementos, com o conjunto *Cidades* (TOM = 60%) e *Letras* (TOM = 45%). Número médio de acessos a disco (primeira coluna), número médio de cálculos de distâncias (segunda coluna) e tempo médio em milisegundos (terceira coluna) em consultas k-NN variando k. .

Como pode ser visto na Figura 5.3, o desempenho das consultas sobre as estruturas resultantes foi superior ao desempenho das consultas sobre a estrutura original. Pode-se concluir que o procedimento realizado otimizou a estrutura de indexação, que contém exatamente os mesmos elementos antes e depois do processo de remoção/reinserção.

Surgiu daí a idéia de propor uma nova técnica de otimização para MAM utilizando o algoritmo de remoção proposto. Porém, ao invés de remover e reinserir elementos aleatoriamente da árvore, como no experimento mostrado acima, o método proposto procura remover elementos que não estejam próximos ao centro do nó onde ele é armazenado, fazendo com que o raio desse nó seja maior, possivelmente causando maior sobreposição com outros nós, o que piora a performance das consultas realizadas sobre essa árvore. A idéia é remover vários elementos na periferia dos nós e depois reinserí-los de uma vez em uma única operação.

Na execução do algoritmo de otimização *Slim-down* [Traina Jr. et al., 2002a], quando um nó-folha apresenta sobreposição, o elemento mais distante do representante do seu nó é migrado para um nó irmão que já o cubra. Porém, o *Slim-down* restringe as comparações aos nós-folha de uma mesma subárvore, ou seja, apenas os nós-folha ligados a um mesmo nó-índice. Portanto, no caso de sobreposição de dois nós-folha que estejam ligados a diferentes nós-índice, nenhuma operação é realizada para diminuir ou eliminar sobreposições. Em [Skopal et al., 2003] é proposto o algoritmo *Generalized Slim-down*, que pode movimentar os elementos entre qualquer subárvore. Porém, como pode-se inferir pelos experimentos do autor, essa otimização chega a ser até duzentas vezes mais custosa do que a construção inicial da árvore.

5.3.1 Push Pull

A técnica de otimização proposta, chamada *Push-pull*, utiliza o algoritmo de remoção efetiva para remover um número definido de elementos que estejam na periferia dos nós-folha, para posteriormente reinserí-los na árvore. Os elementos escolhidos são os mais distantes de seus respectivos representantes. Dessa maneira, com a remoção, garante-se que o nó-folha reduza seu raio de cobertura.

Na reinserção do elemento removido, caso nenhuma subárvore cubra o elemento, é

escolhida a subárvore que possa cubrí-lo com o menor aumento de raio possível. Caso haja mais de uma subárvore com cobertura, escolhe-se a subárvore com elemento representante mais próximo do elemento a ser reinserido. Muitos elementos tendem a ser reinseridos em outros nós que já o cubram, ou em subárvores com o menor crescimento de raio para tal inserção, reduzindo assim a sobreposição entre os nós da árvore.

Na Figura 5.4 é mostrada passo a passo a execução da técnica de otimização proposta, partindo de uma organização inicial de dois nós-índice, cada um com três nós-folha (Figura 5.4a). Considera-se para este exemplo que a ocupação máxima de elementos por nó é 8 (oito), e a ocupação mínima 2 (dois). Nesse exemplo, o número de elementos a serem removidos por nó-folha foi definido como 2 (aqueles mais distantes dos representantes). Os elementos que serão removidos estão destacados na Figura 5.4b). Na Figura 5.4c) é mostrada a árvore após a remoção dos elementos e antes das reinserções. Pode-se ver que o nó-índice da esquerda passa a ter apenas dois nós-folha, pois um dos nós-folha foi removido por violar a TOM (no caso ficou vazio). Finalmente, na Figura 5.4d) é mostrada a organização final otimizada, após as reinserções.

Com a nova técnica de otimização proposta, a “migração” dos elementos inseridos nas folhas pode ocorrer entre qualquer nós-folha, não limitando-se a trocas entre nós irmãos, como no caso do *Slim-down*. Comparando-se a organização inicial (Figura 5.5a)) com a obtida após a execução do *Slim-down* (Figura 5.5b)), pode-se constatar a diminuição da sobreposição entre os nós-folha ligados a um mesmo nó-índice, mas não da sobreposição entre nós-folha ligados a diferentes nós-índice. Já depois da execução da técnica proposta removendo dois elementos por nó (Figura 5.5c)), pode-se ver que ocorreu tanto a redução da sobreposição entre nós-folha ligados a um mesmo nó-índice como também ligados a diferentes nós-índices.

Pelos resultados dos experimentos (mostrados na próxima seção) pode-se ver que o número ideal de elementos removidos para o melhor resultado da técnica de otimização varia de conjunto para conjunto, mas tende a um ponto de saturação. Foi verificado também que esse número ideal varia de nó para nó, de acordo com sua taxa de sobreposição. A partir dessas constatações, foi criado o algoritmo *Smart Push Pull*, descrito na seção

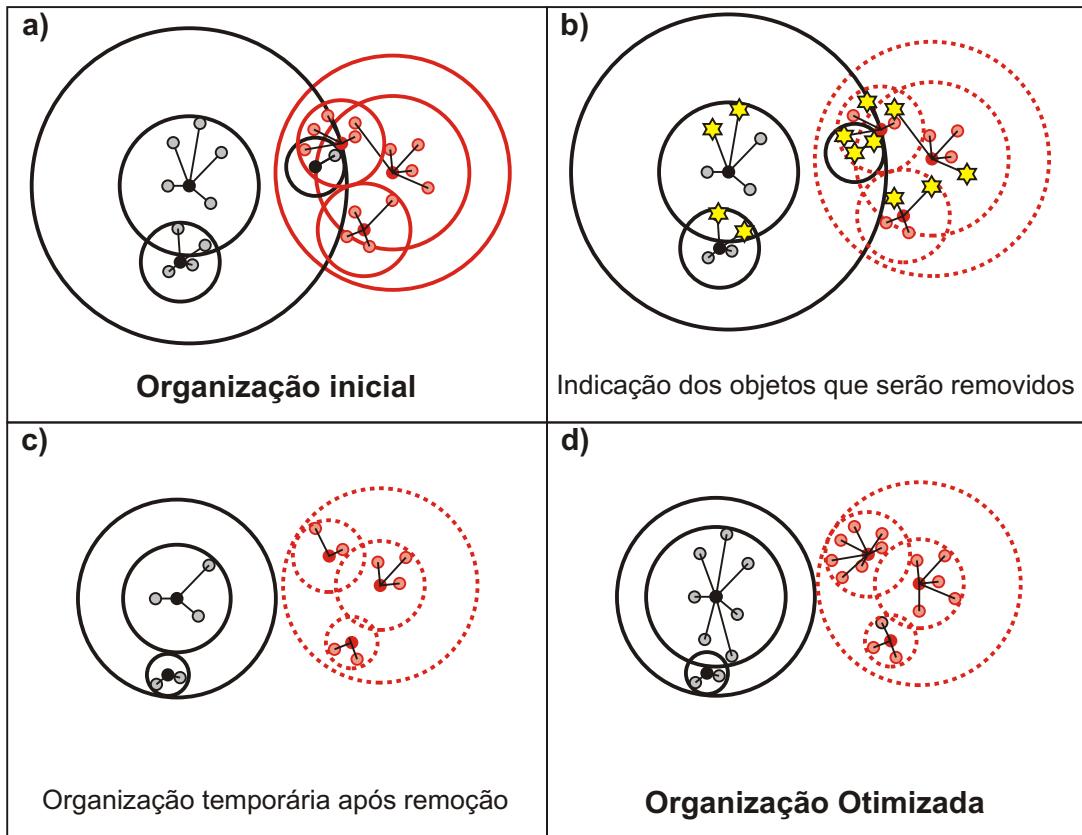


Figura 5.4: Execução da técnica *Push Pull* removendo 2 elementos por nó-folha.

seguinte.

5.3.2 Smart Push Pull

Foi desenvolvida uma técnica para a definição automática de um valor adequado para o número de elementos a ser removido por nó. O cálculo desse valor é baseado em estatísticas obtidas da árvore durante o processo de avaliação da sobreposição (*fat-factor*), normalmente utilizada para indicar a necessidade do processo de otimização. Essa técnica define o número de elementos a serem removidos individualmente para cada nó-folha. Em nós que tenham mais sobreposição, o número de elementos removidos é maior, e em nós com nenhuma sobreposição, nenhum elemento é removido. O algoritmo de otimização que usa essa técnica para definição do número de elementos a ser removido por nó foi chamado de *Smart Push-pull*.

A base para o cálculo desse número é o algoritmo do *fat-factor*, modificado para que para cada nó-folha seja armazenado o número médio de acessos a disco necessários

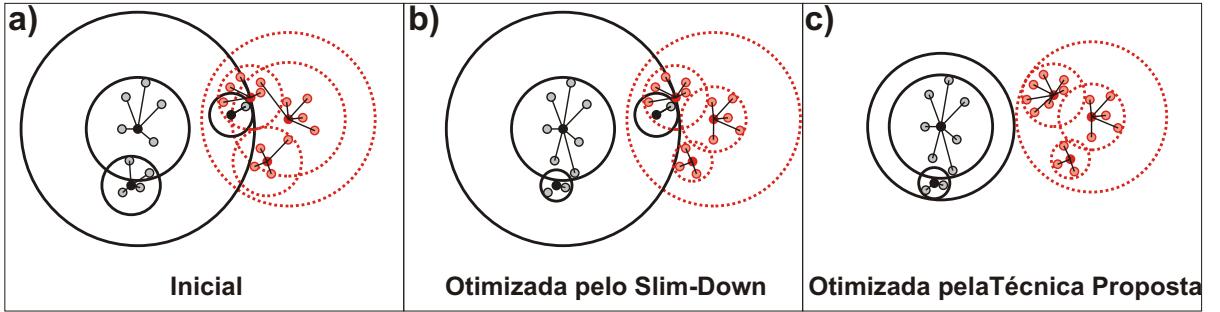


Figura 5.5: Comparação entre as otimizações obtidas pela Técnica proposta e pelo *Slim-Down*.

para recuperar as entradas do nó por consultas pontuais (AVG_{Node}). O valor máximo de AVG_{Node} entre todos os nós-folha da árvore, chamado de AVG_{max} , indica o nó com maior média de sobreposição da árvore. Baseado nessas estatísticas, o número de elementos a serem removidos em um determinado nó-folha, $\#Obj_{Del}$, é definido como:

$$\#Obj_{Del} = \frac{AVG_{Node} - h}{AVG_{max}} * Max_Occup, \quad (5.1)$$

onde Max_Occup é a capacidade dos nós da árvore e h é a altura da árvore.

O valor ótimo para AVG_{Node} é a altura da árvore, indicando que não existe sobreposição entre os elementos armazenados nesse nó com nenhum outro elemento da árvore. Nesse caso, nenhum elemento deve ser removido do nó. Além disso, o valor de $\#Obj_{Del}$ é limitado a 40% da capacidade dos nós, uma vez que os resultados dos experimentos indicam que este valor está próximo do ponto de saturação.

5.3.3 Experimentos

Os conjuntos de dados usados nos experimentos a seguir são apresentados na Tabela 5.3. Na Figura 5.6 são mostrados os resultados de 500 de consultas k-NN ($k=10$) sobre as árvores sem passar nenhuma otimização, otimizadas pelo *Slim-down*, otimizadas pela *Push-pull* variando o número de elementos removidos por nó, e otimizadas pelo *Smart Push-pull*.

Considerando o tempo total de execução, as árvores otimizadas pelo *Push-pull* obtiveram resultados melhores com o aumento do número de elementos removidos por

Tabela 5.3: Conjuntos de dados utilizados nos experimentos.

Nome	Nr Elems.	Dim.	Capacidade dos nós	Descrição
Cidades	5507	2	26	Latitudes e longitudes de cidades brasileiras (www.ibge.gov.br)
Letras	20000	16	56	atributos extraídos de imagens de caracteres UCI Machine Learning Archive (mlearn.ics.uci.edu/MLRepository.html)
ColorHisto	12000	256	49	Histogramas de cor de imagens da Amsterdam Library of Object Images [Geusebroek et al., 2005] (http://www.science.uva.nl/aloi)
SynthData	200000	64	94	Vetores de dados sintético com distribuição Gaussiana com 100 agrupamentos em um hipercubo 64-dimensional (gerado pela ferramenta DBGen [Ferreira et al., 2005])

nó. Porém, pode-se notar um ponto de saturação entre 30 e 40% da capacidade do nó. É visível que a performance das consultas realizadas sobre as árvores otimizadas com o *Smart Push-pull* supera as otimizadas pelo *Slim-down* e pelo *Push-pull*, sendo mais rápidas na maioria dos casos. O mesmo comportamento pode ser notado com relação às outras medidas de desempenho, exceto no número médio de acessos a disco para o conjunto Colorhisto, em que o *Slim-Down* superou em 5% o *Smart Push-pull*.

Quanto ao *Fat-factor relativo* – *rFat*, com todos os conjuntos pode-se notar comportamento parecido: a árvore otimizada pelo *Slim-down* possui *rFat* menor que a árvore não otimizada, mas as árvores otimizadas pela técnica proposta possuem *rFat* ainda menor, indicando árvores com menos sobreposição entre os nós e mais otimizadas.

Outro resultado importante é que o *Smart Push-pull* sempre alcançou resultados próximos ao melhor caso da técnica *Push-pull*, tirando do usuário a responsabilidade de definir o número de elementos a ser removido por nó.

Na Figura 5.7 são mostrados os resultados de 500 de consultas variando *k* entre 5 e 50, comparando seus desempenhos sobre árvores não otimizadas, otimizadas pelo *Slim-down* e otimizadas pelo *Smart Push-pull*.

Como pode ser visto, as árvores otimizadas pelo *Smart Push-pull* responderam às consultas sempre mais rapidamente, sendo até 28% mais rápida em comparação à árvore não otimizada e 23% mais rápida que a árvore otimizada pelo *Slim-down*.

Para testar o comportamento da técnica de otimização proposta com diferentes tamanhos de conjuntos de dados, foi utilizado um conjunto de dados sintético, variando

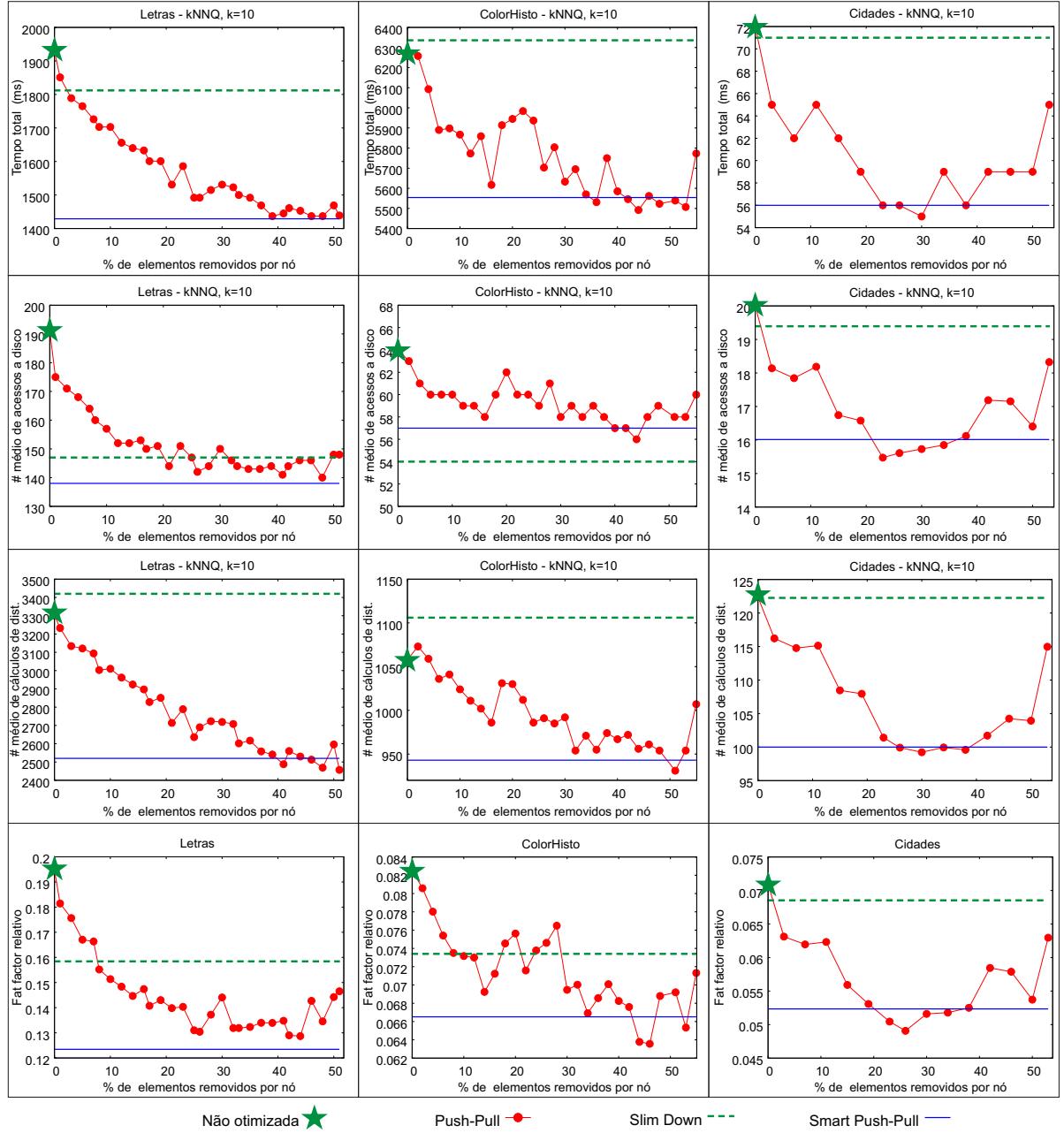


Figura 5.6: Comparação do tempo total de processamento (primeira linha), número médio de acessos a disco (segunda linha) e número médio de cálculos de distância (terceira linha) para processamento de 500 consultas 10-NN, e *relative fat-factor* (quarta linha) de uma *Slim-tree* não otimizada, uma *Slim-tree* otimizada pelo *Slim-down* e *Slim-trees* otimizadas com a técnica *Push-pull* (variando o número de elementos removidos por nó) e *Smart Push-pull*

o tamanho do conjunto de dados entre 20 e 200 mil elementos de 64 dimensões, com 100 agrupamentos (*clusters* com distribuição Gaussiana. A Figura 5.8 mostra a comparação de desempenho de 500 consultas 10-NN realizadas sobre a árvore não otimizada, sobre a

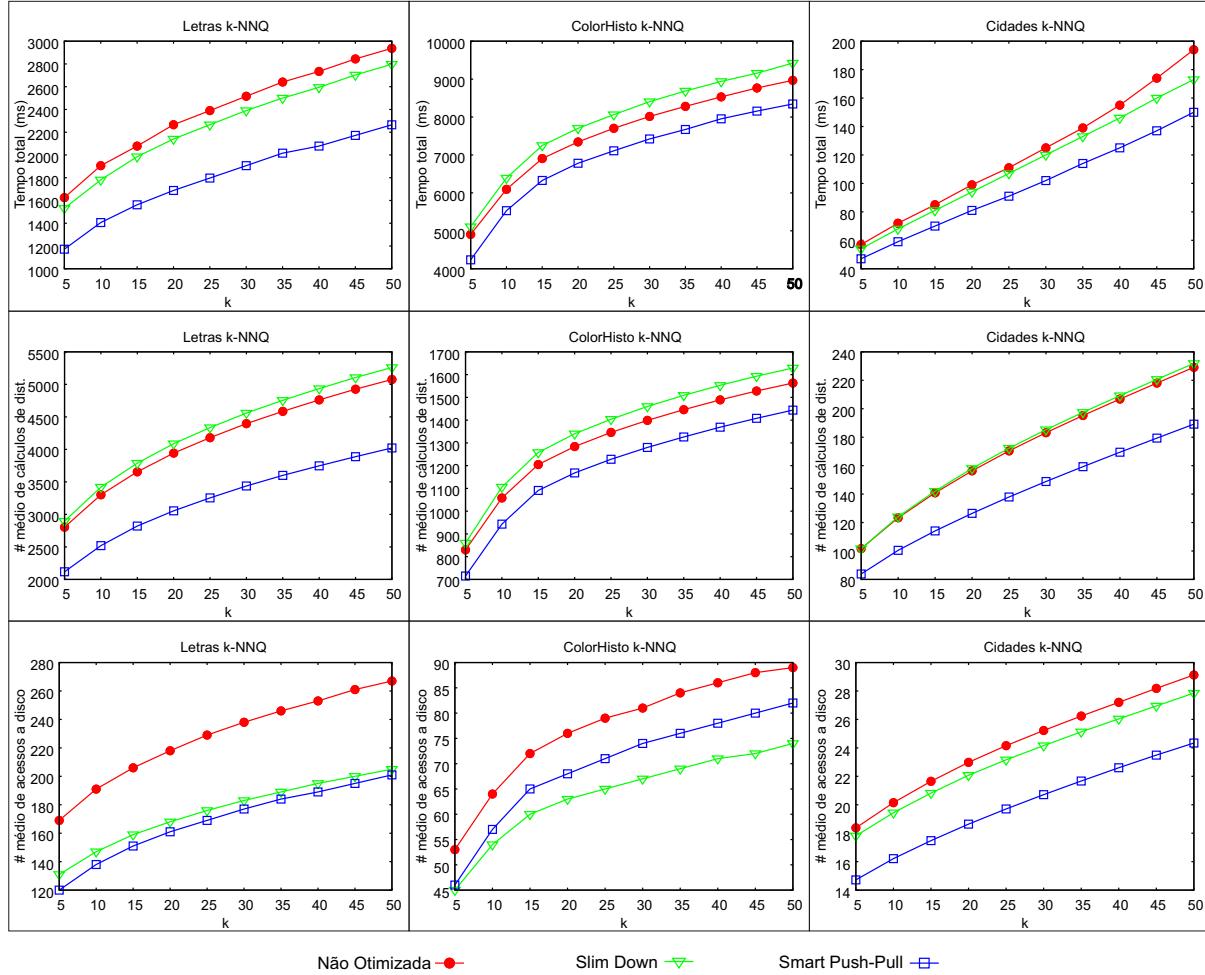


Figura 5.7: Comparação do tempo total de processamento (primeira linha), número médio de acessos a disco (segunda linha) e número médio de cálculos de distância (terceira linha) para processamento de 500 consultas variando o valor de k entre 5 e 50, sobre uma *Slim-tree* não otimizada, uma *Slim-tree* otimizada pelo *Slim-down* e *Slim-tree* otimizada com o método *Smart Push-pull*.

árvore otimizada pelo *Slim-down* e otimizada pelo *Smart Push-pull*.

Na Figura 5.8 pode-se ver que as árvores otimizadas pela técnica proposta sempre apresentaram melhor desempenho do que as árvores não otimizadas e do que as otimizadas pelo *Slim-down*. As consultas realizadas sobre as árvores não otimizadas precisaram de até 196% mais tempo, precisaram de até 140% mais acessos a disco e calcularam até 125% mais distâncias do que as árvores otimizadas pela técnica proposta. As consultas realizadas sobre as árvores otimizadas pelo *Slim-down* gastaram até 150% mais tempo, acessaram até 75% mais disco e calcularam até 186% mais distâncias do que as árvores otimizadas pelo *Smart Push-pull*.

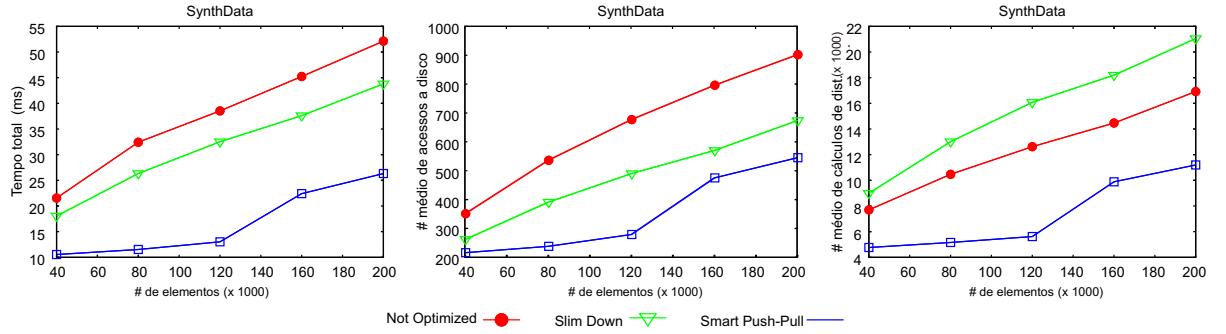


Figura 5.8: Comparação do tempo total de processamento, número médio de acessos a disco e número médio de cálculos de distância para processamento de 500 consultas 10-NN realizadas sobre uma *Slim-tree* não otimizada, uma *Slim-tree* otimizada pelo *Slim-down* e uma *Slim-tree* otimizada com a técnica *Smart Push-pull*, variando o tamanho do conjunto de dados

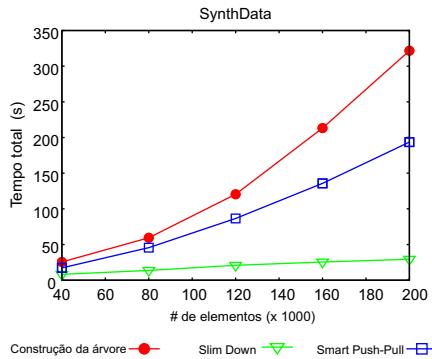


Figura 5.9: Comparação do tempo total de processamento para otimização com o *Slim-down* e *Smart Push-pull*, comparado ao tempo para construção da árvore inicial, variando o tamanho do conjunto.

Como pode ser visto na Figura 5.9, o tempo de processamento do *Smart Push-pull* variou entre 60 e 77% do tempo de construção da árvore. Como esperado, a técnica proposta é mais cara que o *Slim-down*, pois permite a reinserção dos elementos removidos em qualquer parte da árvore, diferente da ação localizada do *Slim-down*. O tempo de execução variou entre 2 e 6.5 vezes o tempo de execução do *Slim-down*. Porém, a otimização é aplicada apenas uma vez na árvore, comparando-se às consultas que são executadas muitas vezes, e tem uma melhora de desempenho de até 150% sobre o *Slim-down*.

5.4 Atualização em MAM dinâmicos

Os MAM existentes, mesmo aqueles considerados dinâmicos, consideram que cada elemento de dado representa um objeto imutável no tempo.

Uma operação de atualização pode ser realizada com a execução de uma operação de remoção do valor antigo seguida de uma operação de inserção do valor novo. No entanto, essas duas operações são executadas tendo toda a estrutura como escopo das operações de busca e correção. A inclusão da operação de atualização pode restringir esse escopo, fazendo com que essa operação tenha um custo equivalente apenas à operação de inserção ou remoção.

A possibilidade de remoção efetiva de elementos da árvore e o fato de re-inserções apresentarem uma tendência a melhorar o desempenho da estrutura para operações de consulta permitem antever a possibilidade de resolver o problema de atualização frequente nos dados da estrutura.

O algoritmo proposto é o primeiro algoritmo de atualização de dados indexados em um MAM. O algoritmo é baseado nas idéias de atualização da R-tree, principalmente no trabalho de [Lee et al., 2003], que apresentada uma estratégia de atualização *bottom-up* para R-trees, generalizando outras técnicas de atualização existentes, como as “*lazy updates*” [Kwon et al., 2002], que atualiza a estrutura de indexação somente se um elemento se move para fora de seu MBR, tentando aumentar a MBR para que o nó continue cobrindo o elemento. Isso pode contribuir para o menor custo da atualização, mas as operações de remoção e reinserção podem gerar uma estrutura melhor, com menos sobreposição e com melhor performance de consultas [Ooi et al., 2002]. No algoritmo desenvolvido procurou-se balancear esses dois fatores, tentando diminuir o custo da operação de atualização, comparado com as operações de remoção e inserção, sem prejudicar a performance das consultas posteriores.

A ausência de coordenadas espaciais no espaços métricos impossibilita a utilização de funções para estimar movimento e trajetórias, comumente usadas em métodos de acesso espaciais. Como os MAM indexam os elementos utilizando apenas a função de distância entre eles, a divisão do espaço de indexação é diferente do que ocorre com a R-tree.

5.4 Atualização em MAM dinâmicos

Dessa forma, não é possível utilizar o mecanismo para estender a MBR especificamente na direção do movimento do objeto, como em [Lee et al., 2003].

No caso de um elemento manter-se coberto pelo mesmo nó-folha após a atualização, apenas o elemento é modificado. Além disso verifica-se se o raio de cobertura do nó-folha diminuiu. Caso tenha diminuído, a redução é propagada para os níveis superiores, com possibilidade de redução de sobreposição na árvore. Caso contrário, a atualização resume-se apenas a alteração nas características do elemento atualizado.

Já no caso de o elemento alterado deixar de ser coberto pelo seu nó-folha atual, remove-se o elemento do nó-folha atual e verifica-se se o elemento é coberto pelo nó-índice imediatamente superior. Caso seja coberto, ao invés de crescer o raio de cobertura do nó-folha até o limite de cobertura do pai, escolhe-se em qual dos filhos desse nó-índice (pai) o elemento será inserido, com o algoritmo de escolha de subárvores em uso pelo MAM. Note que o nó-folha escolhido pode ser o mesmo de onde o elemento foi removido. Nesse caso o nó-folha tem seu raio aumentado, mas usando o algoritmo de escolha entre subárvores, garante-se que o nó escolhido é o melhor para o novo elemento, o que não ocorreria no caso de apenas aumentar-se o raio do nó-folha, sendo que o novo elemento poderia ser coberto por outro nó-folha sem aumento de raio.

Caso não seja coberto pelo nó-índice imediatamente superior, a mesma verificação é feita no nível superior, e assim sucessivamente até chegar a raiz, o que equivaleria ao processo de remoção seguido de reinserção. Porém, nos casos em que as atualizações forem localizadas, ou seja, a distância entre o novo elemento e o antigo seja pequena, esta técnica de atualização *bottom-up* tende a ser menos custosa. E no pior caso, de as atualizações não serem locais, a técnica de atualização proposta não é mais cara que as operações de remoção e inserção, pois os nós-índice acessados na tentativa de reinserir o elemento modificado já foram lidos do disco durante a incursão na árvore.

Quando o elemento é removido de seu nó-folha original, são utilizados os mesmos mecanismos do algoritmo de remoção efetiva, com controle da TOM e propagação das modificações para os níveis superiores.

5.4.1 Experimentos

Vários experimentos foram realizados para a validação do algoritmo desenvolvido, com análise da consistência da estrutura de indexação após as atualizações. Nos experimentos a seguir foi usado o conjunto de imagens *Amsterdam Library of Object Images* (ALOI) [Geusebroek et al., 2005].), que tem 1000 objetos, com imagens de vários ângulos de rotação dos objetos. Inicialmente foram indexadas as imagens com rotações múltiplas de 45 graus (0, 45, 90, 135, 180, 225, 270 e 315), com um total de 8000 imagens indexadas. Para realização das atualizações, escolheu-se a imagem do objeto com rotação de 5 graus sobre a imagem original. Por exemplo, para atualizar o elemento 123_r45 (objeto 123 com rotação de 45 graus), o novo valor equivale aos elemento 123_r50 (mesmo objeto 123, mas com 50 graus de rotação).

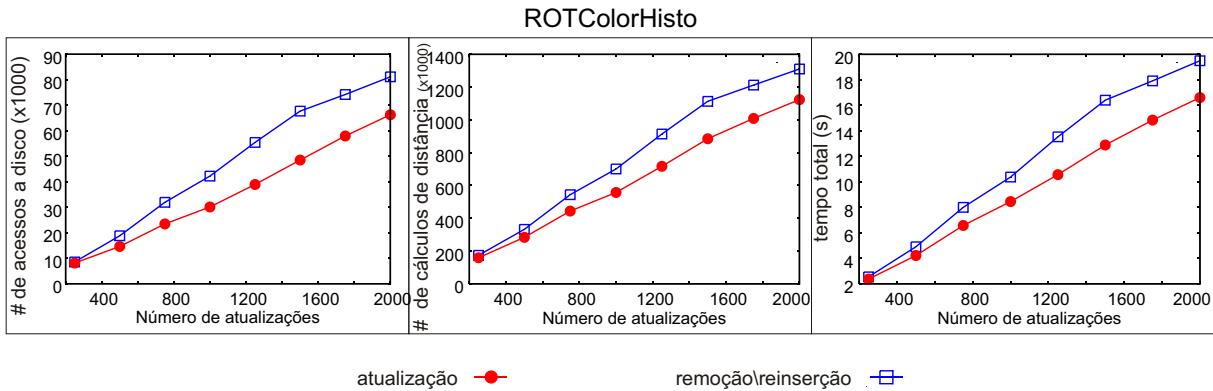


Figura 5.10: Comparação dos custos das atualizações realizadas removendo e reinserindo os elementos e realizadas pelo algoritmo de atualização proposto, variando o número de atualizações realizados antes das consultas.

Como pode ser visto na Figura 5.10, as atualizações com o algoritmo proposto foram sempre mais rápidas que as atualizações removendo e inserindo o elemento com seu novo valor, com ganhos de até 22% no tempo de execução, 21% nos cálculos de distância e até 30% nos acessos a disco.

As consultas realizadas após as atualizações feitas pelo algoritmo proposto acessaram até 9% menos disco. Quanto aos cálculos de distância, os desempenhos foram bem similares, com a diferença máxima de 2,5% a mais ou menos. O tempo total das consultas chegou a ser até 3% mais rápido, sendo a se mais custoso, no máximo, 0,4%. Note-se que

5.4 Atualização em MAM dinâmicos

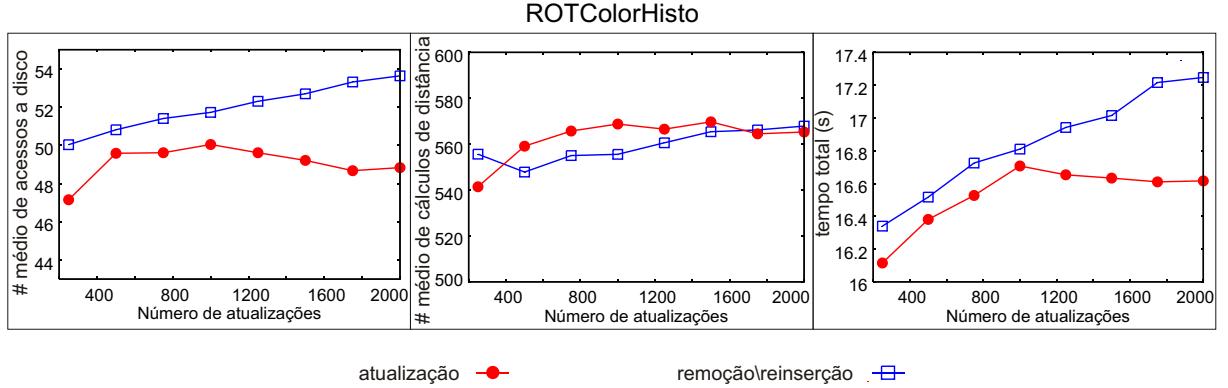


Figura 5.11: Comparação dos desempenho em consultas após a atualizações realizadas removendo e reinserindo os elementos e pelo algoritmo de atualização proposto. Tempo total (em ms) (primeira coluna), número médio de acessos a disco (segunda coluna) e número médio de cálculos de distâncias (terceira coluna) em consultas 10-NN, variando o número de atualizações realizados antes das consultas.

manter uma árvore sempre eficiente é o resultado importante desse algoritmo, impedindo que as atualizações constantes prejudiquem a qualidade da estrutura.

5.5 Considerações Finais

Os MAM dinâmicos não disponibilizam as operações de remoção e atualização, considerando que os elementos de dados indexados são imutáveis com o passar do tempo. A denominação ‘dinâmicos’ deve-se à disponibilidade da operação de inserção, e contrapõe-se aos MAM ‘estáticos’ inicialmente desenvolvidos, que constroem toda a estrutura de indexação numa única operação, sem permitir inserções posteriores.

Neste capítulo foram apresentados os algoritmos de remoção e atualização desenvolvidos para MAM dinâmicos, implementados sobre o MAM *Slim-tree*. Foram realizados vários experimentos que mostraram que as árvores resultante da remoção efetiva são melhores para responder consultas posteriores, comparadas com a remoção apenas marcando os representantes removidos.

Também foi desenvolvido um método de otimização para MAM dinâmicos, baseado no algoritmo de remoção efetiva, que reduz a sobreposição dos nós promovendo a remoção e reinserção de elementos na periferia dos nós da árvore. Os experimentos mostraram que as árvores otimizadas pela técnica *Smart Push-pull* responderam às consultas posteriores de maneira mais eficiente, com ganhos de desempenho superiores a 190% sobre árvores não otimizadas.

Evolução temporal em Dados Métricos

6.1 Considerações Iniciais

Independente das propriedades de um domínio de dados, muitas aplicações devem acompanhar a evolução dos dados durante o tempo. Na literatura existem muitos modelos propostos para representar informações temporais associadas a dados armazenados em SGBDs. Os modelos existentes são focados em aplicações onde os dados são simples ou atômicos (por exemplo números, datas e textos curtos), ou então dados dimensionais (dados espaciais e multidimensionais). A associação do tempo a dados espaciais ocorre com freqüência, mas essa é uma associação natural, pois cada dimensão temporal pode ser considerada como mais uma dimensão em um espaço que já tem diversas dimensões, homogeneizando o tratamento do tempo com as demais dimensões espaciais. Domínios espaciais têm recebido bastante atenção no contexto de aplicações com dados geo-referenciados, nos chamados modelos espaço-temporais.

Entretanto, dados métricos não têm associação com o conceito de dimensionalidade. Um espaço métrico freqüentemente é adimensional, portanto a inclusão de dimensões temporais deve ser feita separada do tratamento dos aspectos métricos. Para ilustrar esse fato, considere-se o seguinte exemplo. Suponha-se que um paciente realiza um determinado exame clínico que mede n sinais vitais. Se numa data posterior o mesmo exame for refeito, mas com medidas de novos sinais vitais (fato que pode ocorrer por

exemplo quando uma medida é composta por uma série de valores cuja quantidade não é definida a priori) não se pode mais associar cada sinal vital a uma dimensão, dado que cada instância de um exame pode ter um número variável de valores. No entanto, o tempo continua a ser uma dimensão que ocorre em todos os exames. Aplicações que acompanhem a evolução de pacientes durante o tratamento baseando-se em exames baseados em imagens também podem ser usadas como exemplo. O conjunto de características extraído das imagens pode ter diferente cardinalidade para cada imagem, como por exemplo no caso de representação da forma dos objetos encontrados nas imagens.

No entanto, não existem trabalhos associando tempo a dados em domínios métricos. Para poder tratar tempo em espaços métricos, é necessário criar um modelo de representação de dados Métrico-temporal, que permita a comparação de elementos métricos associados ao tempo. Na Seção 6.2 é apresentado aquele que é, no melhor de nosso conhecimento, o primeiro modelo Métrico-Temporal descrito na literatura, [Bueno et al., 2009b], o qual explora as propriedades dos espaços métricos para comparar dados complexos por similaridade, considerando informações temporais associadas a eles. Posteriormente, na Seção 6.3 são discutidas maneiras de analisar as trajetórias de dados métricos no decorrer do tempo. Em seguida, na Seção 6.3.4 são apresentadas novas possibilidades de análise visual da evolução temporal dos dados métricos. Por fim, na Seção 6.4 é apresentado uma modificação da técnica de balanceamento proposta para o Espaço Métrico-Temporal, que deu origem a um método para o balanceamento entre múltiplos conjuntos de características para representação de imagens.

6.2 Espaço Métrico-Temporal

Para tratar tempo em espaços métricos, foi criado um modelo de representação de dados Métrico-temporal, que permite comparar elementos métricos associados ao tempo [Bueno et al., 2009b].

Para isso, seja $\langle \mathbb{S}, d_s \rangle$ um espaço métrico, onde \mathbb{S} é um conjunto de dados complexos, não necessariamente dimensionais, em que todos os elementos que atendem às propriedades do domínio, e $d_s : \mathbb{S} \times \mathbb{S} \rightarrow \mathbb{R}^+$ é uma métrica que possibilite o cálculo da similaridade entre elementos desse domínio. Seja também $\langle \mathbb{T}, d_t \rangle$ outro espaço métrico, onde \mathbb{T} corresponde à todas as medidas de tempo que podem ser associadas aos elementos armazenados, e $d_t : \mathbb{T} \times \mathbb{T} \rightarrow \mathbb{R}^+$ é uma métrica que possibilite o cálculo de similaridade entre dois valores de tempo do domínio \mathbb{T} .

Então define-se um **Espaço Métrico-Temporal** como o par $\langle \mathbb{V}, d_v \rangle$, onde $\mathbb{V} = \mathbb{S} \times \mathbb{T}$ e $d_v : \mathbb{V} \times \mathbb{V} \rightarrow \mathbb{R}^+$ é uma métrica entre os elementos de um espaço métrico com informações temporais associadas, que é chamada de *função de distância métrico-temporal*, e agrupa as métricas d_s e d_t de uma maneira particular. Portanto, um espaço métrico-temporal é composto por uma componente métrica \mathbb{S} e uma componente temporal \mathbb{T} .

De acordo com essa definição, um elemento de um espaço métrico-temporal é basicamente a associação de informações temporais a um elemento de um espaço métrico. Por exemplo, pode-se representar um conjunto de elementos do mundo real como um conjunto de pares $\{\langle s_1, t_1 \rangle, \dots, \langle s_n, t_n \rangle\}$, $s_i \in \mathbb{S}$ e $t_i \in \mathbb{T}$, que são os estados dos elementos nos instantes de tempo correspondentes.

Dado um elemento $v_i \in \mathbb{V}$, tal que $v_i = \langle s_i, t_i \rangle | s_i \in \mathbb{S}, t_i \in \mathbb{T}$, s_i é chamada de **projeção métrica** de v_i , denotada como $s_i = \pi_s(v_i)$, e t_i é chamada de **projeção temporal** de v_i , denotada por $t_i = \pi_t(v_i)$. Dois elementos métrico-temporais $v_1, v_2 \in \mathbb{V}$ são o mesmo se e somente se $\pi_s(v_1) = \pi_s(v_2)$ e $\pi_t(v_1) = \pi_t(v_2)$.

Vale ressaltar aqui que tanto a componente métrica quanto a componente temporal pode ser composta pela agregação de vários outros espaços métricos, possibilitando assim a representação de múltiplas informações métricas e temporais.

6.2.1 Similaridade Métrico-Temporal

Uma função de distância métrico-temporal agrupa as métricas das componentes métrica e temporal do espaço métrico-temporal. Comumente em aplicações que exploram características de espaços métricos, cada métrica pode ser considerada como uma “caixa-preta”, definida por um especialista no domínio da aplicação.

Os modelos de dados temporais existentes não tratam as informações temporais como se fossem dados em espaços métricos, o que é proposto nesse trabalho. Neste trabalho foram sugeridas duas métricas básicas para medir a similaridade da componente temporal: uma para o caso em que os valores temporais representem instantes no tempo, e outra para o caso onde eles representem intervalos.

Métrica temporal para instantes

Seja o domínio temporal \mathbb{T}_i representando instantes de tempo, tal que $\mathbb{T}_i \subseteq \mathbb{R}^+$, a distância entre dois instantes $t_1, t_2 \in \mathbb{T}_i$ é dada por $d_{ti}(t_1, t_2) = |t_1 - t_2|$.

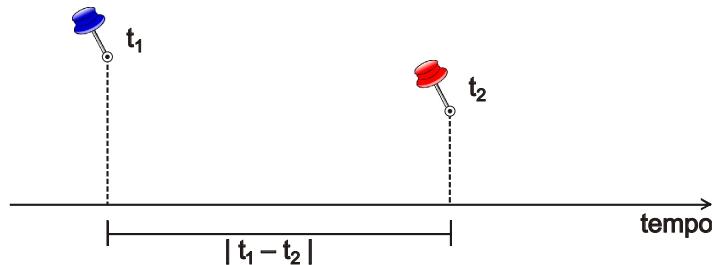


Figura 6.1: Métrica para instantes: $d_{ti}(t_1, t_2) = |t_1 - t_2|$.

A métrica d_{ti} , ilustrada na Figura 6.1, é adequada para comparar instantes de tempo, calculando a diferença absoluta entre eles. Esta função é essencialmente uma função de distância *Manhattan* (L_1) unidimensional (Seção 2.4.1), e satisfaz as propriedades de uma métrica.

Métrica temporal para intervalos

Seja \mathbb{T}_p um domínio temporal com dados que representam períodos de tempo, tal que $\forall t_i \in \mathbb{T}_p, t_i = [l_i, u_i] | l_i, u_i \in \mathbb{R}^+ \text{ e } l_i \leq u_i$, onde l_i e u_i são respectivamente os instantes que definem os limites inferior e superior do período de tempo.

Note que considerar apenas a diferença entre os limites de dois intervalos não é uma métrica, e considerar apenas o instante médio de um intervalo desconsidera informações referentes à duração do mesmo.

A métrica proposta (ilustrada na Figura 6.2) para comparar dois intervalos de tempo $t_1, t_2 \in \mathbb{T}_p$ é definida como:

$$d_{tp}(t_1, t_2) = |M(t_1) - M(t_2)| + |I(t_1) - I(t_2)| \quad (6.1)$$

onde $M(t_i) = l_i + \frac{(u_i - l_i)}{2}$ é o instante de tempo médio e $I(t_i) = u_i - l_i$ é a duração do período t_i .

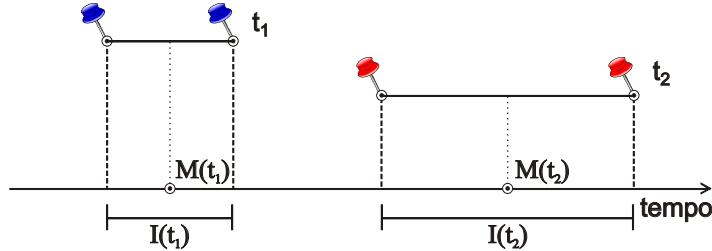


Figura 6.2: Métrica para intervalos : $d_{tp}(t_1, t_2) = |M(t_1) - M(t_2)| + |I(t_1) - I(t_2)|$.

A função d_{tp} claramente satisfaz as propriedades de simetria e não-negatividade, por se tratar de um somatório de valores absolutos.

Como a propriedade de desigualdade triangular também é satisfeita, como mostrado a seguir, d_{tp} é uma métrica, $\forall t_1, t_2, t_3 \in \mathbb{T}_p$

$$\begin{aligned} d_{tp}(t_1, t_2) &= |M(t_1) - M(t_2)| + |I(t_1) - I(t_2)| \\ &= |M(t_1) - M(t_2) + M(t_3) - M(t_3)| + |I(t_1) - I(t_2) + I(t_3) - I(t_3)| \\ &\leq |M(t_1) - M(t_3)| + |M(t_3) - M(t_2)| + |I(t_1) - I(t_3)| + |I(t_3) - I(t_2)| \\ &= d_{tp}(t_1, t_3) + d_{tp}(t_3, t_2) \end{aligned}$$

Métrica produto para espaços métrico-temporais

Para medir a similaridade entre elementos de um espaço métrico-temporal, é necessário compor adequadamente as funções de distância d_s e d_t , relativas às componentes métrica

\mathbb{S} e temporal \mathbb{T} . Como ambas satisfazem as propriedades de uma métrica por definição, a maneira natural de realizar tal composição é agregá-las em uma métrica produto [Searcoid, 2006].

Sejam d_s a métrica para a componente métrica e d_t a métrica para a componente temporal de um espaço métrico-temporal \mathbb{V} , a métrica produto a seguir gera um espaço métrico-temporal, com $v_i, v_j \in \mathbb{V}$ e $w_s, w_t \in \mathbb{R}^+$:

$$d_v(v_i, v_j) = w_s \cdot d_s(\pi_s(v_i), \pi_s(v_j)) + w_t \cdot d_t(\pi_t(v_i), \pi_t(v_j)) \quad (6.2)$$

onde w_s é o peso dado à componente métrica e w_t é o peso dado à componente temporal do espaço métrico-temporal.

Analizando as propriedades de uma métrica, pode-se ver que a não-negatividade e a simetria são satisfeitas diretamente por d_v , pois d_s e d_t são métricas. d_v também satisfaz a propriedade de desigualdade triangulares para qualquer $v_i, v_j, v_k \in \mathbb{V}$, como mostra-se a seguir:

$$\begin{aligned} d_v(v_i, v_j) &= w_s \cdot d_s(\pi_s(v_i), \pi_s(v_j)) + w_t \cdot d_t(\pi_t(v_i), \pi_t(v_j)) \\ &\leq w_s \cdot (d_s(\pi_s(v_i), \pi_s(v_k)) + d_s(\pi_s(v_k), \pi_s(v_j))) + \\ &\quad w_t \cdot (d_t(\pi_t(v_i), \pi_t(v_k)) + d_t(\pi_t(v_k), \pi_t(v_j))) \\ &= w_s \cdot d_s(\pi_s(v_i), \pi_s(v_k)) + w_t \cdot d_t(\pi_t(v_i), \pi_t(v_k)) + \\ &\quad w_s \cdot d_s(\pi_s(v_k), \pi_s(v_j)) + w_t \cdot d_t(\pi_t(v_k), \pi_t(v_j)) \\ &= d_v(v_i, v_k) + d_v(v_k, v_j) \end{aligned}$$

Portanto, a função de composição d_v é uma métrica. Note que uma métrica temporal, seja para instantes ou para intervalos, pode ser diretamente utilizada como d_t . Caso mais de uma informação temporal exista, como por exemplo tempo de transação e tempo de validade, tal como nos modelos tradicionais, todas as métricas relativas aos valores temporais podem ser agregadas em uma métrica produto.

Com isso, definir um modelo de representação de dados métrico-temporal passa a

ser o problema de definir um fator de escala entre as componentes métrica e temporal, definindo os pesos w_s e w_t de maneira que a métrica-produto represente adequadamente a similaridade entre elementos métrico-temporais.

Em outras palavras, é necessário responder adequadamente à seguinte questão: Para cada unidade de similaridade dada pela métrica $d_s(\pi_s(v_i), \pi_s(v_j))$, qual deve ser a equivalência em unidade de tempo dada pela métrica $d_t(\pi_t(v_i), \pi_t(v_j))$ para efeito de comparação entre dois elementos métrico-temporais $u_i, u_j \in \mathbb{U}$?

6.2.2 Fator de escala para a similaridade métrico-temporal

A idéia principal para definir os pesos w_s e w_t é identificar a contribuição relativa das componentes métrica e temporal de um espaço métrico-temporal para o cálculo final da similaridade

Uma propriedade conhecida na teoria dos espaços métricos é que subspaços métricos S com cardinalidade $|S| = s$ podem ser mapeados em um espaço vetorial \mathbb{R}^{s-1} de maneira que as distâncias no espaço original, calculadas pela métrica original, podem ser exatamente preservadas no espaço mapeado por uma função Minkowski de ordem q . Por outro lado, mapeamentos exatos não podem ser garantidos em espaços mapeados com menos de $s - 1$ dimensões. Entretanto, dependendo da distribuição particular do conjunto de dados, os erros nas distâncias no espaço mapeado com dimensionalidade $s^* < s - 1$ podem ser muito pequenos diminuindo o número de dimensões do espaço mapeado até um limite relacionado com a dimensão intrínseca do conjunto de dados. Neste trabalho assume-se que a dimensão fractal de correlação provê uma estimativa próxima à dimensão intrínseca dos conjuntos de dados, e portanto, do número mínimo de dimensões necessárias para representar o conjunto preservando as distâncias entre seus elementos.

A intenção é fazer com que os pesos w_s e w_t sejam proporcionais ao tamanho dos lados dos hiper cubos que cobrem os subespaços mapeados a partir das componentes métrica e temporal no espaço vetorial. No método desenvolvido não é necessário realmente mapear os dados, mas os conceitos envolvidos no mapeamento são utilizados para estimar os pesos.

Seja $V \subset \mathbb{V}$ um conjunto de dados tal que $S \subset \mathbb{S}$ contém as projeções métricas de todos os elementos de V e $T \subset \mathbb{T}$ contém as projeções temporais de todos os elementos

de V . Seja d_v a função que mede a similaridade sobre \mathbb{V} , conforme a Equação 6.2.

Define-se então p_s e p_t como **dimensionalidade métrica intrínseca** e **dimensionalidade temporal intrínseca** do conjunto V , calculados como o menor inteiro superior à dimensão fractal de correlação das componentes métrica e temporal, sendo $p_s = \lceil D_2(S) \rceil$ e $p_t = \lceil D_2(T) \rceil$. A **dimensionalidade métrica intrínseca** e a **dimensionalidade temporal intrínseca** podem ser aproximadas utilizando-se as técnicas *box-counting* ou *distance plot* (ver Seção 2.6) para calcular D_2 , dependendo se os conjuntos são multidimensionais ou não.

Dado um conjunto de dados num espaço métrico-temporal V com as projeções métrica e temporais correspondentes S e T , o **tamanho do lado métrico** ℓ_s de V e o **tamanho do lado temporal** ℓ_t de V são respectivamente os tamanhos dos lados dos hipercubos de dimensões p_s e p_t cobrindo os mapeamentos de S e T . Apesar de o tamanho real dos lados de ambos os hipercubos poderem ser calculados apenas realizando o mapeamento de ambos os subespaços, nesse trabalho foi utilizada uma aproximação desses valores.

O diâmetro de um hipercubo de dimensão p_s e tamanho de lado ℓ_s em um espaço vetorial com uma função de distância Minkowski de ordem q , cobrindo o mapeamento de um conjunto de dados S é dado por $diam(S) = \sqrt[q]{p_s} \cdot \ell_s$. Seja δ_{s_max} a distância máxima entre qualquer par de projeções em S . O valor de δ_{s_max} pode ser obtido diretamente do conjunto de dados original, e é comumente chamado de diâmetro do conjunto de dados. Assumindo-se que o diâmetro do conjunto de dados δ_{s_max} é igual ao $diam(S)$ do hipercubo que cobre o mapeamento de S , tem-se que o tamanho do lado desse hipercubo pode ser aproximado por $\ell_s = \frac{1}{\sqrt[q]{p_s}} \cdot \delta_{s_max}$.

A Figura 6.3 mostra a intuição dessa aproximação, onde o conjunto de dados $S \subset \mathbb{S}$, imerso no espaço métrico $\langle \mathbb{S}, d_s \rangle$, é mapeado para um espaço dimensional de ordem $p_s = 3$, com uma função de distância Euclidiana L_2 . Assim, o tamanho do lado do hipercubo que cobre o mapeamento de S é aproximado por $\ell_s = \frac{1}{\sqrt{3}} \cdot \delta_{s_max}$. O espaço métrico representado na figura é meramente ilustrativo.

O valor de δ_{s_max} pode ser avaliado pelo cálculo das distâncias entre todos os pares de elementos $s_i, s_j \in S$. Porém, esta operação é muito custosa, sendo quadrática

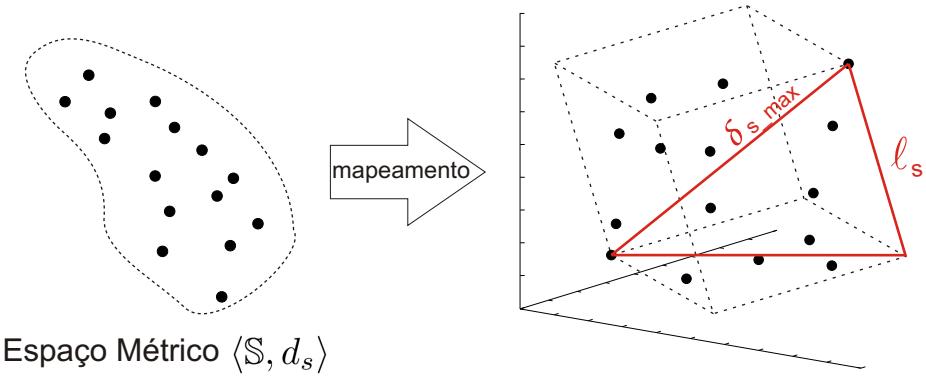


Figura 6.3: Espaço métrico mapeado em um espaço multidimensional \mathbb{R}^3 com uma função de distância L_2 . O tamanho do lado do cubo que cobre o conjunto de dados é dado por $\ell_s = \frac{1}{\sqrt{3}} \cdot \delta_{s_max}$.

com relação ao número de elementos em S . Entretanto, técnicas muito mais baratas computacionalmente podem ser aplicadas para estimar boas aproximações. Se um MAM é utilizado para indexar os dados, δ_{s_max} pode ser aproximado pelo diâmetro da região de cobertura do nó raiz da estrutura, por exemplo.

Apesar de todos os exemplos e discussões acima citarem ℓ_s , o mesmo procedimento é utilizado para estimar o valor de ℓ_t .

A partir desses valores, os pesos dados a cada componente são definidos como o inverso do tamanho do lado do hipercubo correspondente. Os pesos w_s e w_t das componentes métrica e temporal na distância métrico-temporal (segundo a Equação 6.2) são definidos respectivamente por $w_s = \frac{\sqrt[q]{p_s}}{\delta_{s_max}}$ e $w_t = \frac{\sqrt[q]{p_t}}{\delta_{t_max}}$, onde p_s , δ_{s_max} , p_t e δ_{t_max} podem ser medidos diretamente a partir de V .

No caso comum de uma única dimensão temporal, em que as informações temporais em T são valores simples de instantes de tempo, $[D_2(T)] = 1$. Assim, $\sqrt[q]{p_t} = 1$ para qualquer q , e $w_t = \frac{1}{t_{max}}$, onde t_{max} é a maior diferença absoluta de tempo em T .

Na próxima seção são apresentados os resultados de experimentos que confirmam que a maneira proposta para calcular os pesos w_s e w_t representam bem a contribuição relativa das componentes métrica e temporal de um espaço métrico-temporal para o cálculo final da similaridade.

6.2.3 Experimentos

Nesta seção são apresentados os resultados de experimentos realizados para avaliar o espaço métrico-temporal proposto. A implementação foi feita em C++ utilizando o MAM Slim-tree, disponível na biblioteca *Arboretum*, para indexar os dados. Os testes foram executados em um computador equipado com processador AMD Athlon 2.6 GHz, e 4Gb de memória RAM.

Metodologia e Conjuntos de dados

Os experimentos exploraram o cenário onde é analisado um conjunto de imagens de objetos que evoluem durante o tempo. As imagens são comparadas a partir de características extraídas por algoritmos de processamento de imagens. A cada imagem foi associado o tempo em que a imagem foi obtida, relativo ao tempo inicial em que o objeto começou a evoluir.

O conjunto de imagens utilizado foi o *Amsterdam Library of Object Images* (ALOI)¹ [Geusebroek et al., 2005]. Trata-se de uma coleção de imagens coloridas feitas a partir de mil pequenos objetos. A Figure 6.4 mostra um pequeno exemplo desse conjunto. Nessa biblioteca de imagens, cada objeto foi fotografado variando sistematicamente algum parâmetro da imagem. Nos experimentos apresentados aqui, foram selecionados conjuntos de imagens em que cada objeto foi fotografado em 72 ângulos de visão, cada um com rotação de 5 graus com relação ao anterior. Nos experimentos foi assumido que cada ângulo corresponde ao *timestamp* do objeto na imagem. Ou seja, a imagem de um objeto rotacionado em t graus corresponde a fotografia tirada após t unidades de tempo.

Cada uma das 72000 imagens foi processada por três extratores de características: momentos de Zernike, histogramas métricos e histogramas tradicionais. A partir das características obtidas por cada um dos extratores foram criados os conjuntos de dados apresentados na Tabela 6.1

O primeiro conjunto foi chamado *Zernike*, e as características que representam as imagens nesse conjunto são os 256 primeiros momentos de Zernike descrevendo as formas

¹disponível em <http://staff.science.uva.nl/~aloi/>



Figura 6.4: Exemplos de imagens da biblioteca de imagens ALOI.

contidas nas imagens [Khotanzad & Hong, 1990]. O segundo conjunto de dados foi chamado de *Histogramas métricos*. Este conjunto é adimensional e contém os histogramas métricos das imagens [Traina et al., 2003]. O terceiro e último conjunto de dados foi chamado de *Histogramas*, e contém histogramas de níveis de cinza das imagens. Este conjunto é constituído por 36 imagens de cada objeto (variando o ângulo de rotação de 0 a 175 graus).

A métrica utilizada para comparar as informações temporais (componente temporal) entre os elementos dos conjuntos de dados foi a diferença absoluta de tempo. Para a componente métrica, as métricas utilizadas para calcular a similaridade entre os elementos foram L_1 , *Metric Histogram Distance* (MHD) e L_1 , respectivamente para os conjunto *Zernike*, *Histogramas métricos* e *Histogramas*.

Tabela 6.1: Conjuntos de dados utilizados nos experimentos.

Nome	Tamanho	Dim.	Métrica	Descrição
<i>Zernike</i>	72000	256	L_1	Primeiros 256 momentos de Zernike das imagens do conjunto ALOI
<i>Histogramas métricos</i>	72000	—	MHD	Histogramas métricos das imagens do conjunto ALOI
<i>Histogramas</i>	36000	256	L_1	Histogramas de níveis de cinza das imagens do conjunto ALOI com rotações entre 0 e 175 graus

Para avaliar a qualidade dos resultados foram utilizados gráficos de precisão *versus*

revocação ($P \times R$) [Baeza-Yates & Ribeiro-Neto, 1999]. A *precisão* dos resultados de uma consulta é a fração dos elementos recuperados que são relevantes. Dessa forma, *precisão* = $\frac{|Ra|}{|A|}$, onde $|Ra|$ é o número de elementos relevantes recuperados e $|A|$ é o tamanho do conjunto de resposta da consulta. A *revocação* dos resultados de uma consulta é dada pela fração de elementos relevantes que foram recuperados. Portanto, *revocação* = $\frac{|Ra|}{|R|}$, onde $|Ra|$ é o número de elementos relevantes recuperados e $|R|$ é o número total de elementos relevantes que deveriam ser recuperados. De maneira simples, quanto mais perto do topo estiver uma curva de $P \times R$, melhores os resultados.

Curvas de $P \times R$ podem ser sumarizadas por um único valor numérico para permitir a avaliação e comparação de diferentes consultas de uma só vez. Neste trabalho foi utilizada a *precisão média* da curva, que é dada pela média dos valores de precisão em todos os níveis de revocação. Intuitivamente, a *precisão média* é a área sob uma curva de $P \times R$.

Para definir quais elementos são relevantes para uma determinada consulta, foi utilizado o seguinte critério: *As imagens mais relevantes para uma consulta são as imagens que sejam do mesmo objeto da imagem utilizada como elemento central da consulta, e que apresentem a menor diferença de tempo com relação ao elemento central da consulta.* A Figura 6.5 apresenta alguns exemplos para ilustrar essa idéia. Como pode-se ver na figura, o suporte para esse critério baseia-se na observação de que os objetos fotografados são significativamente diferentes entre si. O aspecto de um mesmo objeto fotografado em tempos próximos varia suavemente, o que levou a suposição de que suas imagens são semanticamente similares.

Contribuição ideal das componentes métrica e temporal

O objetivo desse primeiro conjunto de experimentos foi identificar o balanceamento ideal entre as componentes métrica e temporal no cálculo da similaridade entre os elementos. Nos experimentos, foi utilizada a função de distância Minkowski com $q = 1$ para os espaços mapeados, por ser computacionalmente mais barata e experimentalmente produziu resultados tão bons quanto as outras. Dessa forma, $w_s = p_s / \delta_{s_max}$ e $w_t = p_t / \delta_{t_max}$. Os valores de p_s e p_t foram estimados utilizando a técnica *distance plot* para todos os conjuntos de dados. Os valores δ_{s_max} e δ_{t_max} também foram obtidos pela

6.2 Espaço Métrico-Temporal

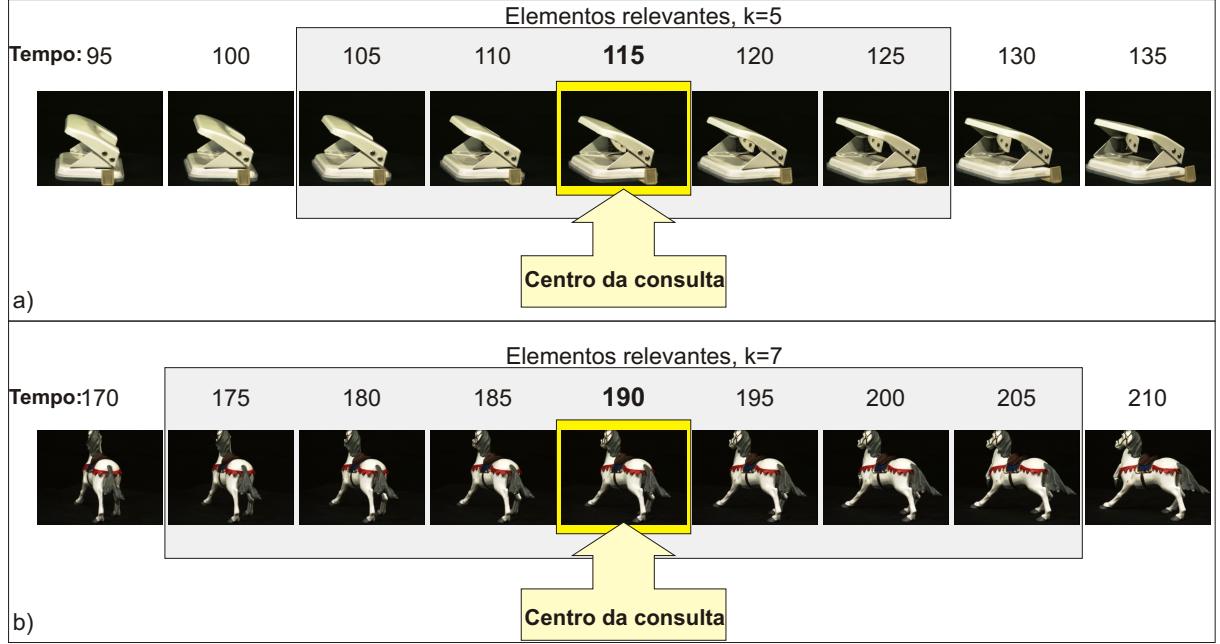


Figura 6.5: Elementos relevantes para diferentes elementos centrais de consulta. Resultados para 5-NNq (a) e 7-NNq (b).

execução da técnica *distance plot*, que computa as distâncias entre todos os elementos.

A componente temporal é sempre a mesma para os três conjuntos de dados avaliados. A Figura 6.6 mostra o gráfico gerado pela técnica *distance plot*. O valor da dimensão fractal de correlação para o espaço métrico $\langle \mathbb{T}, d_t \rangle$ foi $D_2(T) = 0.98$, e dessa forma, $w_t = \lceil D_2(T) \rceil = 1$. Destaca-se que apesar de no conjunto *Histogramas* serem utilizados apenas metade dos elementos, o valor estimado foi praticamente o mesmo, o que explica-se pelo fato dos dois conjuntos manterem a mesma distribuição dos dados.

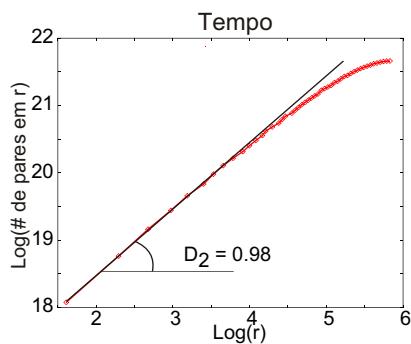


Figura 6.6: *Distance plot* da componente temporal T .

Os gráficos gerados pela técnica *Distance plot* para as componentes métricas de todos os conjuntos de dados são mostrados na Figura 6.7. Como pode-se ver na figura, o ajuste

da linha de inclinação pode ser realizado facilmente para o conjunto *Histogramas métricos*, porém não com os outros conjuntos. A estratégia de ajuste da linha de inclinação deve ser adequada ao objetivo para o qual o valor da dimensão fractal será utilizado. Nesse caso, o interesse é responder consultas por similaridade em um espaço métrico-temporal. Nesse contexto, consultas por vizinhos mais próximos, k -NN, tipicamente empregam valores pequenos de k , e as consultas por abrangência devem retornar um número também não muito grande de elementos. Portanto, assumiu-se nesses experimentos que valores de k entre 5 e 100 são adequados para a maioria das aplicações. Assim, o interesse é ajustar a linha de inclinação na região do gráfico de *Distance plot* que refere-se ao intervalo entre 5 e 100 elementos.

A equação $PC(r) = K_p \cdot r^D$ (ver Seção 2.6) utiliza o número de pares $PC(r)$ com uma distância r , porém, aqui o interesse recai sobre o número k de elementos envolvidos. Portanto é necessário converter o “número de elementos” em “número de pares” dentro de um limite de distância. O número de pares em um subconjunto de k elementos, contando cada par somente uma vez, é dado por $Pares(k) = k(k - 1)/2$. Portanto, a região de interesse para o ajuste da linha de inclinação é limitada pelos valores $\log(Pairs(5)) = 2.302$ e $\log(Pairs(100)) = 8.507$, como mostrado na Figura 6.7. Os valores de dimensão fractal de correlação calculados a partir deste procedimento foram 8.9, 15.1 e 3.2, respectivamente para os conjuntos Histograma, Zernike e *Histogramas métricos*, o que levou à definição dos valores das **dimensionalidades métricas intrínsecas** p_s respectivamente como 9, 16 e 4.

Para verificar se os pesos estimados eram adequados, foram executadas consultas sobre cada conjunto variando o valor de p_s . Na Figura 6.8 são mostradas as *precisões médias* obtidas em gráficos de $P \times R$ com variação do valor de p_s para os três conjuntos de dados. Vale lembrar aqui que cada ponto nesse gráfico representa a *precisão média* de uma curva de $P \times R$, calculada pela média de 500 consultas k -NN, onde o centro de consulta foi randomicamente selecionado dentre os elementos indexados.

Na figura, os valores $p_s = 9, 16$ e 4 , estimados anteriormente para cada conjunto de dados, estão em destaque. Como pode-se ver, os valores estimados de p_s sempre levaram

6.2 Espaço Métrico-Temporal

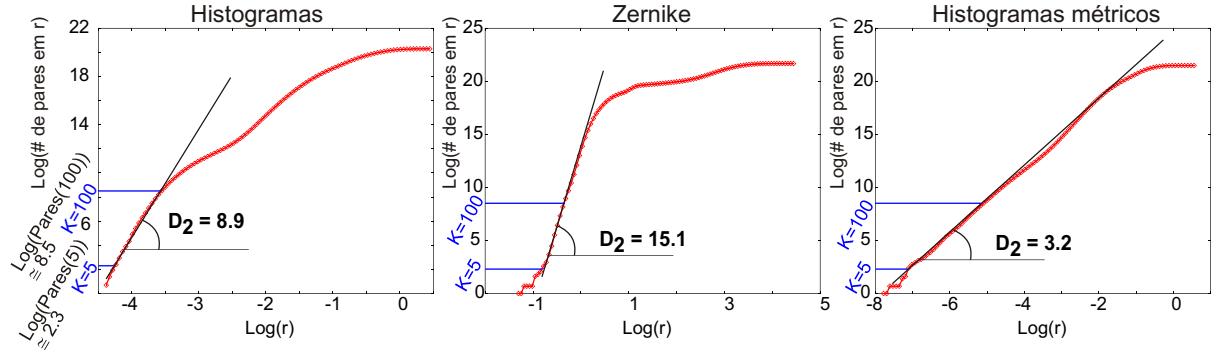


Figura 6.7: *Distance plots* das componentes métricas S dos conjuntos: (a) *Histogramas* (b) *Zernike* (c) *Histogramas métricos*.

aos melhores, ou muito próximo dos melhores resultados de precisão para as consultas.

Outro ponto a ser avaliado é se a inclusão da componente temporal no cálculo da similaridade realmente auxilia no aumento de precisão das consultas. Para isso, foram comparados os resultados de consultas utilizando o espaço métrico-temporal com os resultados de consultas utilizando apenas as componentes métricas dos conjuntos de dados. Essa comparação pode ser vista na Figura 6.8, em que a linha tracejada apresenta o resultado das mesmas consultas utilizando apenas a componente métrica (vale lembrar aqui que a variação dos valores de p_s não afeta essas consultas). Como pode-se ver, a utilização da componente temporal sempre aumentou a precisão das respostas, mesmo para valores ruins de p_s .

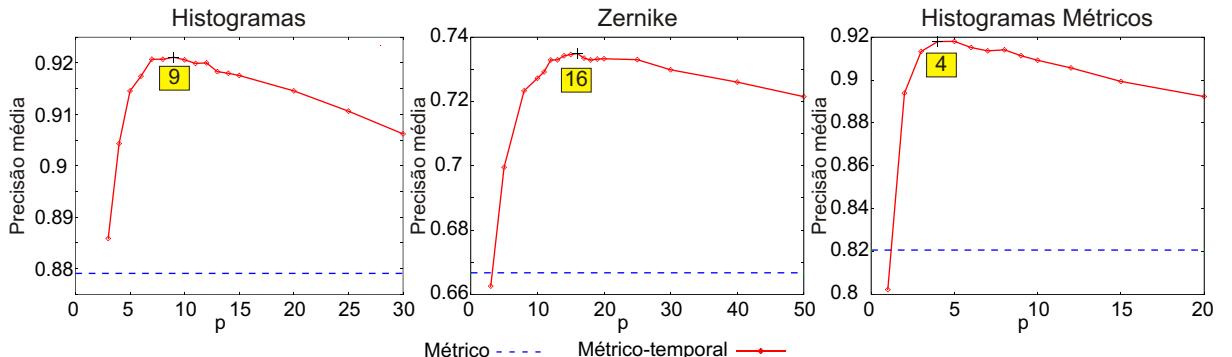


Figura 6.8: Precisões médias de consultas k -NN variando o valor de p_s para os três conjuntos de dados. Os pontos mostram as precisões médias relativas às consultas realizadas sobre o espaço métrico-temporal, e a linha tracejada mostra as precisões médias relativas às consultas realizadas somente sobre a componente métrica.

Na Figura 6.9 são mostrados gráficos de $P \times R$ para consultas aos 10 vizinhos mais próximos, utilizando os pesos com os valores estimados pelo método proposto: $p_s = 9, 16$ e 4 respectivamente para os conjuntos *Histogramas*, *Zernike* e *Histogramas métricos*. Os resultados são comparados com aqueles obtidos utilizando-se apenas as componentes métricas dos conjuntos de dados. Os gráficos mostram que o espaço métrico-temporal apresenta melhores resultados de precisão em todos os níveis de revocação para todos os conjuntos de dados. A *precisão média* das consultas aos vizinhos mais próximos aumentou de 0.88 para 0.92 com o conjunto *Histogramas*, de 0.67 para 0.74 com o conjunto *Zernike*, e de 0.82 para 0.92 com o conjunto *Histogramas métricos*.

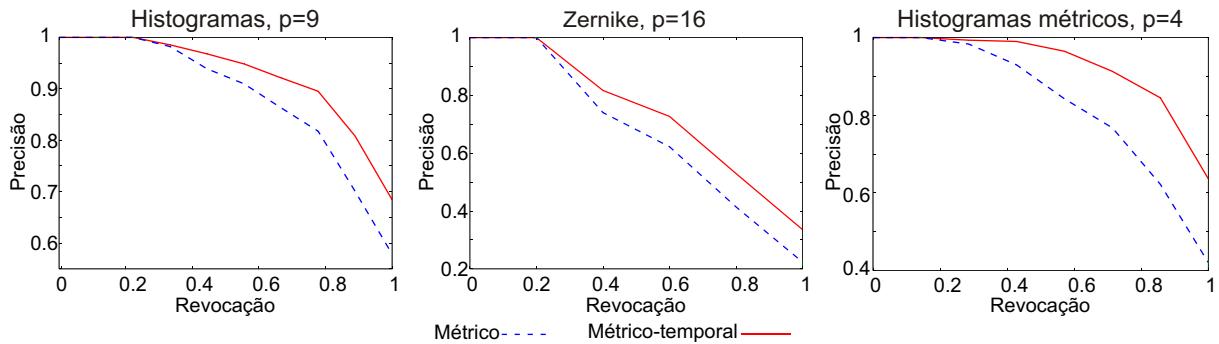


Figura 6.9: Comparação dos resultados de consultas 10-NN utilizando o espaço métrico-temporal com os pesos estimados com resultados obtidos com a utilização apenas da componente métrica.

É importante mencionar que o tempo necessário para responder as consultas usando o espaço métrico-temporal foi praticamente o mesmo do necessário para responder as mesmas consultas utilizando apenas as componentes métricas, pois o custo adicional relativo à computação da componente temporal foi irrelevante comparado aos custos das métricas originais. O único custo adicional significativo na utilização do espaço métrico-temporal refere-se àquele necessário para calcular os valores de dimensão fractal de correlação dos conjuntos, necessários para a definição de p_s e p_t . Entretanto, a dimensão fractal de correlação pode ser calculada utilizando-se apenas uma pequena amostra do conjunto de dados original, pois é invariante à amostragem imparcial (*unbiased sampling*). Nos experimentos foram testadas várias taxas de amostragem, e comprovando o que é descrito na literatura, para os conjuntos de dados utilizados, os valores de p_s e p_t

6.2 Espaço Métrico-Temporal

calculados com amostragens de até 1% dos conjuntos originais foram os mesmos que os calculados utilizando todos os elementos dos conjuntos. Na Figura 6.10 é mostrado um exemplo do cálculo da dimensão fractal de correlação utilizando tamanhos variados de amostra do conjunto de dados *Histogramas métricos* para confecção dos gráficos de *distance plot*. Como pode-se ver, as linhas de ajuste de inclinação com os diversos tamanho de amostra são paralelas, tendo todas a mesma inclinação, resultando todas no mesmo valor da dimensão fractal de correlação.

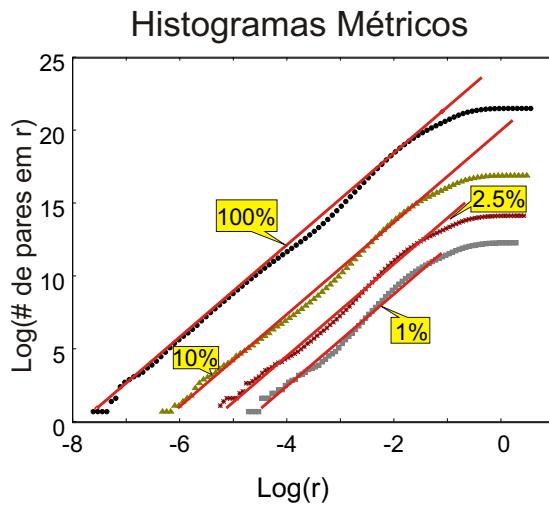


Figura 6.10: *Distance plots* da componente métrica do conjunto *Histogramas métricos* calculados com diferentes taxas de amostragem sobre o conjunto original.

Na Tabela 6.2 são mostrados os tempos necessários para definir os valores de p_s e p_t utilizados nos experimentos, utilizando amostras de 2.5% dos conjuntos originais. É importante lembrar que esses valores são calculados apenas uma vez para cada conjunto de dados.

Tabela 6.2: Tempo de processamento para obtenção de p_s e p_t .

Nome	Tempo
Zernike (p_s)	18s
<i>Histogramas métricos</i> (p_s)	14s
<i>Histogramas</i> (p_s)	5s
Tempo (p_t)	0.7s

6.3 Trajetórias de dados métrico-temporais

Na seção anterior avaliou-se o modelo Métrico-temporal proposto, mostrando que a incorporação do tempo nas comparações por similaridade aumentou a precisão das respostas. Ou seja, trata-se de um modelo direcionado para aplicações em que o tempo influí na similaridade entre os elementos.

Nesta seção o foco são aplicações em que tem-se o interesse de analisar o comportamento evolutivo dos dados métricos no decorrer do tempo, analisando as trajetórias dos dados métrico-temporais. Para que este tipo de análise possa ser feito, deve existir a possibilidade de definição de uma relação de ordem nos dados métricos com relação a informação temporal. No modelo proposto não existe a intenção de representar o tempo absoluto em espaços métricos, mas apenas o efeito do tempo sobre os dados métricos armazenados.

Para a análise das trajetórias dos elementos métrico-temporais, foi utilizada uma heurística para imersão do espaço métrico-temporal em um espaço dimensional, com realização de consultas sobre este novo espaço mapeado, estimando os resultados em tempo futuro, passado e intermediário, com relação aos elementos indexados.

6.3.1 Mapeamento do espaço métrico-temporal

Em um espaço multidimensional pode-se facilmente estimar trajetórias de elementos de dados baseando-se nos valores de cada atributo (em cada dimensão) e nos valores de tempo. Por exemplo, na evolução de um elemento em um espaço bi-dimensional, pode-se estimar a posição do elemento em um tempo futuro, baseando-se em suas duas últimas posições. Porém, em espaços métricos não existe o conceito de dimensões, e não pode-se analisar atributos individualmente. As únicas informações disponíveis sobre os elementos métricos são as distâncias entre eles. Considerando um elemento métrico-temporal, como estimar seu estado futuro baseando-se em seus últimos estados? Como não existe o conceito de dimensionalidade em espaços métricos, os espaços métrico-temporais podem ser mapeados para espaços dimensionais, e a partir daí pode-se utilizar todo o arcabouço teórico dos espaços dimensionais para análise da trajetória dos dados métrico-

temporais.

Neste trabalho foi utilizado o algoritmo FastMap [Faloutsos & Lin, 1995] para realização desse mapeamento. O algoritmo FastMap trata-se de uma heurística para mapear os elementos métricos como pontos em um espaço k-dimensional, visando preservar as distâncias (dissimilaridades) entre eles. A cada iteração do algoritmo são utilizados dois elementos escolhidos como pivôs, e os elementos do conjunto são mapeados pela projeção dos elementos sobre a linha entre esses pivôs, considerada como um dos eixos do espaço mapeado.

6.3.2 Consultas aproximadas no espaço mapeado

No espaço multidimensional mapeado, pode-se analisar o comportamento das trajetórias dos elementos métrico-temporais. Como não existe a possibilidade de estimar o estado de um elemento no espaço métrico-temporal, essa estimativa é feita no espaço mapeado, e a comparação entre dois elementos ser feita de maneira puramente relativa. O método proposto utiliza a interpolação/extrapolação de valores de acordo com a dimensão temporal dos dados para as estimativas no espaço mapeado.

Porém, na grande maioria dos casos não existe a possibilidade de percorrer o caminho inverso ao mapeamento: a partir de um elemento estimado no espaço mapeado, não pode-se “reconstruir” tal elemento no espaço métrico-temporal original. Então, a partir da estimativa da ‘posição’ do elemento no espaço mapeado, são realizadas consultas por similaridade nesse espaço para encontrar os elementos que estejam próximos ao elemento estimado. Dessa maneira, os elementos recuperados pela consulta são os elementos indexados que são os mais próximos do elemento estimado. A distância entre o elemento estimado e os resultados da consulta podem ser utilizados para avaliar a qualidade das respostas.

Aqui é usada como exemplo uma aplicação de acompanhamento de pacientes baseada em exames de imagens médicas. Na Figura 6.11 é mostrado um espaço bidimensional representando o espaço mapeado, onde as várias imagens de pacientes são representadas como pontos. O paciente P_A tem, no exemplo, duas imagens indexadas, uma antes de iniciar o tratamento ($P_A, t = 0$) e outra imagem com 12 meses de tratamento ($P_A, t = 12$).

A partir dessas duas imagens indexadas, na figura são exemplificadas duas consultas. O objetivo da consulta ilustrada na Figura 6.11(a) é estimar o estado do paciente quando ele estiver com 15 meses de tratamento. Para responder a esta consulta, estima-se no espaço mapeado qual seria a “posição” do paciente com 15 meses de tratamento, a partir dos dados indexado. Realiza-se então a consulta aos vizinhos mais próximos ao elemento estimado no espaço mapeado, e retorna-se as imagens associadas a esses vizinhos. Já o objetivo da consulta ilustrada na Figura 6.11(b) é estimar como seria o estado do paciente no decorrer do tratamento, em um tempo intermediário à realização dos exames, com 6 meses de tratamento. Da mesma forma que no exemplo anterior, são retornadas as imagens mais similares obtidas pela realização da consulta no espaço mapeado, utilizando-se a estimativa do estado do paciente neste espaço.

6.3.3 Experimentos

Como na seção anterior, os experimentos exploraram o cenário onde um conjunto de imagens de objetos que evoluem durante o tempo é analisado, e a cada imagem foi associado o tempo em que a imagem foi obtida, relativo ao tempo inicial em que o objeto começo a evoluir. Foi utilizado o mesmo conjunto de imagens (*Amsterdam Library of Object Images (ALOI)*) [Geusebroek et al., 2005], porém foram utilizados apenas 36 imagens para cada objeto, variando o ângulo de rotação de 0 a 175 graus. Cada uma das imagens foi processada por dois extratores de características: histogramas de níveis de cinza e momentos de Zernike [Khotanzad & Hong, 1990], e foram criados os conjuntos de dados apresentados na Tabela 6.3

O primeiro conjunto de dados, chamado de *Histogramas*, contém histogramas de níveis de cinza das imagens, e o segundo conjunto, chamado *Zernike*, descreve as formas contidas nas imagens. Também como nos experimentos da seção anterior, a métrica utilizada para comparar a componente temporal foi a diferença absoluta de tempo, e a métrica utilizada para a componente métrica foi L_1 para ambos os conjuntos.

Os dados foram mapeados para espaços dimensionais, de acordo com a dimensão fractal de correlação dos conjuntos de dados calculados na Seção 6.2.3. Os valores calculados foram 8.9 para o conjunto *Histogramas*, 15.1 para o conjunto *Zernike*, e 0.98

6.3 Trajetórias de dados métrico-temporais

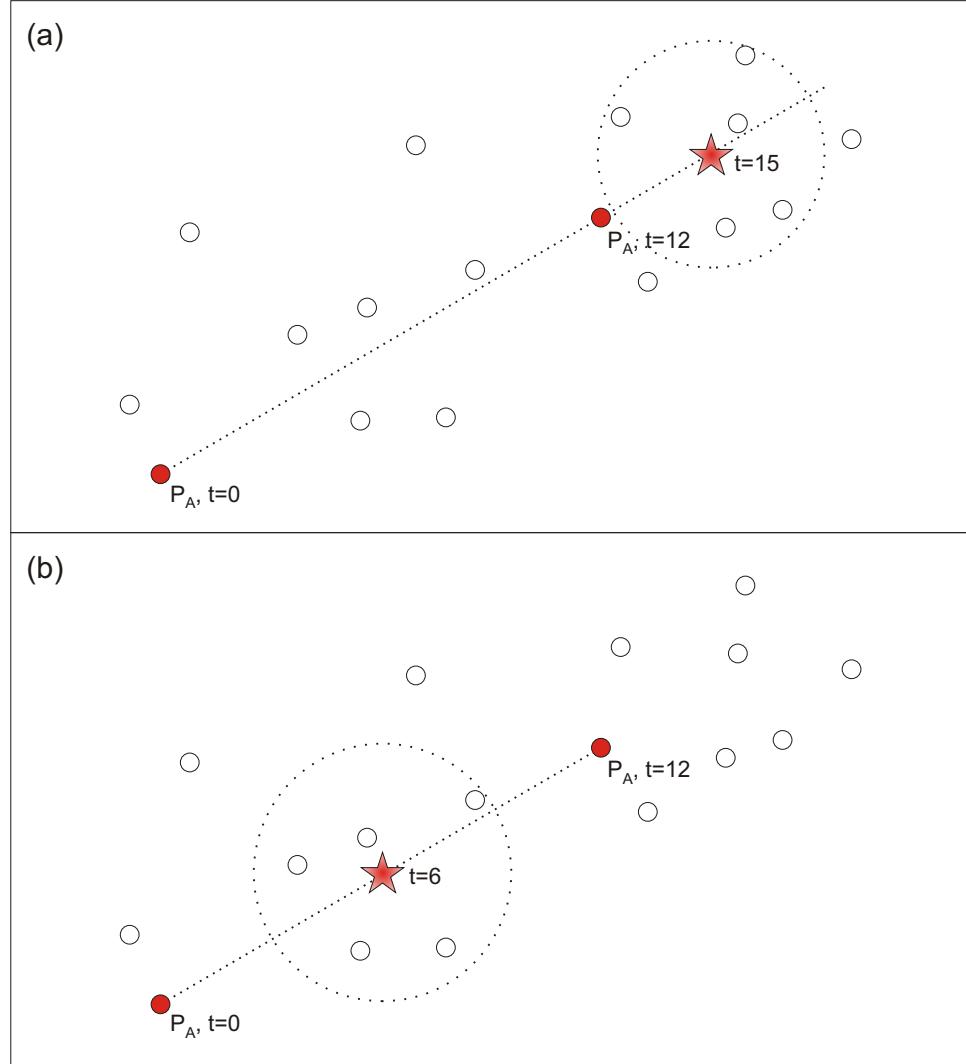


Figura 6.11: Exemplos de consultas no espaço mapeado: (a) estimativa do estado do paciente P_A com 15 meses de tratamento, (b) estimativa do estado do paciente P_A com 6 meses de tratamento.

Tabela 6.3: Conjuntos de dados utilizados nos experimentos.

Name	Size	Dim.	Metric	Description
<i>Histogramas</i>	36000	256	L_1	Histogramas de níveis de cinza das imagens do conjunto ALOI com rotações entre 0 e 175 graus
<i>Zernike</i>	36000	256	L_1	Primeiros 256 momentos de Zernike das imagens do conjunto ALOI com rotações entre 0 e 175 graus

para a componente temporal. Dessa forma, como discutido anteriormente, neste trabalho assume-se que a dimensão fractal de correlação provê uma estimativa próxima ao número mínimo de dimensões necessárias para representar o conjunto de dados preservando as distâncias entre seus elementos. Portanto, o conjunto *Histogramas* foi mapeado para um espaço de 10 dimensões, e o conjunto *Zernike* foi mapeado em um espaço de 17 dimensões.

Para avaliar os resultados das consultas realizadas no espaço mapeado, foram realizados vários tipos diferentes de consulta, apresentadas na Figura 6.12. Em todas as consultas foram utilizados dois elementos métrico-temporais referentes a uma mesma imagem (duas instâncias do mesmo objeto em tempos diferentes). Pela interpolação dos valores em cada uma das dimensões do espaço dimensional poderia-se estimar a posição do elemento de consulta em qualquer valor intermediário. Porém, nos experimentos, restringiu-se essa posição temporal do elemento central da consulta estimado somente aos valores presentes na base, para que os resultados pudessem ser comparados com os resultados das consultas realizadas a partir desses objetos.

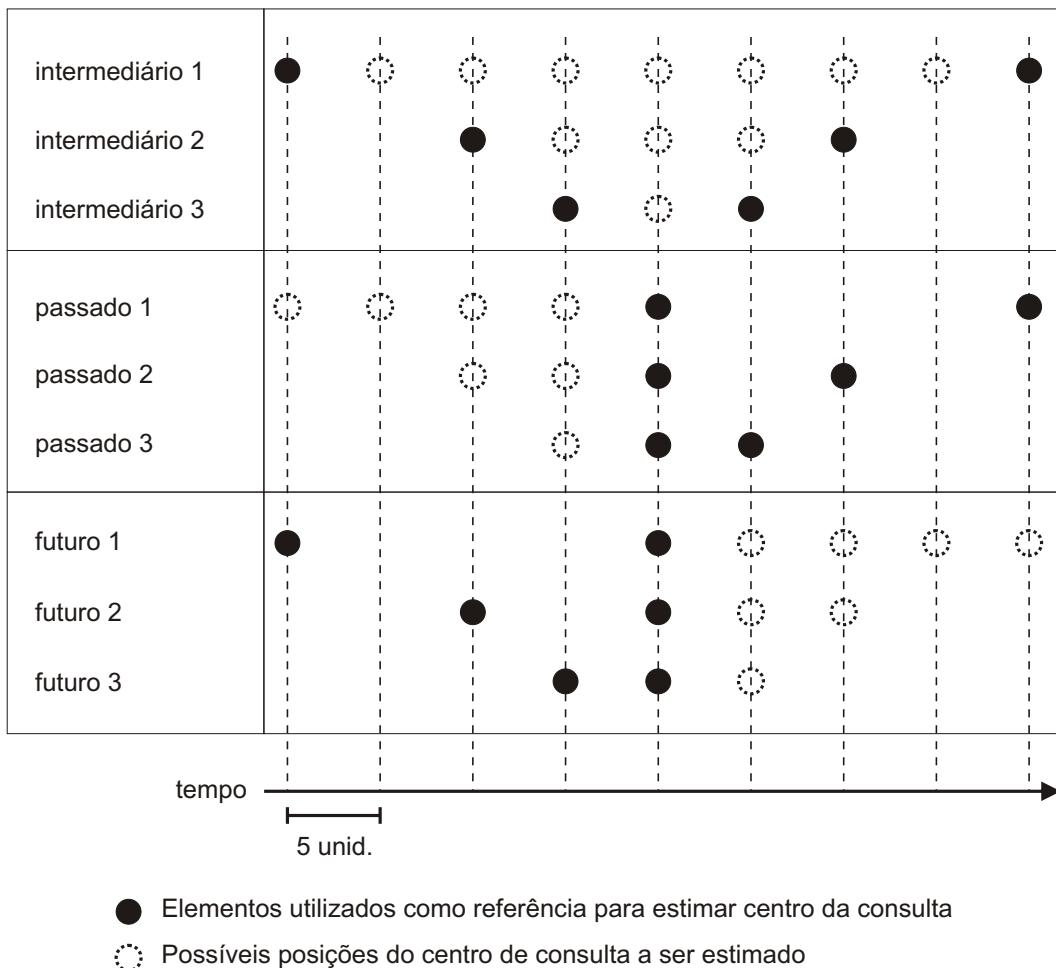


Figura 6.12: Tipos de consultas realizadas nos experimentos.

São realizadas então consultas aos 10 vizinhos mais próximos no espaço mapeado, utilizando-se o centro de consulta estimado a partir de dois valores presentes na base. Por exemplo, no tipo de consulta *intermediário 2*, são utilizadas duas instâncias de um mesmo

6.3 Trajetórias de dados métrico-temporais

objeto separadas por 20 unidades de tempo, e escolhe-se aleatoriamente uma das 3 posições intermediárias para estimar o elemento central da consulta. No espaço mapeado, a partir das instâncias de um dado objeto, chamado aqui de *objeto X* nos tempos 10 e 30, estima-se o *objeto X* no tempo 20, e realiza-se a consulta $10 - nn$ utilizando esse objeto estimado como centro da consulta. Os resultados dessa consulta são então comparados ao resultado da consulta aos 10 vizinhos mais próximos realizada no espaço métrico-temporal original, tendo como centro da consulta o objeto original no tempo do objeto estimado. Voltando ao exemplo, realiza-se uma consulta $10 - NN$ no espaço métrico-temporal, utilizando como centro da consulta o *objeto X* com tempo 20. Os resultados dessa última consulta são considerados como corretos, e os elementos retornados são considerados relevantes.

Na Figura 6.13 são mostrados gráficos de precisão *versus* revocação ($P \times R$) para os diversos tipos de consulta adotados nos experimentos apresentados anteriormente, utilizando o conjunto *Histogramas*.

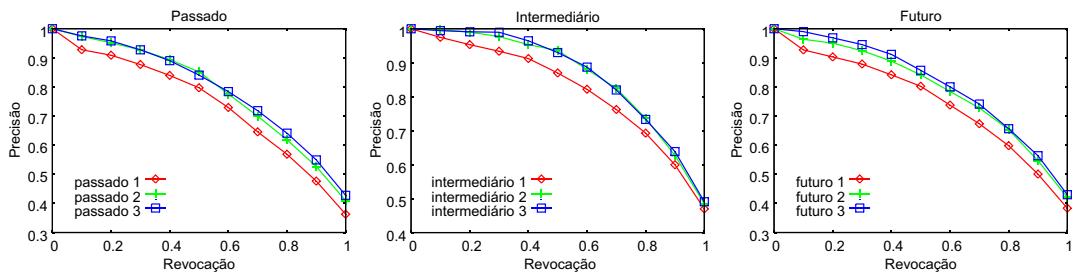


Figura 6.13: Consultas $10 - NN$ realizadas sobre o espaço mapeado utilizando o conjunto *Histogramas*.

Como pode ser visto na Figura 6.13, quanto maior a proximidade entre o elemento central da consulta estimado e os elementos utilizados como referência para a estimativa, melhores são os resultados. Todas as curvas de $(P \times R)$ apresentaram *precisão média* superior a 74%, chegando até 86%.

Na Figura 6.14 são mostrados gráficos de $(P \times R)$ utilizando o conjunto *Zernike*.

Assim como nos experimentos com o conjunto *Histogramas*, a maior proximidade entre o elemento central estimado e os elementos utilizados como referência para a estimativa resulta em melhores resultados. Porém, a precisão das respostas foi significativamente menor com este conjunto. A *precisão média* das curvas de $(P \times R)$ ficou entre 26% e 42%.

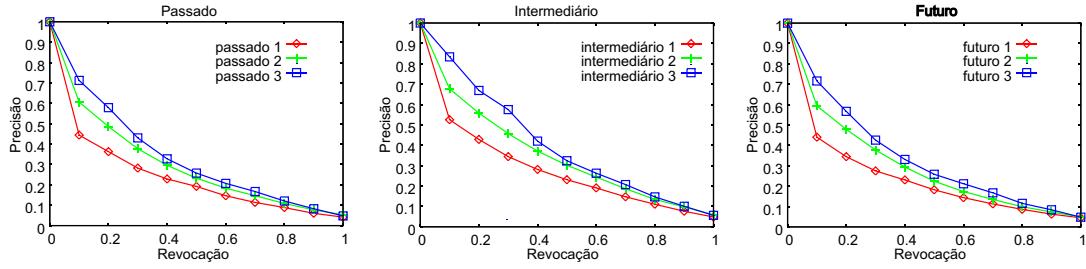


Figura 6.14: Consultas 10 – NN realizadas sobre o espaço mapeado utilizando o conjunto *Zernike*.

Nos experimentos anteriores, apresentados nas Figuras 6.14 e 6.13, o objetivo foi analisar a qualidade das estimativas dos elementos centrais da consulta no espaço mapeado. Porém, outro fator que afeta a qualidade das estimativas refere-se à qualidade do mapeamento obtido, objeto de estudo no experimento a seguir.

Para analisar a qualidade dos mapeamentos foram utilizadas curvas de $P \times R$. Escolhe-se aleatoriamente um elemento métrico-temporal e realiza-se uma consulta aos 10 vizinhos mais próximos no espaço métrico-temporal original. O resultado dessa consulta é considerado correto, e os elementos retornados são considerados relevantes. Então realiza-se a mesma consulta $10 - nn$ no espaço mapeado, utilizando o mesmo elemento central da consulta, agora mapeado no espaço dimensional. No caso de um mapeamento perfeito, as curvas apresentariam precisão total para todos os níveis de revocação. Ou seja, em um mapeamento perfeito todas as distâncias entre os elementos seriam mantidas, e as respostas obtidas no espaço mapeado seriam as mesmas obtidas no espaço original.

Na Figura 6.15 são mostradas as curvas de $P \times R$ obtidas dessa forma, identificadas no gráfico com o nome de *exato*. Para facilitar as comparações com as consultas realizadas com os elementos centrais da consulta estimados, são apresentadas novamente nos gráficos as curvas de $P \times R$ de melhor desempenho dos experimentos anteriores.

Como pode-se ver nos gráficos da Figura 6.15, os melhores resultados obtidos anteriormente (consultas *intermediário 3*) aproximam-se muito dos resultados das consultas exatas. Ou seja, os elementos centrais de consulta estimados no espaço mapeado foram muito próximos dos mapeamentos exatos de tais elementos. Com o conjunto *Histogramas*, que apresentou melhores resultados do mapeamento, praticamente não houve diferenças entre os elementos estimados e reais. Portanto, as baixas precisões

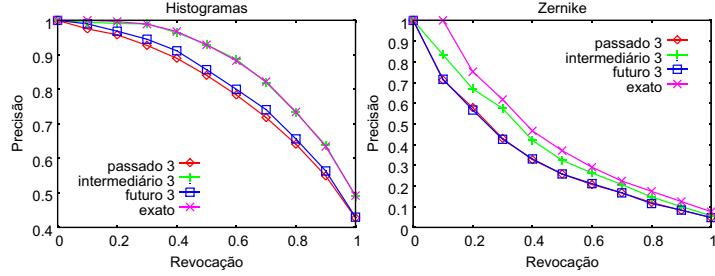


Figura 6.15: Avaliação da qualidade dos mapeamentos.

obtidas nas consultas com o conjunto *Zernike* podem ser justificadas pela baixa qualidade do mapeamento do conjunto.

Os resultados preliminares discutidos nessa seção tem o intuito de demonstrar a possibilidade da utilização da trajetória de elementos métrico-temporais para realização de consultas por similaridade aproximadas, inferindo o “estado” dos objetos em tempos passado, intermediário e futuro, baseadas nos tempos disponíveis relacionados ao objeto. A precisão dos resultados pode ainda ser muito aprimorada em trabalhos futuros. Nos experimentos realizados foram utilizados apenas duas instâncias (com dois tempos diferentes) dos elementos indexados para estimar o centro da consulta em um tempo diferente. Pode-se, no entanto, utilizar todas as instâncias disponíveis de determinado elemento para a estimativa. Outra atividade em um futuro trabalho seria estudar outras opções de mapeamento, visando melhorar a qualidade do espaço mapeado.

6.3.4 Visualização de dados métrico-temporais

Na seção anterior foi descrito o processo de imersão do espaço métrico-temporal em um espaço dimensional para possibilitar a análise das trajetórias dos elementos métricos. Nesta seção, decreve-se o processo semelhante para possibilitar a análise visual da evolução temporal dos dados métricos. Porém, ao invés de utilizar a dimensão intrínseca dos dados para definir o número de dimensões do espaço mapeado, os dados são mapeados em espaços tridimensionais.

Duas novas possibilidades de visualização de dados foram adicionadas à ferramenta FastMapDB [Traina et al., 2001, Barioni et al., 2002, Paterlini et al., 2005, Fedel et al., 2006]. Além da possibilidade de visualização de dados segundo o modelo

métrico-temporal, outra funcionalidade de visualização de dados já presente na ferramenta foi modificada.

A ferramenta de análise visual FastMapDB utiliza o algoritmo Fastmap [Faloutsos & Lin, 1995] para mapear elementos de dados complexos em um espaço tridimensional, possibilitando assim sua visualização e auxiliando no processo de análise dos dados indexados em SGBDs. A ferramenta apresenta uma interface gráfica com suporte a seleção de atributos, filtros para seleção de dados e manipulação das funções de distância. Depois do mapeamento dos dados, a ferramenta propicia ao usuário várias ferramentas interativas que possibilitam operações de rotação, escala e translação das imagens, além de ferramentas para seleção visual dos elementos. Além disso, a ferramenta também tem a funcionalidade de estimar a dimensão fractal de correlação dos dados, facilitando assim a definição dos pesos para dados no modelo métrico-temporal 6.2.2.

Na versão anterior da ferramenta, existia a funcionalidade para visualização da evolução temporal de dados [Razente, 2004]. Segundo esta técnica, para cada valor de tempo presente na base de dados, todos os elementos presentes são mapeados em um espaço bidimensional, dando origem a um plano (x, y). Então, esses planos são dispostos de maneira equidistante sobre o eixo z . Tal técnica foi modificada para que tais planos sejam distribuídos sobre o eixo z de maneira a representar a distância temporal entre os elementos, e não uniformemente.

Porém, a técnica discutida no parágrafo anterior define que dois dos eixos do espaço a ser visualizado representam a ‘componente métrica’ dos dados, e um eixo representa sua ‘componente temporal’. Na outra funcionalidade de visualização incluída na ferramenta FastMapDB, utilizou-se o método proposto juntamente com o modelo métrico-temporal para identificar o balanceamento ideal entre as componentes métrica e temporal, e então esse espaço métrico-temporal foi mapeado para um espaço tridimensional. Caso exista a possibilidade de definição de uma relação de ordem nos dados métricos com relação a informação temporal, a ferramenta possibilita que instâncias consecutivas sejam conectadas por segmentos de linha.

Na Figura 6.16 são mostrados exemplos de visualização utilizando histogramas

6.3 Trajetórias de dados métrico-temporais

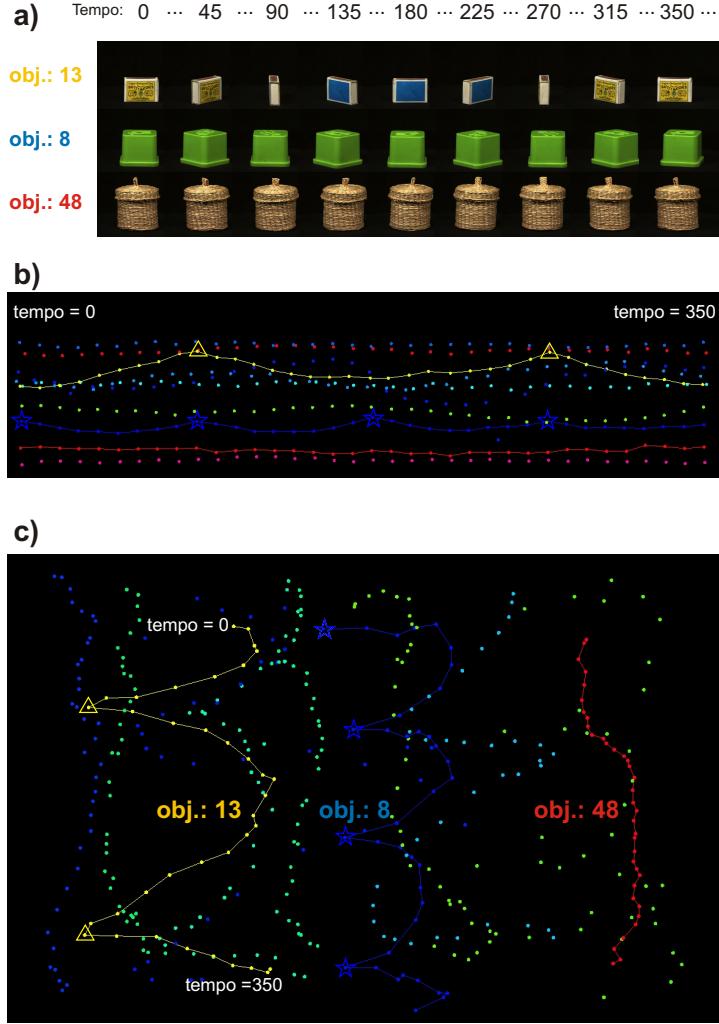


Figura 6.16: Exemplos de visualização: a) objetos visualizados; b) visualização bidimensional em planos paralelos; c) visualização utilizando o modelo métrico-temporal.

tradicionais das imagens do conjunto ALOI (ver conjunto *Histogramas* na Tabela 6.1), com variação temporal em intervalos de 10 unidades, resultando portanto em 36 instâncias para cada imagem. Em 6.16b) é mostrada a visualização bidimensional em planos paralelos, e na Figura 6.16c) é apresentada a visualização utilizando o modelo métrico-temporal. Em ambos os exemplos foram selecionadas 10 objetos do conjunto original para serem visualizados. Além disso, três desses objetos foram destacados nas visualizações, e suas imagens são mostradas na Figura 6.16a). Nas visualizações foram incluídos segmentos de reta conectando suas instâncias consecutivas.

Pode-se notar que a trajetória do objeto 48 apresenta a menor variação durante sua evolução, enquanto o objeto 13 apresenta maior variação. Este comportamento pode

ser validado pela análise visual das imagens desses objetos no decorrer do tempo (Figura 6.16a)). Outra observação interessante pode ser feita com relação aos formatos dos objetos 8 e 13. Durante sua evolução temporal, o objeto 8 apresenta quatro faces, todas idênticas. Já o objeto 13 também apresenta 4 faces, porém, apenas duas delas são idênticas. Na visualização das trajetórias dos objetos, pode-se identificar as imagens mapeadas dos quatro lados idênticos do objeto 8, nos instantes 0, 90, 180 e 270, destacados como estrelas. Da mesma forma, pode-se também indentificar as duas faces idênticas do objeto 13, nos instantes 90 e 270, destacados com triângulos.

6.4 Fator de escala para similaridade aplicado ao balanceamento de múltiplos descritores de imagens médicas

Nesta seção é apresentado um método que utiliza a dimensão fractal de correlação para balanceamento entre múltiplos conjuntos de características para representação de imagens em consultas por similaridade, chamado de *Fractal-scaled Product Metric* (FPM) [Bueno et al., 2009a]. O método FPM é uma modificação da técnica de balanceamento proposta juntamente com o Espaço Métrico-Temporal (Seção 6.2) para平衡ar o peso dado às componentes métrica e temporal no cálculo da similaridade.

Como discutido no Capítulo 2), as imagens são processadas por algoritmos que geram uma assinatura matemática descrevendo o conteúdo das imagens, ou seja, algoritmos extratores de características. São essas características que são comparadas no processo de recuperação por similaridade. Em muitas aplicações é necessário empregar vários extratores de características na representação das imagens para aumentar a qualidade das respostas às consultas por similaridade.

Seja uma imagem x representada por um conjunto de n descritores x_1, \dots, x_n , cada um deles gerado por um algoritmo de extração de características, e que $\delta_1, \dots, \delta_n$ sejam as funções de distância definidas nos domínios dos respectivos descritores. A função de distância de composição Δ entre duas imagens x, y é uma combinação das distâncias individuais $\delta_i(x_i, y_i)$.

$$\Delta(x, y) = \sum_{i=1}^n w_i \cdot \delta_i(x_i, y_i) \quad (6.3)$$

onde w_i é o peso dado ao respectivo descritor.

O método FPM integra múltiplos descritores de imagens seguindo a mesma idéia da Equação 6.3, satisfazendo as propriedades de uma métrica. Os pesos w_i , que podem ser considerados como fatores de escala entre as métricas utilizadas na composição, são calculados baseando-se na dimensão fractal de correlação dos espaços métricos de cada conjunto de características.

O método FPM é uma modificação do algoritmo utilizado para definir o balanceamento

entre as componentes métrica e temporal no Espaço Métrico-Temporal, apresentado na Seção 6.2. No Espaço Métrico-Temporal, estima-se o fator de escala entre as componentes métrica e temporal, ambas representadas em espaços métricos. Já no método FPM, utiliza-se o mesmo algoritmo para realizar o balanceamento entre vários espaços métricos, cada um representando um conjunto de características.

A idéia principal do método FPM é identificar a *contribuição* de cada descritor para o cálculo da similaridade global entre as imagens. O valor da dimensão intrínseca reflete a existência de correlações entre os atributos de um conjunto de dados. Assim, a dimensão fractal de correlação fornece uma estimativa do número mínimo de “características” necessários para manter as características essenciais dos dados em consultas por similaridade. Portanto, dar pesos aos múltiplos descritores em uma métrica produto de acordo com as dimensões intrínsecas dos respectivos espaços métricos, nem subestima e nem superestima a contribuição de cada descritor no cálculo da similaridade final entre as imagens.

Os resultados das métricas individuais (de cada um dos descritores) devem ser normalizadas, evitando assim que conjuntos de características com valores maiores dominem os resultados finais. Essa normalização é feita utilizando-se o maior valor de distância entre dois elementos $dmax_i$ para cada descritor. Dessa maneira, a FPM é definida como:

$$\Delta(x, y) = \sum_i^n D_{2i} \cdot \frac{\delta_i(x_i, y_i)}{dmax_i} \quad (6.4)$$

Da mesma forma como foi comentado na definição do fator de escala entre as componentes do modelo métrico-temporal na Seção 6.2.2, pode-se calcular o valor de $dmax_i$ calculando-se as distâncias entre todos os pares de elementos utilizando-se a respectiva métrica. Esta operação é muito custosa, mas técnicas muito mais baratas computacionalmente podem ser aplicadas para estimar boas aproximações. Uma boa estimativa ou até o valor exato de $dmax_i$ podem ser obtidos durante o cálculo da dimensão fractal de correlação.

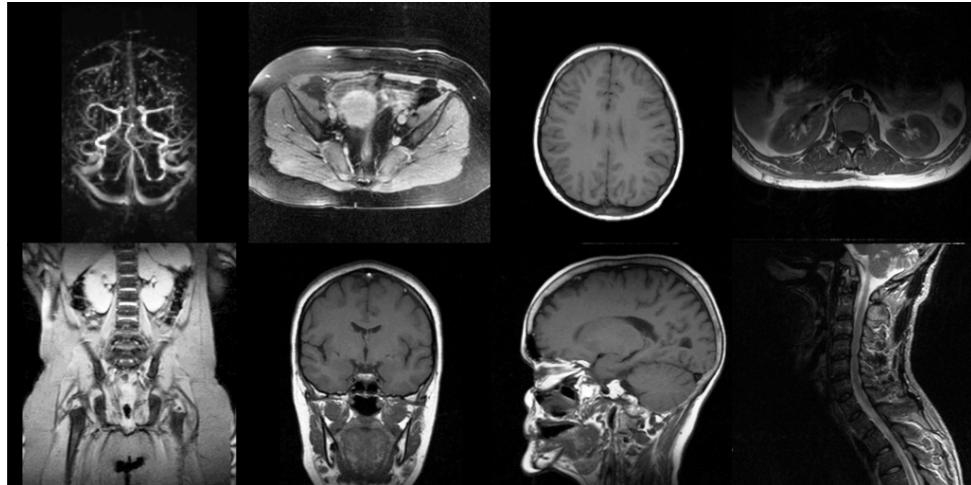


Figura 6.17: Exemplos de imagens do conjunto MRI.

6.4.1 Experimentos

Nesta seção são apresentados os resultados de experimentos realizados para avaliar o método proposto. Os experimentos foram realizados com conjuntos de imagens médicas cedidas pelo Centro de Ciência das Imagens e Física Médica (CCIFM) do Hospital das Clínicas da Faculdade de Medicina de Ribeirão Preto (HCFMRP-USP).

O primeiro conjunto de imagens, chamado de MRI, é composto por 704 imagens de exames de ressonância magnética e de angiogramas. As imagens deste conjunto são divididas em 40 classes de acordo com a região do corpo examinada, plano de visão e posição de corte. Na Figura 6.17 são apresentados exemplos dessas imagens.

As imagens, que tem tamanho de 256x256 pixels, tiveram a profundidade de cor reduzida para 8 bits, resultando em 256 níveis de cinza. Cada uma das imagens do conjunto MRI foi processada por três extratores de características, gerando três conjuntos de características distintos para o conjunto de imagens. O primeiro conjunto, *histogramas métricos*, é adimensional e contém os histogramas métricos das imagens. O segundo, chamado *Haralick*, contém características de textura, e utiliza descritores de Haralick [Haralick et al., 1973]. Os descritores de variância, entropia, energia, homogeneidade, momento de 3a ordem, variância inversa e *step* foram combinados em um vetor de características de 140 posições. Já o terceiro conjunto de características contém os 256 primeiros momentos de Zernike [Khotanzad & Hong, 1990], características que

representam as formas das imagens

O segundo conjunto de imagens, chamado de CT_Pulmão_ROIs, é uma coleção de 3257 imagens de 64 x 64 pixels e 256 níveis de cinza, contendo regiões de interesse (*Regions of Interest - ROIs*) de imagens de exames de tomografia computadorizada de pulmão. Este conjunto de imagens é organizado em 6 classes, sendo que uma dessas classes é composta por imagens normais de pulmão, e as outras 5 classes são compostas por imagens que contém diferentes padrões anormais.

As imagens do conjunto CT_Pulmão_ROIs foram processadas pelos extratores de características, gerando os conjuntos de dados com características de textura (Haralick) e forma (Zernike). Não foram utilizados os histogramas de níveis de cinza desse conjunto pois essas características não apresentaram bom desempenho com as imagens usadas neste experimento.

Os detalhes dos conjuntos de dados utilizados nos experimentos são apresentados na Tabela 6.4.

Tabela 6.4: Conjuntos de dados utilizados nos experimentos.

Nome	Tam.	Dim.	Métrica	Descrição
<i>Hist. Métricos_MRI</i>	704	—	MHD	Histogramas métricos das imagens do conjunto MRI
<i>Haralick_MRI</i>	704	140	Canberra	Descritores de Haralick das imagens do conjunto MRI
<i>Zernike_MRI</i>	704	256	L_2	Primeiros 256 momentos de Zernike das imagens do conjunto MRI
<i>Haralick_Pulmão</i>	3257	140	Canberra	Descritores de Haralick das imagens do conjunto CT_Pulmão_ROIs
<i>Zernike_Pulmão</i>	3257	256	Canberra	Primeiros 256 momentos de Zernike das imagens do conjunto CT_Pulmão_ROIs

Foram utilizados gráficos de precisão *versus* revocação para avaliar a qualidade dos resultados. Da mesma forma como discutido na Seção 6.2.3, as curvas de $P \times R$ podem ser sumarizadas por um único valor numérico. Nestes experimentos também foi utilizada a *precisão média* (área sob uma curva de $P \times R$).

Cada um dos elementos dos conjuntos de dados foi utilizado como elemento central da consulta e foram executadas consultas aos vizinhos mais próximos recuperando todos os elementos dos conjuntos, ordenados pela distância ao centro de consulta. Portanto, cada curva de $P \times R$, assim como cada valor de *precisão média*, foram obtidos pela média de 704 e 3257 consultas, respectivamente para os conjuntos MRI e CT_Pulmão_ROIs.

Para cada um dos conjuntos de características foram testadas várias métricas, e

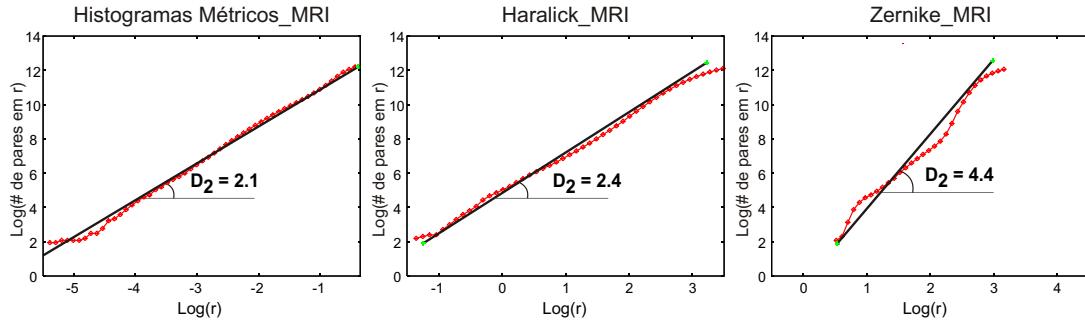


Figura 6.18: Conjunto de imagens MRI: *Distance plots* de *Histogramas Métricos_MRI*, *Haralick_MRI* e *Zernike_MRI*.

escolhidas aquelas que apresentaram os melhores resultados para cada um dos cinco conjuntos, mostrados na Tabela 6.4.

Para avaliar o método FPM, uma mesma sequência de experimentos foi realizada utilizando-se os dois conjuntos de imagens, iniciando-se com o conjunto MRI. Primeiramente calculou-se a dimensão fractal de correlação (D_2) dos conjuntos de características das imagens do conjunto MRI. Este procedimento resultou nos valores de D_2 de 2.1, 2.4 e 4.4, respectivamente para os conjuntos *Histogramas Métricos_MRI*, *Haralick_MRI* e *Zernike_MRI*. Os gráficos, gerados pela técnica *distance plot*, são mostrados na Figura 6.18.

A primeira análise a ser feita refere-se às combinações de pares de descritores. Foram calculadas curvas de $P \times R$ variando a proporção entre os descritores, e cada curva de $P \times R$ é representada por um valor de *precisão média* no gráfico. Os resultados são mostrados na Figura 6.19. Os valores da *precisão média* para cada descritor individualmente são denotadas pelas linhas horizontais. Estes valores foram de 69.2% para *Histogramas Métricos_MRI*, 68% para *Haralick_MRI* e 80.7% para *Zernike_MRI*.

Pode-se notar que combinações dos descritores apresentaram os melhores resultados na maioria dos casos, mesmo quando a proporção entre eles não é ideal. Nos gráficos também são mostrados, como linhas verticais, os valores propostos pelo método FPM para a proporção entre os descritores. Como pode ser visto, os valores propostos pelo método FPM foram muito próximos do ótimo para as três combinações de pares de descritores do conjunto MRI.

Finalmente, em um último experimento realizado com os descritores do conjunto

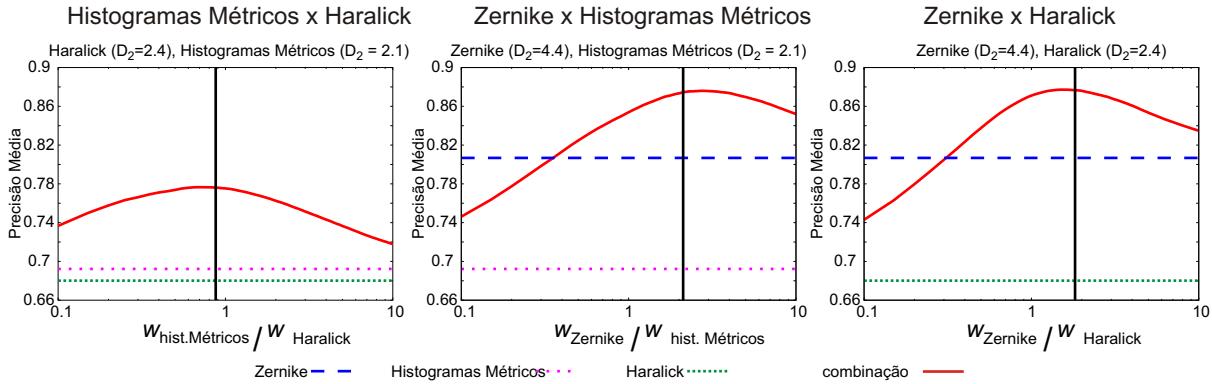


Figura 6.19: Conjunto de imagens MRI: *precisão média* dos descritores individuais e da combinação entre pares de descritores com a variação do fator de escala entre eles.

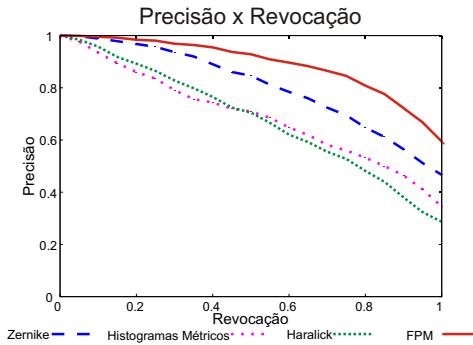


Figura 6.20: Conjunto de imagens MRI: Combinação dos descritores com o método FPM comparada com a utilização de descritores individualmente.

MRI, os valores propostos pelo método FPM foram utilizados na combinação dos três descritores, com $w_{Hist.\text{Métrico_MRI}} = 2.1$, $w_{Haralick_MRI} = 2.4$ e $w_{Zernike_MRI} = 4.4$. Essa combinação alcançou 89.2% de precisão média e superou todas as outras, seja individualmente ou utilizando pares de descritores. O aumento da precisão média foi de 29%, 31% e 11% respectivamente sobre os descritores *Histogramas Métricos_MRI*, *Haralick_MRI* e *Zernike_MRI* utilizados individualmente.

Os valores de *precisão média* dão uma visão concisa da eficácia dos resultados das consultas. Na Figura 6.20 são mostradas todas as curvas de $P \times R$ dos descritores individualmente e da combinação dos três descritores com o método FPM. Como pode ser visto, a combinação utilizando o método FPM obteve melhores resultados para todos os níveis de revocação, e em alguns casos superou a precisão dos descritores utilizados individualmente em mais de 100%.

A mesma sequência de experimentos realizados com o conjunto MRI foi realizada com o conjunto CT_Pulmão_ROIs. O método *Box Counting* foi utilizado para estimar a dimensão fractal de correlação dos conjuntos de características *Haralick_Pulmão* e *Zernike_Pulmão*, como apresentado na Figura 6.21. Os valores estimados para D_2 foram de 3.9 para o conjunto *Haralick_Pulmão* e 5.1 para o conjunto *Zernike_Pulmão*.

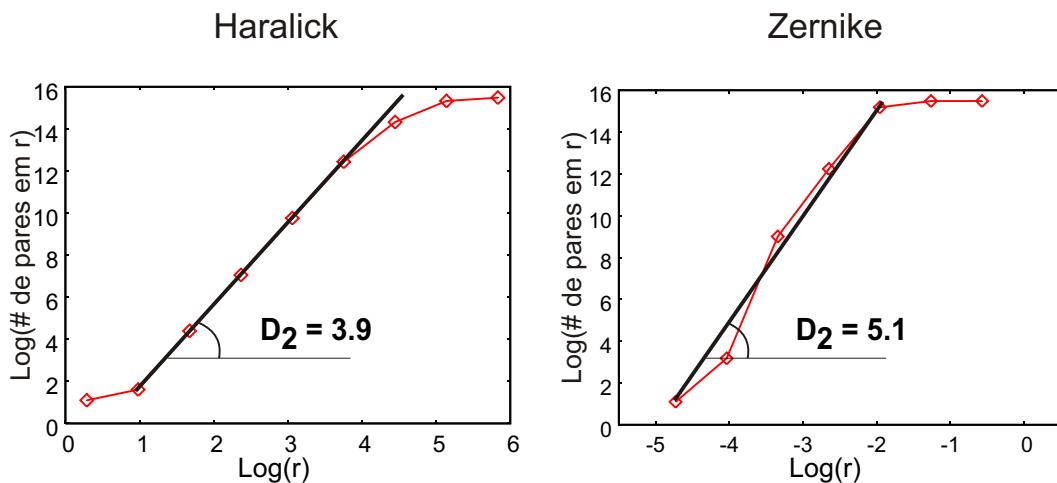


Figura 6.21: Conjunto de imagens CT_Pulmão_ROIs: *Box counting plots* dos conjuntos *Haralick Pulmão* e *Zernike Pulmão*.

Foram calculadas curvas de $P \times R$ variando-se a proporção entre os dois descritores, e os valores de *precisão média* são mostrados na Figura 6.22, assim como os valores de *precisão média* dos descritores utilizados individualmente, sendo 36.7% para o conjunto *Haralick_Pulmão* e 29.2% para o conjunto *Zernike_Pulmão*. No gráfico também encontra-se representado como uma linha vertical o fator de escala proposto pelo método FPM para a combinação dos dois descritores.

Um resultado interessante pode ser constatado a partir do gráfico da Figura 6.22. Utilizando-se os descritores individualmente, o descritor *Haralick_Pulmão* apresenta melhor precisão do que o descritor *Zernike_Pulmão*. Porém, na combinação dos dois descritores, melhores resultados de precisão são alcançados quando utiliza-se um peso maior para o descritor *Zernike_Pulmão*. Todavia, o método FPM conseguiu identificar corretamente as contribuições dos descritores no cálculo final da similaridade e novamente alcançou resultados quase ótimos, definindo os pesos como $w_{\text{Haralick_Pulmão}} = 3.9$ e $w_{\text{Zernike_Pulmão}} = 5.1$.

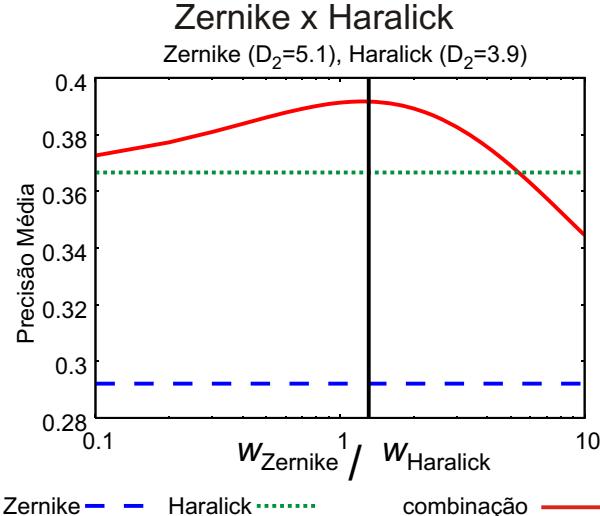


Figura 6.22: Conjunto de imagens CT_Pulmão_ROIs: *precisão média* dos descritores individuais e da combinação entre eles com a variação do fator de escala.

Por fim, na Figura 6.23 são mostradas as curvas de $P \times R$ dos descritores individualmente e da combinação entre eles com os pesos definidos pelo método FPM para o conjunto de imagens CT_Pulmão_ROIs.

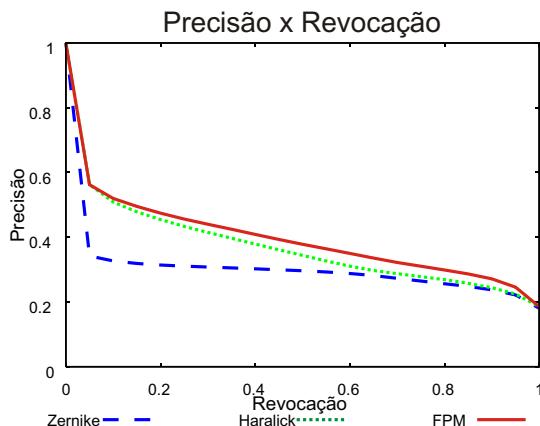


Figura 6.23: Conjunto de imagens CT_Pulmão_ROIs: Combinação dos descritores com o método FPM comparada com a utilização dos descritores individualmente.

O método FPM alcançou 39.2% de *precisão média* e superou os descritores utilizados individualmente em precisão na grande maioria dos níveis de revocação. Os ganhos chegaram a 64% sobre o descritor *Zernike Pulmão* utilizado individualmente, e a 13% sobre o descritor *Haralick Pulmão*.

A Figura 6.24 mostra a interface de uma aplicação desenvolvida para avaliação do método FPM. A figura ilustra a realização de uma consulta real aos 5 vizinhos mais

próximos sobre o conjunto de imagens MRI. São mostrados os resultados da mesma consulta utilizando todos os descritores balanceados com o método FPM e os resultados obtidos com a utilização cada descritor utilizado separadamente. As imagens assinaladas são falsos positivos.

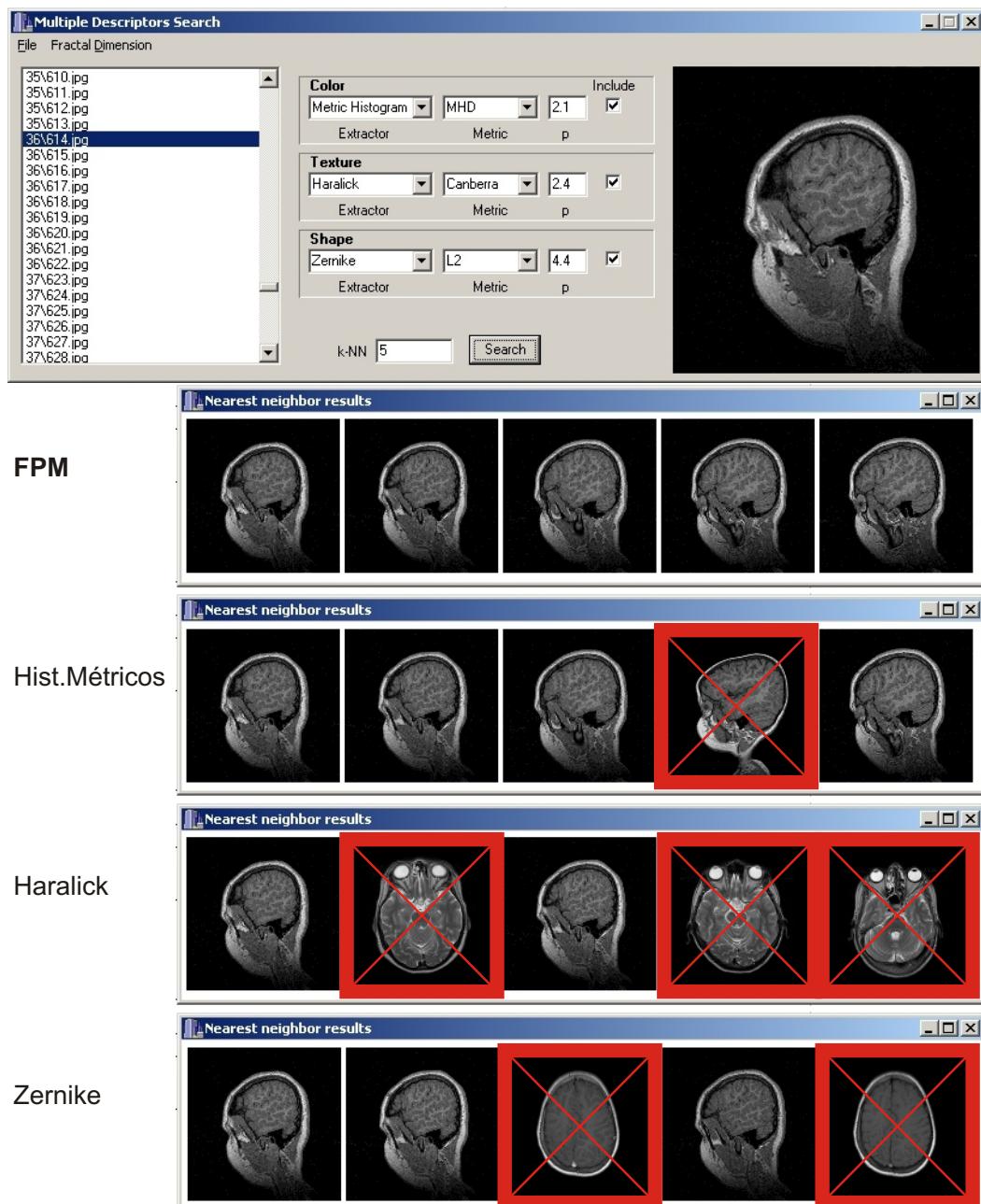


Figura 6.24: Aplicação desenvolvida para avaliação do método FPM: exemplo de uma consulta aos 5 vizinhos mais próximos utilizando o conjunto de imagens MRI.

6.5 Considerações Finais

Neste capítulo foi apresentado o Espaço Métrico-temporal, um modelo de representação de dados que permite a comparação de elementos métricos associados com informações temporais. Diferentemente dos modelos que representam o tempo como dimensões temporais, nesse espaço as informações temporais são representadas em espaços métricos, em uma componente temporal associada a componente métrica dos dados. Foi apresentado um método para identificar as contribuições relativas das componentes métrica e temporal no cálculo da similaridade. Experimentos mostraram que as informações temporais associadas aos dados métricos ajudaram a aumentar a precisão das consultas por similaridade, e que os valores estimados para o balanceamento entre as componentes foram adequados.

Em seguida foram apresentadas estratégias para analisar trajetórias de dados métricos no decorrer do tempo, através do mapeamento de espaços métrico-temporais para espaços dimensionais. A análise das trajetórias no espaço mapeado possibilita estimativas de consultas por similaridade em tempo futuro, passado e intermediário, com relação aos elementos indexados. Também foram apresentadas novas possibilidades de visualização da evolução temporal de dados métricos, com a incorporação dessas funcionalidades na ferramenta de análise visual FastMapDB.

A partir de modificações no método proposto para identificar as contribuições das componentes do espaço métrico-temporal foi apresentado um método para o balanceamento de múltiplos descritores para representação de imagens. Em experimentos realizados com imagens médicas, os fatores de balanceamento estimados pelo método proposto foram muito próximos dos valores ótimos.

Conclusão

Dados complexos, como dados multimídia, sequências genéticas, séries temporais, entre outros, necessitam ser suportados pelos Sistemas de Gerenciamento de Bancos de Dados. Porém, esses dados diferem dos dados tradicionais, como números e pequenas cadeias de caracteres, principalmente pelo modo como são recuperados. Dados complexos geralmente são recuperados por meio de consultas por similaridade, que podem ser adequadamente expressadas com a representação desses dados em espaços métricos.

Para indexar dados em espaços métricos e agilizar consultas por similaridade foram desenvolvidos os Métodos de Acesso Métrico (MAM). Porém, os MAM existentes consideram que os elementos indexados representam objetos imutáveis com o decorrer do tempo. A grande maioria sequer descreve as operações de remoção e atualização de dados. Além disso, não existem trabalhos associando tempo a dados em domínios métricos.

O objetivo deste trabalho de doutorado foi abordar a dinamicidade e a inclusão do conceito de evolução temporal de dados em espaços métricos armazenados em MAM.

Na primeira parte do trabalho foram descritos algoritmos para as operações de remoção e atualização de elementos em MAM, aumentando a dinamicidade dos MAM, que até então era limitada à inserção de novos dados. Utilizando o algoritmo de remoção proposto, foi também desenvolvido um novo método de otimização de árvores métricas.

A segunda parte do trabalho correspondeu a incorporar o suporte temporal a dados em espaços métricos. Foi desenvolvido um modelo para representação informações temporais

associadas a dados métricos, voltado principalmente para aplicações em que o tempo influi na similaridade entre os elementos. A idéia utilizada para o balanceamento das componentes deste modelo foi utilizada para integrar múltiplos descritores de imagens representados em espaços métricos, também apresentando resultados promissores. Por fim, também foram discutidas estratégias para análise de trajetórias de dados métricos no decorrer do tempo, através da imersão de espaços métrico-temporais em espaços dimensionais.

7.1 Principais contribuições

As principais contribuições deste trabalho de doutorado estão descritas a seguir:

Algoritmos de remoção efetiva e atualização de dados em MAM Os MAM existentes, mesmo aqueles considerados dinâmicos, consideram que cada elemento de dado representa um objeto imutável no tempo. Foram desenvolvidos algoritmos para atualização e de remoção efetiva de dados em MAM [Bueno et al., 2008b, Bueno et al., 2008a]. Os agoritmos foram incorporados ao MAM *Slim-tree*, que passou a permitir a atualização ou remoção efetiva de qualquer elemento indexado.

Redução de sobreposição e otimização em MAM Foi desenvolvida uma nova técnica de otimização de MAM dinâmicos [Bueno et al., 2008a], baseada no algoritmo de remoção proposto. O método *Push-Pull* reduz a sobreposição dos nós promovendo a remoção de elementos que estejam na periferia dos nós, para posteriormente reinserí-los na árvore.

Espaço Métrico-Temporal Foi proposto um modelo de representação de dados Métrico-temporal [Bueno et al., 2009b] que permite a comparação dos elementos métricos associados ao tempo. Foram desenvolvidos métodos para definir a contribuição relativa das componentes métrica e temporal de um espaço métrico-temporal para o cálculo final da similaridade.

Trajetória de dados métrico-Temporais Foi proposto um método que permite a

análise das trajetórias dos elementos métrico-temporais através da imersão do espaço métrico-temporal em um espaço dimensional. Com isso, pode-se realizar consultas aproximadas sobre este novo espaço mapeado, estimando os resultados em tempo futuro, passado e intermediário. Também foram adicionadas novas possibilidades de visualização na ferramenta FastMapDB para permitir a análise visual da evolução temporal dos dados métricos.

Balanceamento de múltiplos descritores Foi desenvolvido um método não supervisionado para o balanceamento entre múltiplos conjuntos de características para representação de imagens em consultas por similaridade [Bueno et al., 2009a]. O método proposto utiliza a dimensão fractal de correlação e é uma generalização da técnica de balanceamento proposta para o Espaço Métrico-Temporal.

7.2 Trabalhos Futuros

Várias possibilidades de trabalhos futuros podem ser exploradas a partir deste trabalho.

Os resultados discutidos neste trabalho demonstraram a possibilidade da utilização da trajetória de elementos métrico-temporais para realização de consultas por similaridade aproximadas em diferentes posições temporais, através do mapeamento para espaços multidimensionais. Em um trabalho futuro pode-se estudar outras opções de mapeamento, visando melhorar a qualidade do espaço mapeado e, consequentemente, aumentar a precisão das respostas.

Neste trabalho foram utilizados apenas duas instâncias de elementos indexados para estimativa das trajetórias. Em uma possível extensão, pode-se utilizar todas as instâncias disponíveis para essa estimativa. Além disso, pode-se utilizar conhecimentos de especialistas do domínio, ou então conhecimento extraído dos dados, para identificar padrões de evolução dos mesmos e melhorar as estimativas das trajetórias.

Outra possibilidade é estender o método proposto para balanceamento de múltiplos descritores, estudando outras métricas-produto que possibilitem a definição de funções mais complexas. Em outro trabalho, pretende-se manipular múltiplos conjuntos de

características de dados complexos para satisfazer as expectativas do usuário, incluindo variedade nos resultados de consultas por similaridade.

Referências Bibliográficas

- [Achtert et al., 2009] Achtert, E., Kriegel, H.-P., Kröger, P., Renz, M., e Züfle, A. (2009). Reverse k-nearest neighbor search in dynamic and general metric databases. In *EDBT '09: Proceedings of the 12th International Conference on Extending Database Technology*, pp. 886–897, New York, NY, USA. ACM.
- [Adjerooh et al., 1999] Adjerooh, D. A., Lee, M. C., e King, I. (1999). A distance measure for video sequences. *Computer Vision and Image Understanding*, 75(1/2):25–45.
- [Agarwal et al., 2000] Agarwal, P. K., Arge, L., e Erickson, J. (2000). Indexing moving points (extended abstract). In *PODS '00: Proceedings of the nineteenth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pp. 175–186. New York, NY, USA.
- [Agarwal & Procopiuc, 2002] Agarwal, P. K. e Procopiuc, C. M. (2002). Advances in indexing for mobile objects. *IEEE Data Eng. Bull.*, 25(2):25–34.
- [Aggarwal & Agrawal, 2003] Aggarwal, C. C. e Agrawal, D. (2003). On nearest neighbor indexing of nonlinear trajectories. In *PODS '03: Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pp. 252–259. New York, NY, USA.
- [Aksoy & Haralick, 2001] Aksoy, S. e Haralick, R. M. (2001). Feature normalization and likelihood-based similarity measures for image retrieval. *Pattern Recognition Letters*, 22(5):563–582.
- [Amato et al., 1997] Amato, G., Mainetto, G., e Savino, P. (1997). A query language for similarity-based retrieval of multimedia data. In *Advances in Databases and Information Systems (ADBIS)*, v. 1, pp. 196–203, St.-Petersburg, Russia. Nevsky Dialect.
- [Andaló et al., 2010] Andaló, F. A., Miranda, P. A. V., Torres, R. d. S., e Falcão, A. X. (2010). Shape feature extraction and description based on tensor scale. *Pattern Recogn.*, 43(1):26–36.
- [Arantes et al., 2003] Arantes, A. S., Vieira, M. R., Traina, A. J. M., e Traina Jr., C. (2003). The fractal dimension making similarity queries more efficient. In *Second Workshop on Fractals and Self-similarity in Data Mining: Issues and Approaches (in conjunction with 9th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining)*, pp. 12–17, Washington, DC. ACM Press.
- [Ashwin et al., 2002] Ashwin, T., Gupta, R., e Ghosal, S. (2002). Adaptable similarity search using non-relevant information. In *International Conference on Very Large Databases (VLDB)*, pp. 47–58, Hong Kong, China. Morgan Kaufmann.

- [Atnafu et al., 2004] Atnafu, S., Chbeir, R., Coquil, D., e Brunie, L. (2004). Integrating similarity-based queries in image dbmss. In Haddad, H., Omicini, A., Wainwright, R. L., e Liebrock, L. M., editors, *ACM symposium on Applied computing*, pp. 735 – 739, Nicosia, Cyprus. ACM Press.
- [Attikos & Doumpos, 2009] Attikos, C. e Doumpos, M. (2009). Faster estimation of the correlation fractal dimension using box-counting. *CoRR*, abs/0905.4138.
- [Baeza-Yates et al., 1994] Baeza-Yates, R. A., Cunto, W., Manber, U., e Wu, S. (1994). Proximity matching using fixed-queries trees. In *Combinatorial Pattern Matching (CPM)*, v. 807 of *Lecture Notes in Computer Science*, pp. 198–212, Asilomar, CA. Springer Verlag.
- [Baeza-Yates & Ribeiro-Neto, 1999] Baeza-Yates, R. A. e Ribeiro-Neto, B. A. (1999). *Modern Information Retrieval*. Addison-Wesley, Wokingham, UK.
- [Barioni et al., 2002] Barioni, M. C. N., Botelho, E., Faloutsos, C., Razente, H. L., Traina, A. J. M., e Traina Jr., C. (2002). Data visualization in rdbms. In Kiyoki, Y., Yoshikawa, M., e Tanaka, K., editors, *IASTED Intl. Conference Information Systems and Databases (ISDB 2002)*, pp. 264–269, Tokyo, Japan. Acta Press.
- [Beckmann et al., 1990] Beckmann, N., Kriegel, H.-P., Schneider, R., e Seeger, B. (1990). The r*-tree: An efficient and robust access method for points and rectangles. In *ACM SIGMOD International Conference on Management of Data*, pp. 322–331.
- [Belussi & Faloutsos, 1995] Belussi, A. e Faloutsos, C. (1995). Estimating the selectivity of spatial queries using the correlation fractal dimension. In Dayal, U., Gray, P. M. D., e Nishio, S., editors, *International Conference on Very Large Databases (VLDB)*, pp. 299–310, Zurich, Switzerland. Morgan Kaufmann.
- [Berchtold et al., 1998] Berchtold, S., Ertl, B., Keim, D. A., Kriegel, H.-P., e Seidl, T. (1998). Fast nearest neighbor search in high-dimensional space. In *IEEE International Conference on Data Engineering (ICDE)*, pp. 209–218, Orlando, FL.
- [Bertino et al., 1996] Bertino, E., Ferrari, E., e Guerrini, G. (1996). A formal temporal object-oriented data model. In *EDBT '96: Proceedings of the 5th International Conference on Extending Database Technology*, pp. 342–356, London, UK. Springer-Verlag.
- [Böhm et al., 2001] Böhm, C., Berchtold, S., e Keim, D. A. (2001). Searching in high-dimensional spaces - index structures for improving the performance of multimedia databases. *ACM Computing Surveys*, 33(3):322 – 373.
- [Bozkaya & Özsoyoglu, 1997] Bozkaya, T. e Özsoyoglu, Z. M. (1997). Distance-based indexing for high-dimensional metric spaces. In *ACM SIGMOD International Conference on Management of Data*, pp. 357–368, Tucson, AZ. ACM Press.
- [Bozkaya & Özsoyoglu, 1999] Bozkaya, T. e Özsoyoglu, Z. M. (1999). Indexing large metric spaces for similarity search queries. *ACM Transactions on Database Systems (TODS)*, 24(3):361–404.

-
- [Braunmüller et al., 2000] Braunmüller, B., Ester, M., Kriegel, H.-P., e Sander, J. (2000). Efficiently supporting multiple similarity queries for mining in metric databases. In *IEEE International Conference on Data Engineering (ICDE)*, pp. 256–267, San Diego, CA. IEEE Computer Society.
- [Brin, 1995] Brin, S. (1995). Near neighbor search in large metric spaces. In Dayal, U., Gray, P. M. D., e Nishio, S., editors, *International Conference on Very Large Databases (VLDB)*, pp. 574–584, Zurich, Switzerland. Morgan Kaufmann.
- [Bueno et al., 2009a] Bueno, R., Kaster, D. d. S., Paterlini, A. A., Traina, A. J. M., e Traina, Caetano, J. (2009a). Unsupervised scaling of multi-descriptor similarity functions for medical image datasets. In *22th IEEE Intl. Symposium on Computer-Based Medical Systems (CBMS 2009)*, pp. 1–8, Albuquerque, NM, EUA. IEEE Computer Society.
- [Bueno et al., 2008a] Bueno, R., Kaster, D. d. S., Traina, A. J. M., e Traina, Caetano, J. (2008a). A new approach for optimization of dynamic metric access methods using an algorithm of effective deletion. In *20th International Conference on Scientific and Statistical Database Management (SSDBM 2008)*, v. 5069/2008, pp. 366–383, Hong Kong S.A.R., China. Springer Berlin / Heidelberg.
- [Bueno et al., 2009b] Bueno, R., Kaster, D. d. S., Traina, A. J. M., e Traina, Caetano, J. (2009b). Time-aware similarity search: a metric-temporal representation for complex data. In Mamoulis, N., Seidl, T., Pedersen, T. B., Torp, K., e Assent, I., editors, *11th International Symposium on Advances in Spatial and Temporal Databases (SSTD 2009)*, pp. 302–319, Aalborg, Denmark. Springer.
- [Bueno et al., 2008b] Bueno, R., Traina, A. J. M., e Traina, Caetano, J. (2008b). An algorithm for effective deletion and a new optimization technique for metric access methods. In *23rd Annual ACM Symposium on Applied Computing (SAC2008)*, pp. 1034–1035, Fortaleza, Ceará - Brazil. ACM Press.
- [Bueno et al., 2005a] Bueno, R., Traina, Caetano, J., e Traina, A. J. M. (2005a). Algoritmos genéticos para consultas por similaridade aproximadas. In Heuser, C. A. e de Amo, S. A., editors, *Simpósio Brasileiro de Banco de Dados*, v. 1, pp. 190–204, Uberlândia, MG. SBC.
- [Bueno et al., 2005b] Bueno, R., Traina Jr., C., e Traina, A. J. M. (2005b). Accelerating approximate similarity queries using genetic algorithms. In *ACM Intl. Conference on Applied Computing (SAC)*, v. 1, pp. 621–626, Santa Fe, New Mexico. ACM Press.
- [Bugatti et al., 2008] Bugatti, P. H., Traina, A. J. M., e Traina Jr., C. (2008). Assessing the best integration between distance-function and image-feature to answer similarity queries. In *SAC*, pp. 1225–1230, Fortaleza, CE, Brazil. ACM.
- [Burkhard & Keller, 1973] Burkhard, W. A. e Keller, R. M. (1973). Some approaches to best-match file searching. *Communications of the ACM (CACM)*, 16(4):230–236.
- [Bustos et al., 2004] Bustos, B., Keim, D., Saupe, D., Schreck, T., e Vrancic, D. (2004). Automatic selection and combination of descriptors for effective 3d similarity search. In *Multimedia Software Engineering*, pp. 514–521, Miami, FL, USA. IEEE.

- [Bustos & Navarro, 2009] Bustos, B. e Navarro, G. (2009). Improving the space cost of -nn search in metric spaces by using distance estimators. *Multimedia Tools Appl.*, 41(2):215–233.
- [Caicedo et al., 2007] Caicedo, J. C., González, F. A., Triana, E., e Romero, E. (2007). Design of a medical image database with content-based retrieval capabilities. In *Advances in Image and Video Technology*, pp. 919–931, Santiago, Chile. Springer.
- [Cantone et al., 2005] Cantone, D., Ferro, A., Pulvirenti, A., Recupero, D. R., e Shasha, D. (2005). Antipole tree indexing to support range search and k-nearest neighbor search in metric spaces. *IEEE Transactions on Knowledge and Data Engineering*, 17(4):535–550.
- [Chen & Zaniolo, 1999] Chen, C. X. e Zaniolo, C. (1999). Universal temporal extensions for database languages. In *IEEE International Conference on Data Engineering (ICDE)*, pp. 428–437, Sydney, Australia. IEEE Computer Society.
- [Chen et al., 2007] Chen, Y.-S., Hung, Y.-P., Yen, T.-F., e Fuh, C.-S. (2007). Fast and versatile algorithm for nearest neighbor search based on a lower bound tree. *Pattern Recogn.*, 40(2):360–375.
- [Chon et al., 2003] Chon, H. D., Agrawal, D., e Abbadi, A. E. (2003). Range and knn query processing for moving objects in grid model. *Mob. Netw. Appl.*, 8(4):401–412.
- [Chávez & Navarro, 2001] Chávez, E. e Navarro, G. (2001). Towards measuring the searching complexity of metric spaces. In *Proc. Mexican Computing Meeting*, v. II, pp. 969–978, Aguascalientes, México. Sociedad Mexicana de Ciencias de la Computación.
- [Chávez et al., 2001] Chávez, E., Navarro, G., Baeza-Yates, R. A., e Marroquín, J. L. (2001). Searching in metric spaces. *ACM Computing Surveys*, 33(3):273–321.
- [Ciaccia et al., 1997] Ciaccia, P., Patella, M., e Zezula, P. (1997). M-tree: An efficient access method for similarity search in metric spaces. In Jarke, M., editor, *International Conference on Very Large Databases (VLDB)*, pp. 426–435, Athens, Greece. Morgan Kaufmann.
- [Clifford & Croker, 1987] Clifford, J. e Croker, A. (1987). The historical relational data model (hrdm) and algebra based on lifespans. In *ICDE*, pp. 528–537.
- [Corral et al., 2008] Corral, A., Torres, M., Vassilakopoulos, M., e Manolopoulos, Y. (2008). Predictive join processing between regions and moving objects. In *ADBIS*, pp. 46–61, Pori, Finland. Springer.
- [Datta et al., 2008] Datta, R., Joshi, D., Li, J., e Wang, J. Z. (2008). Image retrieval: Ideas, influences, and trends of the new age. *ACM Comput. Surv.*, 40(2):1–60.
- [de Oliveira et al., 1995] de Oliveira, J. P. M., Edelweiss, N., Arruda, E., Laender, A. H. F., e Cavalcanti, J. M. B. (1995). Implementation of an object-oriented temporal model. In *DEXA Workshop*, pp. 35–44.
- [Dohnal et al., 2003] Dohnal, V., Gennaro, C., Savino, P., e Zezula, P. (2003). D-index: Distance searching index for metric data sets. *Multimedia Tools and Applications Journal (MTAJ)*, 21(1):9–33.

-
- [Downie & Nelson, 2000] Downie, S. e Nelson, M. (2000). Evaluation of a simple and effective music information retrieval method. In Belkin, N. J., Ingwersen, P., e Leong, M.-K., editors, *23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 73 – 80, Athens, Greece. ACM Press.
- [Edelweiss, 1998] Edelweiss, N. (1998). Bancos de dados temporais: Teoria e prática. In *XVII Jornada de Atualização em Informática: Anais do XVIII Congresso Nacional da Sociedade Brasileira de Computação “Rumo à Sociedade do Conhecimento”*, pp. 225–282. Sociedade Brasileira de Computação.
- [Edelweiss et al., 1993] Edelweiss, N., de Oliveira, J. P. M., e Pernici, B. (1993). An object-oriented temporal model. In *CAiSE '93: Proceedings of Advanced Information Systems Engineering*, pp. 397–415, London, UK. Springer-Verlag.
- [Elmasri et al., 1993] Elmasri, R., Kouramajian, V., e Fernando, S. (1993). Temporal database modeling: an object-oriented approach. In *CIKM '93: Proceedings of the second international conference on Information and knowledge management*, pp. 574–585, New York, NY, USA. ACM Press.
- [Elmasri & Navathe, 2006] Elmasri, R. e Navathe, S. B. (2006). *Fundamentals of Database Systems (5th Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.
- [Erwig & Schneider, 2002] Erwig, M. e Schneider, M. (2002). Spatio-temporal predicates. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 14(4):881–901.
- [Faloutsos, 1996] Faloutsos, C. (1996). *Searching Multimedia Databases by Content*. Kluwer Academic Publishers, Boston, MA.
- [Faloutsos, 1997] Faloutsos, C. (1997). Indexing of multimedia data. In *Multimedia Databases in Perspective*, pp. 219–245. Springer Verlag.
- [Faloutsos & Kamel, 1994] Faloutsos, C. e Kamel, I. (1994). Beyond uniformity and independence: Analysis of r-trees using the concept of fractal dimension. In *ACM Symposium on Principles of Database Systems (PODS)*, pp. 4–13, Minneapolis, MN. ACM Press.
- [Faloutsos & Lin, 1995] Faloutsos, C. e Lin, K.-I. D. (1995). Fastmap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. In Carey, M. J. e Schneider, D. A., editors, *ACM SIGMOD International Conference on Management of Data*, pp. 163–174, San Jose, CA. ACM Press.
- [Faria et al., 1998] Faria, G., Medeiros, C. B., e Nascimento, M. A. (1998). An extensible framework for spatio-temporal database applications. In *SSDBM '98: Proceedings of the 10th International Conference on Scientific and Statistical Database Management*, pp. 202–205, Washington, DC, USA. IEEE Computer Society.
- [Fauvet et al., 1997] Fauvet, M., Canavaggio, J., e Scholl, P. (1997). Modeling histories in object dbms. In *DEXA*, pp. 112–121, Toulouse, France.
- [Fedel et al., 2006] Fedel, G. d. S., Razente, H. L., Traina, A. J. M., e Traina, Caetano, J. (2006). Fastmapdb: Uma ferramenta para visualização em sgbdrs com uma

- implementação interativa do algoritmo para detecção de agrupamentos k-medoid. In *3^a Sessão de Demos em Banco de Dados, junto com o 21º Simpósio Brasileiro de Bases de Dados (SBBD '06)*, v. 1, pp. 31–36, Florianópolis, SC. SBC.
- [Ferreira et al., 2008] Ferreira, C. D., da Silva Torres, R., Gonçalves, M. A., e Fan, W. (2008). Image retrieval with relevance feedback based on genetic programming. In *SBBD*, pp. 120–134.
- [Ferreira et al., 2005] Ferreira, M. R. P., Bueno, R., e Traina Jr., C. (2005). Dbgen - gerador de dados sintéticos com distribuição fractal. In Brayner, n. e Dorneles, C. F., editors, *2^a Sessão de Demos em Banco de Dados - junto ao 20º Simpósio Brasileiro de banco de Dados*, pp. 25–30, Uberlândia, MG.
- [Forlizzi et al., 2000] Forlizzi, L., Güting, R. H., Nardelli, E., e Schneider, M. (2000). A data model and data structures for moving objects databases. In *SIGMOD '00: Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pp. 319–330. New York, NY, USA.
- [Fu et al., 2000] Fu, A. W.-c., Chan, P. M.-s., Cheung, Y.-L., e Moon, Y. S. (2000). Dynamic vp-tree indexing for n-nearest neighbor search given pair-wise distances. *The International Journal on Very Large Databases*, 9(2):154–173.
- [Gadia & Yeung, 1988] Gadia, S. K. e Yeung, C.-S. (1988). A generalized model for a relational temporal database. In *SIGMOD '88: Proceedings of the 1988 ACM SIGMOD international conference on Management of data*, pp. 251–259, New York, NY, USA. ACM Press.
- [Gaede & Günther, 1998] Gaede, V. e Günther, O. (1998). Multidimensional access methods. *ACM Computing Surveys*, 30(2):170–231.
- [Gennaro et al., 2001] Gennaro, C., Savino, P., e Zezula, P. (2001). Similarity search in metric databases through hashing. In *3rd International Workshop on Multimedia Information Retrieval*, pp. 1–5, Ottawa, Canada.
- [Geusebroek et al., 2005] Geusebroek, J. M., Burghouts, G. J., e Smeulders, A. W. M. (2005). The Amsterdam library of object images. *Int. J. Comput. Vis.*, 61(1):103–112.
- [Güting, 1994] Güting, R. H. (1994). An introduction to spatial database systems. *VLDB Journal*, 3(4):357–399.
- [Güting et al., 2000] Güting, R. H., Böhlen, M. H., Erwig, M., Jensen, C. S., Lorentzos, N. A., Schneider, M., e Vazirgiannis, M. (2000). A foundation for representing and querying moving objects. *ACM Trans. Database Syst.*, 25(1):1–42.
- [Guliato et al., 2008] Guliato, D., de Carvalho, J. D., Rangayyan, R. M., e Santiago, S. A. (2008). Feature extraction from a signature based on the turning angle function for the classification of breast tumors. *J. Digital Imaging*, 21(2):129–144.
- [Guttman, 1984] Guttman, A. (1984). R-tree : A dynamic index structure for spatial searching. In *ACM SIGMOD International Conference on Management of Data*, pp. 47–57, Boston, MA. ACM Press.

-
- [Haralick et al., 1973] Haralick, R. M., Shanmugam, K., e Dinstein, I. (1973). Textural features for image classification. *IEEE Transactions on Systems, Man, and Cybernetics*, 3:610–621.
- [Heesch & Rüger, 2002] Heesch, D. e Rüger, S. (2002). Combining features for content-based sketch retrieval – a comparative evaluation of retrieval performance. In *In Proc. 24th BCS-IRSG European Colloquium on IR Research*, pp. 41–52. Springer.
- [Hjaltason & Samet, 1999] Hjaltason, G. R. e Samet, H. (1999). Distance browsing in spatial databases. *ACM Transactions on Database Systems (TODS)*, 24(2):265 – 318.
- [Hjaltason & Samet, 2003] Hjaltason, G. R. e Samet, H. (2003). Index-driven similarity search in metric spaces. *ACM Transactions on Database Systems (TODS)*, 21(4):517 – 580.
- [Howarth & Rüger, 2004] Howarth, P. e Rüger, S. M. (2004). Evaluation of texture features for content-based image retrieval. In *CIVR*, pp. 326–334.
- [Hu & Lee, 2006] Hu, H. e Lee, D. L. (2006). Range nearest-neighbor query. *IEEE Transactions on Knowledge and Data Engineering*, 18(1):79–91.
- [Huang & Dai, 2003] Huang, P.-W. e Dai, S. K. (2003). Image retrieval by texture similarity. *Pattern Recognition Letters*, 36(3):665–679.
- [Huang et al., 2009] Huang, Y.-K., Liao, S.-J., e Lee, C. (2009). Evaluating continuous k-nearest neighbor query on moving objects with uncertainty. *Inf. Syst.*, 34(4-5):415–437.
- [Inam & Matin, 2003] Inam, O. e Matin, A. (2003). A survey of indexing techniques for moving object trajectories. Technical report, University of Waterloo, Canada.
- [Ishikawa et al., 2000] Ishikawa, M., Chen, H., Furuse, K., Yu, J. X., e Ohbo, N. (2000). Mb+tree: A dynamically updatable metric index for similarity searches. In Lu, H. e Zhou, A., editors, *Web-Age Information Management*, v. 1846 of *Lecture Notes in Computer Science*, pp. 356–373, Shanghai, China. Springer Verlag.
- [Iwerks et al., 2003] Iwerks, G. S., Samet, H., e Smith, K. (2003). Continuous k-nearest neighbor queries for continuously moving points with updates. In *VLDB*, pp. 512–523.
- [Jensen et al., 1997] Jensen, C. S., Dyrson, C. E., Böhlen, M. H., Clifford, J., Elmasri, R., Gadia, S. K., Grandi, F., Hayes, P. J., Jajodia, S., Käfer, W., Kline, N., Lorentzos, N. A., Mitsopoulos, Y. G., Montanari, A., Nonen, D. A., Peressi, E., Pernici, B., Roddick, J. F., Sarda, N. L., Scalas, M. R., Segev, A., Snodgrass, R. T., Soo, M. D., Tansel, A. U., Tiberio, P., e Wiederhold, G. (1997). The consensus glossary of temporal database concepts - february 1998 version. In *Temporal Databases, Dagstuhl*, pp. 367–405.
- [Jensen et al., 2004] Jensen, C. S., Lin, D., e Ooi, B. C. (2004). Query and update efficient b+-tree based indexing of moving objects. In *VLDB*, pp. 768–779.
- [Jensen et al., 2007] Jensen, C. S., Lin, D., e Ooi, B. C. (2007). Continuous clustering of moving objects. *IEEE Trans. Knowl. Data Eng.*, 19(9):1161–1174.

- [Jensen & Saltenis, 2002a] Jensen, C. S. e Saltenis, S. (2002a). Indexing of moving objects for location-based services. In *ICDE '02: Proceedings of the 18th International Conference on Data Engineering*, pp. 463–472, Washington, DC, USA. IEEE Computer Society.
- [Jensen & Saltenis, 2002b] Jensen, C. S. e Saltenis, S. (2002b). Towards increasingly update efficient moving-object indexing. *IEEE Data Eng. Bull.*, 25(2):35–40.
- [Jeong et al., 2007] Jeong, S., Kim, S.-W., e Choi, B.-U. (2007). Dimensionality reduction in high-dimensional space for multimedia information retrieval. In *DEXA*, pp. 404–413.
- [Jiang et al., 2000] Jiang, L., Salzberg, B., Lomet, D. B., e Barrena, M. (2000). The b-tree: A branched and temporal access method. In El Abbadi, A., Brodie, M. L., Chakravarthy, S., Dayal, U., Kamel, N., Schlageter, G., e Whang, K.-Y., editors, *International Conference on Very Large Databases (VLDB)*, pp. 451–460, Cairo - Egypt. Morgan Kaufmann.
- [Johnson & Shasha, 1993] Johnson, T. e Shasha, D. (1993). The performance of current b-tree algorithms. *ACM Transactions on Database Systems (TODS)*, 18(1):51–101.
- [Khotanzad & Hong, 1990] Khotanzad, A. e Hong, Y. H. (1990). Invariant image recognition by zernike moments. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 12(5):489–497.
- [Kollios et al., 1999] Kollios, G., Gunopulos, D., e Tsotras, V. J. (1999). On indexing mobile objects. In *PODS '99: Proceedings of the eighteenth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pp. 261–272. New York, NY, USA.
- [Kollios et al., 2005] Kollios, G., Papadopoulos, D., Gunopulos, D., e Tsotras, V. J. (2005). Indexing mobile objects using dual transformations. *The International Journal on Very Large Databases*, 14(2):238 – 256.
- [Korn et al., 2001] Korn, F., Pagel, B.-U., e Faloutsos, C. (2001). On the 'dimensionality curse' and the 'self-similarity blessing'. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 13(1):96–111.
- [Korn et al., 1996] Korn, F., Sidiropoulos, N., Faloutsos, C., Siegel, E. L., e Protopapas, Z. (1996). Fast nearest neighbor search in medical image databases. In Vijayaraman, T. M., Buchmann, A. P., Mohan, C., e Sarda, N. L., editors, *International Conference on Very Large Databases (VLDB)*, pp. 215–226, Bombay, India. Morgan Kaufmann.
- [Koudas et al., 2004] Koudas, N., Ooi, B. C., Tan, K.-L., e 0003, R. Z. (2004). Approximate nn queries on streams with guaranteed error/performance bounds. In *VLDB*, pp. 804–815.
- [Kriegel et al., 2002] Kriegel, H.-P., Pfeifle, M., Pötke, M., e Seidl, T. (2002). A cost model for interval intersection queries on ri-trees. In Kennedy, J. e Lamb, J., editors, *14th International Conference on Scientific and Statistical Database Management*, pp. 131–141, Edinburgh, Scotland, UK. IEEE Computer Society.

-
- [Kriegel et al., 2004] Kriegel, H.-P., Pfeifle, M., Pötke, M., e Seidl, T. (2004). A cost model for spatial intersection queries on ri-trees. In Lee, Y., Li, J., Whang, K.-Y., e Lee, D., editors, *9th International Conference on Database Systems for Advanced Applications: DASFAA 2004*, v. 2973 / 2004 of *Lecture Notes in Computer Science*, pp. 331–338, Jeju Island, Korea. Springer Verlag.
- [Kriegel et al., 2000] Kriegel, H.-P., Pötke, M., e Seidl, T. (2000). Managing intervals efficiently in object-relational databases. In Abbadi, A. E., Brodie, M. L., Chakravarthy, S., Dayal, U., Kamel, N., Schlageter, G., e Whang, K.-Y., editors, *International Conference on Very Large Databases (VLDB)*, pp. 407–418, Cairo, Egypt. Morgan Kaufmann.
- [Ku et al., 2006] Ku, W.-S., Zimmermann, R., Wang, H., e Nguyen, T. (2006). Annatto: Adaptive nearest neighbor queries in travel time networks. In *IEEE International Conference on Mobile Data Management (MDM'06)*, pp. 50–55. IEEE Computer Society.
- [Kwon et al., 2002] Kwon, D., Lee, S., e Lee, S. (2002). Indexing the current positions of moving objects using the lazy update r-tree. In *MDM '02: Proceedings of the Third International Conference on Mobile Data Management*, pp. 113–120. Washington, DC, USA.
- [Lane et al., 2000] Lane, M. A., Edwards, J. L., e Nielsen, E. S. (2000). Biodiversity informatics: The challenge of rapid development, large databases, and complex data. In El Abbadi, A., Brodie, M. L., Chakravarthy, S., Dayal, U., Kamel, N., Schlageter, G., e Whang, K.-Y., editors, *International Conference on Very Large Databases (VLDB)*, pp. 188–199, Cairo, Egypt. Morgan Kaufmann.
- [Lee et al., 2008] Lee, K. C. K., Zheng, B., e Lee, W.-C. (2008). Ranked reverse nearest neighbor search. *IEEE Trans. on Knowl. and Data Eng.*, 20(7):894–910.
- [Lee et al., 2003] Lee, M.-L., Hsu, W., Jensen, C. S., Cui, B., e Teo, K. L. (2003). Supporting frequent updates in r-trees: A bottom-up approach. In *VLDB*, pp. 608–619.
- [Levenshtein, 1966] Levenshtein, V. (1966). Binary codes capable of correcting deletions, insertions, and reversals. *Cybernetics and Control Theory*, 10(8):707–710.
- [Li et al., 2008] Li, D., Peng, Y.-h., e Yin, J.-l. (2008). Quadtree and hash table based index structure for indexing the past, present and future positions of moving objects. In *CSA '08: Proceedings of the International Symposium on Computer Science and its Applications*, pp. 17–21, Washington, DC, USA. IEEE Computer Society.
- [Lim et al., 2006] Lim, S.-H., Ku, K.-I., Kim, K., e Kim, Y.-S. (2006). A node split algorithm reducing overlapped index spaces in m-tree index. In *ICDEW '06: Proceedings of the 22nd International Conference on Data Engineering Workshops (ICDEW'06)*, pp. 15–23, Washington, DC, USA. IEEE Computer Society.
- [Lin et al., 2005] Lin, D., Jensen, C. S., Ooi, B. C., e altenis, S. (2005). Efficient indexing of the historical, present, and future positions of moving objects. In *MDM '05: Proceedings of the 6th international conference on Mobile data management*, pp. 59–66, New York, NY, USA. ACM Press.

- [Liu & Chen, 2002] Liu, C.-C. e Chen, A. L. P. (2002). 3d-list: A data structure for efficient video query processing. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 14(1):106–122.
- [Manica et al., 2009a] Manica, E., CERVI, C. R., DORNELES, C. F., e GALANTE, R. (2009a). Emap - uma interface de consultas temporais em sgbds relacionais. In *Sessão de Demos - XXIV Simpósio Brasileiro de Banco de Dados*, pp. 1–6, Fortaleza, CE.
- [Manica et al., 2009b] Manica, E., CERVI, C. R., DORNELES, C. F., e GALANTE, R. (2009b). Ferramenta para suporte a consultas temporais em sgbds convencionais. In *Escola Regional de Banco de Dados 2009, Ijuí.*, pp. 1–10, Porto Alegre, RS. Sociedade Brasileira de Computação.
- [Megalou & Hadzilacos, 2003] Megalou, E. e Hadzilacos, T. (2003). Semantic abstractions in the multimedia domain. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 15(1):136–160.
- [Müller et al., 2004] Müller, H., Michoux, N., Bandon, D., e Geissbuhler, A. (2004). A review of content-based image retrieval systems in medical applications-clinical benefits and future directions. *International Journal of Medical Informatics*, 73(1):1–23.
- [Mokbel et al., 2003] Mokbel, M. F., Ghanem, T. M., e Aref, W. G. (2003). Spatio-temporal access methods. *IEEE Data Engineering Bulletin*, 26(2):40–49.
- [Mokbel et al., 2004] Mokbel, M. F., Xiong, X., e Aref, W. G. (2004). Sina: scalable incremental processing of continuous queries in spatio-temporal databases. In *SIGMOD*, pp. 623–634, Paris, France. ACM.
- [Montoya-Zegarra et al., 2008] Montoya-Zegarra, J. A., Beeck, J., Leite, N. J., da Silva Torres, R., e Falcão, A. X. (2008). Combining global with local texture information for image retrieval applications. In *ISM*, pp. 148–153.
- [Mouratidis et al., 2005] Mouratidis, K., Hadjieleftheriou, M., e Papadias, D. (2005). Conceptual partitioning: An efficient method for continuous nearest neighbor monitoring. In *SIGMOD*, pp. 634–645, Baltimore, Maryland, USA. ACM.
- [Nascimento et al., 1999] Nascimento, M. A., Silva, J. R. O., e Theodoridis, Y. (1999). Evaluation of access structures for discretely moving points. In *STDBM '99: Proceedings of the International Workshop on Spatio-Temporal Database Management*, pp. 171–188. London, UK.
- [Navathe & Ahmed, 1989] Navathe, S. B. e Ahmed, R. (1989). A temporal relational model and a query language. *Inf. Sci.*, 49(1-3):147–175.
- [Ooi et al., 2002] Ooi, B. C., Tan, K. L., e Yu, C. (2002). Frequent update and efficient retrieval: an oxymoron on moving object indexes? In *WISEW '02: Proceedings of the Third International Conference on Web Information Systems Engineering (Workshops) - (WISEW'02)*, pp. 3–12. Washington, DC, USA.
- [Ozsoyoglu & Snodgrass, 1995] Ozsoyoglu, G. e Snodgrass, R. T. (1995). Temporal and real-time databases: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 7(4):513–532.

-
- [Papadias et al., 2000] Papadias, D., Mamoulis, N., e Delis, V. (2000). Approximate spatio-temporal retrieval. *ACM Transactions on Information Systems (TOIS)*, 19(1):53–96.
- [Papadias et al., 1995] Papadias, D., Theodoridis, Y., Sellis, T. K., e Egenhofer, M. J. (1995). Topological relations in the world of minimum bounding rectangles: A study with r-trees. In Carey, M. J. e Schneider, D. A., editors, *ACM SIGMOD International Conference on Management of Data*, pp. 92–103, San Jose, CA. ACM Press.
- [Papadopoulos et al., 2003] Papadias, D., Zhang, J., Mamoulis, N., e Tao, Y. (2003). Query processing in spatial network databases. In *VLDB*, pp. 802–813.
- [Papadopoulos et al., 2007] Papadopoulos, S., Sacharidis, D., e Mouratidis, K. (2007). Continuous medoid queries over moving objects. In *SSTD*, pp. 38–56, Boston, MA, USA. Springer.
- [Park & Kim, 2003] Park, D.-J. e Kim, H.-J. (2003). An enhanced technique for k-nearest neighbor queries with non-spatial selection predicates. *Multimedia Tools and Applications Journal (MTAJ)*, 19(1):79–103.
- [Patel et al., 2004] Patel, J. M., Chen, Y., e Chakka, V. P. (2004). Stripes: an efficient index for predicted trajectories. In *SIGMOD '04: Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pp. 635–646. New York, NY, USA.
- [Patella & Ciaccia, 2009] Patella, M. e Ciaccia, P. (2009). Approximate similarity search: A multi-faceted problem. *J. of Discrete Algorithms*, 7(1):36–48.
- [Paterlini et al., 2005] Paterlini, A. A., de Faria, R. F. T., Razente, H. L., Traina Jr., C., e Traina, A. J. M. (2005). Fastmapdb: Uma ferramenta para visualização em sgbdrs com suporte à filtragem e seleção visual dos dados. In Brayner, n. e Dorneles, C. F., editors, *2ª Sessão de Demos em Banco de Dados - junto ao 20 Simpósio Brasileiro de banco de Dados*, pp. 1–6, Uberlândia, MG.
- [Pelanis et al., 2006] Pelanis, M., Šaltenis, S., e Jensen, C. S. (2006). Indexing the past, present, and anticipated future positions of moving objects. *ACM Trans. Database Syst.*, 31(1):255–298.
- [Petrakis & Faloutsos, 1997] Petrakis, E. G. e Faloutsos, C. (1997). Similarity searching in medical image databases. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 9(3):435–447.
- [Petrakis et al., 2002] Petrakis, E. G., Faloutsos, C., e Lin, K.-I. D. (2002). Imagemap: An image indexing method based on spatial similarity. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 14(5):979–987.
- [Pfoser, 2002] Pfoser, D. (2002). Indexing the trajectories of moving objects. *IEEE Data Eng. Bull.*, 25(2):3–9.
- [Pfoser et al., 2000] Pfoser, D., Jensen, C. S., e Theodoridis, Y. (2000). Novel approaches in query processing for moving object trajectories. In *The VLDB Journal*, pp. 395–406.

- [Povinelli & Feng, 2003] Povinelli, R. J. e Feng, X. (2003). A new temporal pattern identification method for characterization and prediction of complex time series events. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 15(2):339–352.
- [Praing & Schneider, 2007] Praing, R. e Schneider, M. (2007). Modeling historical and future movements of spatio-temporal objects in moving objects databases. In *CIKM '07: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pp. 183–192, New York, NY, USA. ACM.
- [Razente, 2004] Razente, H. L. (2004). *Suporte à Análise Visual em Processos de Redução de Dimensionalidade em Sistemas de Bancos de Dados para Data Mining*. Dissertação de mestrado, Universidade de São Paulo.
- [Roddick & Spiliopoulou, 1999] Roddick, J. F. e Spiliopoulou, M. (1999). A bibliography of temporal, spatial, and spatio-temporal data mining research. *ACM SIGKDD Explorations*, 1(1):34–38.
- [Rose & Segev, 1991] Rose, E. e Segev, A. (1991). Toodm - a temporal object-oriented data model with temporal constraints. In *Proceedings of the 10th International Conference on Entity-Relationship Approach*, pp. 205–229. ER Institute.
- [Roussopoulos et al., 1995] Roussopoulos, N., Kelley, S., e Vincent, F. (1995). Nearest neighbor queries. In Carey, M. J. e Schneider, D. A., editors, *ACM SIGMOD International Conference on Management of Data*, pp. 80–91, San Jose, CA. ACM Press.
- [Rui et al., 1998] Rui, Y., Huang, T., Ortega, M., e Mehrotra, S. (1998). Relevance feedback: A power tool for interactive content-based image retrieval. In *Circuits and Systems for Video Technology (Special Issue on Segmentation, Description, and Retrieval of Video Content)*, pp. 644–655, San Jose, CA. IEEE Computer Society.
- [Rui & Huang, 2000] Rui, Y. e Huang, T. S. (2000). Optimizing learning in image retrieval. In *CVPR*, pp. 1236–.
- [Sahinalp et al.,] Sahinalp, S. C., Tasan, M., Macker, J., e Özsoyoglu, Z. M. Distance based indexing for string proximity search. In *IEEE International Conference on Data Engineering (ICDE'2003)*, pp. 125–136, Bangalore, India. IEEE Computer Society.
- [Saltenis et al., 2000] Saltenis, S., Jensen, C. S., Leutenegger, S. T., e Lopez, M. A. (2000). Indexing the positions of continuously moving objects. In *SIGMOD '00: Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pp. 331–342. New York, NY, USA.
- [Samet, 2003] Samet, H. (2003). Depth-first k-nearest neighbor finding using the maxnearestdist estimator. In *12th International Conference on Image Analysis and Processing (IEEE ICIAP03)*, pp. 486–491, Mantova, Italy. IEEE Computer Society.
- [Santos Filho et al., 2001] Santos Filho, R. F., Traina, A. J. M., Traina Jr., C., e Faloutsos, C. (2001). Similarity search without tears: The omni family of all-purpose access methods. In *IEEE International Conference on Data Engineering (ICDE)*, pp. 623–630, Heidelberg, Germany. IEEE Computer Society.

-
- [Schroeder, 1991] Schroeder, M. (1991). *Fractals, Chaos, Power Laws*. W. H. Freeman, New York, 6 edition.
- [Searcoid, 2006] Searcoid, M. (2006). *Metric Spaces (Springer Undergraduate Mathematics Series)*. Springer.
- [Sellis, 1999] Sellis, T. K. (1999). Research issues in spatio-temporal database systems. In Güting, R. H., Papadias, D., e Lochovsky, F. H., editors, *6th International Symposium in Advances in Spatial Databases, SSD'99*, v. 1651 of *Lecture Notes in Computer Science*, pp. 5–11, Hong Kong, China. Springer Verlag.
- [Sellis et al., 1987] Sellis, T. K., Roussopoulos, N., e Faloutsos, C. (1987). The r+-tree: A dynamic index for multi-dimensional objects. In Stocker, P. M., Kent, W., e Hammersley, P., editors, *International Conference on Very Large Databases (VLDB)*, pp. 507–518, Brighton, England. Morgan Kaufmann.
- [Silva et al., 2009a] Silva, M. P. d., Traina, A. J. M., Azevedo-Marques, P. M. d., Felipe, J. C., e Traina, Caetano, J. (2009a). Including the perceptual parameter to tune the retrieval ability of pulmonary cbir systems. In *22th IEEE Intl. Symposium on Computer-Based Medical Systems (CBMS 2009)*, pp. 1–8, Albuquerque, NM, EUA. IEEE Computer Society.
- [Silva et al., 2009b] Silva, Y. N., Xiong, X., e Aref, W. G. (2009b). The rum-tree: supporting frequent updates in r-trees using memos. *The VLDB Journal*, 18(3):719–738.
- [Sistla et al., 1997] Sistla, A. P., Wolfson, O., Chamberlain, S., e Dao, S. (1997). Modeling and querying moving objects. In *ICDE '97: Proceedings of the Thirteenth International Conference on Data Engineering*, pp. 422–432. Washington, DC, USA.
- [Skjellaug, 1997] Skjellaug, B. (1997). Temporal data: Time and relational databases. Research Report 246, Department of Informatics, University of Oslo.
- [Skopal & Hoksza, 2007] Skopal, T. e Hoksza, D. (2007). Improving the performance of m-tree family by nearest-neighbor graphs. In *ADBIS*, v. 4690 of *Lecture Notes in Computer Science*, pp. 172–188. Springer.
- [Skopal & Lokoč, 2009] Skopal, T. e Lokoč, J. (2009). New dynamic construction techniques for M-tree. *J. of Discrete Algorithms*, 7(1):62–77.
- [Skopal et al., 2003] Skopal, T., Pokorný, J., Krátký, M., e Snásel, V. (2003). Revisiting m-tree building principles. In *ADBIS*, v. 2798 of *Lecture Notes in Computer Science*, pp. 148–162, Dresden, Germany. Springer.
- [Smeulders et al., 2000] Smeulders, A. W. M., Worring, M., Santini, S., Gupta, A., e Jain, R. (2000). Content-based image retrieval at the end of the early years. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 22(12):1349–1380.
- [Snodgrass, 1995] Snodgrass, R. T. (1995). *The TSQL2 Temporal Query Language*. Kluwer Academic Publishers.

- [Song & Roussopoulos, 2001] Song, Z. e Roussopoulos, N. (2001). K-nearest neighbor search for moving query point. In *SSTD '01: Proceedings of the 7th International Symposium on Advances in Spatial and Temporal Databases*, pp. 79–96, London, UK. Springer-Verlag.
- [Stehling et al., 2002] Stehling, R. O., Nascimento, M. A., e Falcão, A. X. (2002). A compact and efficient image retrieval approach based on border/interior pixel classification. In *International Conference on Information and Knowledge Management (CIKM)*, pp. 102 – 109, McLean, VA, USA. ACM Press.
- [Stejić et al., 2003] Stejić, Z., Takama, Y., e Hirota, K. (2003). Genetic algorithm-based relevance feedback for image retrieval using local similarity patterns. *Inf. Process. Manage.*, 39(1):1–23.
- [Sun et al., 2004] Sun, J., Papadias, D., Tao, Y., e Liu, B. (2004). Querying about the past, the present, and the future in spatio-temporal databases. In *ICDE '04: Proceedings of the 20th International Conference on Data Engineering*, pp. 202–213, Washington, DC, USA. IEEE Computer Society.
- [Talavera, 1999] Talavera, L. (1999). Feature selection as a preprocessing step for hierarchical clustering. In *The Sixteenth International Conference on Machine Learning*, pp. 389–397. Morgan Kaufmann.
- [Tansel et al., 1993] Tansel, A. U., Clifford, J., Gadia, S., Jajodia, S., Segev, A., e Snodgrass, R., editors (1993). *Temporal databases: theory, design, and implementation*. Benjamin-Cummings Publishing Co., Inc., Redwood City, CA, USA.
- [Tao & Papadias, 2002] Tao, Y. e Papadias, D. (2002). Time-parameterized queries in spatio-temporal databases. In *SIGMOD '02: Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, pp. 334–345. New York, NY, USA.
- [Tao & Papadias, 2003] Tao, Y. e Papadias, D. (2003). Spatial queries in dynamic environments. *ACM Trans. Database Syst.*, 28(2):101–139.
- [Tao et al., 2002] Tao, Y., Papadias, D., e Shen, Q. (2002). Continuous nearest neighbor search. In *International Conference on Very Large Databases (VLDB)*, pp. 287–298, Hong Kong, China. Morgan Kaufmann.
- [Tao et al., 2003] Tao, Y., Papadias, D., e Sun, J. (2003). The tpr*-tree: An optimized spatio-temporal access method for predictive queries. In Freytag, J. C., Lockemann, P. C., Abiteboul, S., Carey, M. J., Selinger, P. G., e Heuer, A., editors, *International Conference on Very Large Databases (VLDB)*, pp. 790–801, Berlin, Germany. Morgan Kaufmann.
- [Tao et al., 2009] Tao, Y., Yi, K., Sheng, C., e Kalnis, P. (2009). Quality and efficiency in high dimensional nearest neighbor search. In *SIGMOD '09: Proceedings of the 35th SIGMOD international conference on Management of data*, pp. 563–576, New York, NY, USA. ACM.
- [Tao et al., 2006] Tao, Y., Yiu, M. L., e Mamoulis, N. (2006). Reverse nearest neighbor search in metric spaces. *IEEE Trans. on Knowl. and Data Eng.*, 18(9):1239–1252.

-
- [Tayeb et al., 1998] Tayeb, J., Ulusoy, z., e Wolfson, O. (1998). A quadtree-based dynamic attribute indexing method. *The Computer Journal*, 41(3):185–200.
- [Theodoulidis et al., 1996] Theodoulidis, B., Ait-Braham, A., Hanias, K., e Kakoudakis, I. (1996). Review of temporal object-oriented approaches. Technical Report TR-96-1, TimeLab - TimeLab, University of Manchester.
- [Torres et al., 2009] Torres, R. d. S., Falcão, A. X., Gonçalves, M. A., Papa, Jo a. P., Zhang, B., Fan, W., e Fox, E. A. (2009). A genetic programming framework for content-based image retrieval. *Pattern Recogn.*, 42(2):283–292.
- [Torres & Falcão, 2006] Torres, R. d. S. e Falcão, A. X. (2006). Content-based image retrieval: Theory and applications. *rita* 13(2):. *Revista de Informática Teórica e Aplicada - RITA*, 13(2):61–185.
- [Traina & Traina Jr., 2003] Traina, A. J. M. e Traina Jr., C. (2003). Similarity search in multimedia databases. In Furht, B. e Marques, O., editors, *Handbook of Video Databases - Design and Applications*, v. 1, pp. 711–738. CRC Press.
- [Traina et al., 2001] Traina, A. J. M., Traina Jr., C., Barioni, M. C. N., Botelho, E., e Bueno, R. (2001). Visualização de dados em sistemas de bancos de dados relacionais. In Mattoso, M. L. d. Q. e Xexéo, G., editors, *Simpósio Brasileiro de Bancos de Dados (SBBD)*, pp. 95–109, Rio de Janeiro, RJ. SBC.
- [Traina et al., 2003] Traina, A. J. M., Traina Jr., C., Bueno, J. M., Chino, F. J. T., e Marques, P. M. d. A. (2003). Efficient content-based image retrieval through metric histograms. *World Wide Web Journal (WWWJ)*, 6(2):157–185.
- [Traina et al., 2007] Traina, Jr., C., Filho, R. F., Traina, A. J., Vieira, M. R., e Faloutsos, C. (2007). The omni-family of all-purpose access methods: a simple and effective way to make similarity search more efficient. *The VLDB Journal*, 16(4):483–505.
- [Traina Jr. et al., 2000a] Traina Jr., C., Traina, A. J. M., e Faloutsos, C. (2000a). Distance exponent: a new concept for selectivity estimation in metric trees. In *IEEE International Conference on Data Engineering (ICDE)*, pp. 195 – 195, San Diego - CA. IEEE CS Press.
- [Traina Jr. et al., 2002a] Traina Jr., C., Traina, A. J. M., Faloutsos, C., e Seeger, B. (2002a). Fast indexing and visualization of metric datasets using slim-trees. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 14(2):244–260.
- [Traina Jr. et al., 2002b] Traina Jr., C., Traina, A. J. M., Santos Filho, R. F., e Faloutsos, C. (2002b). How to improve the pruning ability of dynamic metric access methods. In *International Conference on Information and Knowledge Management (CIKM)*, pp. 219–226, McLean, VA, USA. ACM Press.
- [Traina Jr. et al., 2000b] Traina Jr., C., Traina, A. J. M., Seeger, B., e Faloutsos, C. (2000b). Slim-trees: High performance metric trees minimizing overlap between nodes. In Zaniolo, C., Lockemann, P. C., Scholl, M. H., e Grust, T., editors, *International Conference on Extending Database Technology (EDBT)*, v. 1777 of *Lecture Notes in Computer Science*, pp. 51–65, Konstanz, Germany. Springer Verlag.

- [Traina Jr. et al., 2000c] Traina Jr., C., Traina, A. J. M., Wu, L., e Faloutsos, C. (2000c). Fast feature selection using fractal dimension. In Medeiros, C. M. B. e Becker, K., editors, *Brazilian Symposium on Databases (SBBD)*, pp. 158–171, João Pessoa, PB.
- [Tran et al., 2009] Tran, Q. T., Taniar, D., e Safar, M. (2009). Reverse k nearest neighbor and reverse farthest neighbor search on spatial networks. pp. 353–372.
- [Tseng, 1999] Tseng, Y.-H. (1999). Content-based retrieval for music collections. In *International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 176–182, Berkeley, CA. ACM Press.
- [U et al., 2007] U, L. H., Mamoulis, N., e Yiu, M. L. (2007). Continuous monitoring of exclusive closest pairs. In *SSTD*, pp. 1–19, Boston, MA, USA. Springer.
- [Uhlmann, 1991] Uhlmann, J. K. (1991). Satisfying general proximity/similarity queries with metric trees. *Information Processing Letters*, 40(4):175–179.
- [Vespa et al., 2007] Vespa, T. G., Traina, Caetano, J., e Traina, A. J. M. (2007). Bulk-loading dynamic metric access methods. In *22º Simpósio Brasileiro de Bases de Dados (SBBD '07)*, v. 1, pp. 160–173, João Pessoa, PB. SBC.
- [Vieira et al., 2004] Vieira, M. R., Traina Jr., C., Traina, A. J. M., e Chino, F. J. T. (2004). Dbm-tree: A dynamic metric access method sensitive to local density data. In Lifschitz, S., editor, *Brazilian Symposium on Databases (SBBD)*, v. 1, pp. 33–47, Brasília, DF. SBC.
- [Volnyansky & Pestov, 2009] Volnyansky, I. e Pestov, V. (2009). Curse of dimensionality in pivot-based indexes. *CoRR*, abs/0906.0391.
- [Wang et al., 2003] Wang, T., Rui, Y., Hu, S.-m., e Sun, J.-G. (2003). Adaptive tree similarity learning for image retrieval. *Multimedia Systems*, 9(2):131 – 143.
- [Wolfson et al., 1998] Wolfson, O., Xu, B., Chamberlain, S., e Jiang, L. (1998). Moving objects databases: Issues and solutions. In *SSDBM '98: Proceedings of the 10th International Conference on Scientific and Statistical Database Management*, pp. 111–122. Washington, DC, USA.
- [Xia et al., 2005] Xia, C., Hsu, W., e Lee, M. L. (2005). Erknn: efficient reverse k-nearest neighbors retrieval with local knn-distance estimation. In *CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management*, pp. 533–540, New York, NY, USA. ACM Press.
- [Xiong & Aref, 2006] Xiong, X. e Aref, W. G. (2006). R-trees with update memos. In *ICDE '06: Proceedings of the 22nd International Conference on Data Engineering* 22, p., Washington, DC, USA. IEEE Computer Society.
- [Yi, 2004] Yi, B. (2004). *Um Modelo de Dados para Objetos Móveis*. Dissertação de mestrado, Universidade Estadual de Campinas.
- [Yi & Medeiros, 2002] Yi, B. e Medeiros, C. B. (2002). Um modelo de dados para objetos móveis. In *GeoInfo*, Caxambu, MG.

-
- [Yianilos, 1993] Yianilos, P. N. (1993). Data structures and algorithms for nearest neighbor search in general metric spaces. In *Fourth Annual ACM/SIGACT-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 311–321, Austin, TX.
- [Yianilos, 1999] Yianilos, P. N. (1999). Excluded middle vantage point forests for nearest neighbor search. Technical report, NEC Research Institute.
- [Yoshitaka & Ichikawa, 1999] Yoshitaka, A. e Ichikawa, T. (1999). A survey on content-based retrieval for multimedia databases. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 11(1):81–93.
- [Zhang & Lu, 2004] Zhang, D. e Lu, G. (2004). Review of shape representation and description techniques. *Pattern Recognition*, 37(1):1–19.
- [Zhang et al., 2008a] Zhang, R., Lin, D., Ramamohanarao, K., e Bertino, E. (2008a). Continuous intersection joins over moving objects. In *ICDE*, pp. 863–872, Cancun, Mexico. IEEE.
- [Zhang et al., 2008b] Zhang, Z., Yang, Y., Tung, A. K. H., e Papadias, D. (2008b). Continuous k-means monitoring over moving objects. *IEEE Trans. on Knowl. and Data Eng.*, 20(9):1205–1216.
- [Zhou et al., 2003] Zhou, X., Wang, G., Yu, J. X., e Yu, G. (2003). M+-tree: a new dynamical multidimensional index for metric spaces. In *ADC '03: Proceedings of the 14th Australasian database conference*, pp. 161–168, Darlinghurst, Australia, Australia. Australian Computer Society, Inc.
- [Zhou et al., 2005] Zhou, X., Wang, G., Zhou, X., e Yu, G. (2005). Bm⁺-tree: A hyperplane-based index method for high-dimensional metric spaces. In *International Conference Database Systems for Advanced Applications, (DASFAA)*, v. 3453 of *Lecture Notes in Computer Science*, pp. 398–409. Springer.
- [Zisman, 1993] Zisman, A. (1993). *A Árvore-B e uma fronteira de implementação*. Dissertação de mestrado, Universidade de São Paulo.