UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

CÉSAR FEIJÓ NADVORNY

TM-tree: um Método de Acesso para consultas por similaridade

Dissertação apresentada como requisito parcial
para a obtenção do grau de
Mestre em Ciência da Computação

Prof. Dr. Carlos Alberto Heuser
Orientador

Porto Alegre, abril de 2005

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Nadvorny, César Feijó

TM-tree: um Método de Acesso para consultas por similaridade / César Feijó Nadvorny. – Porto Alegre: PPGC da UFRGS, 2005.

49 f.: il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2005. Orientador: Carlos Alberto Heuser.

1. Consulta por similaridade. 2. Árvores métricas. 3. Método de Acesso Métrico. 4. Indexação. I. Heuser, Carlos Alberto. II. Título.

AGRADECIMENTOS

Agradeço a meu orientador, Prof. Carlos Heuser, por me mostrar os caminhos a seguir, pela paciência durante o mestrado e por ter me ensinado Fundamentos de Banco de Dados, onde começou meu interesse por essa área.

Agradeço a meus maiores professores, meus pais, por tudo que eles têm me ensinado.

Agradeço a meus colegas do grupo de banco de dados pelas várias prévias que assistiram, pelas sugestões para melhorar o trabalho, pelo apoio. Especialmente, gostaria de agradecer a Vanessa Braganholo pelas correções no texto e apresentação do artigo do SBBD'04 e a Carina Dorneles por todo acompanhamento que ela teve no meu trabalho, principalmente pelas diversas reuniões que tivemos, de onde tirei muitas idéias e esclareci muitas dúvidas.

Agradeço também à CAPES e ao CNPq pelo apoio financeiro a esse trabalho.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

árvore BSP	Binary Space Partitioning tree
árvore DBM	Density-Based Metric tree
árvore DF	Distance Fields tree
árvore GH	Generalized Hyperplane tree
árvore GNAT	Geometric Near-Neighbor Access tree
árvore TM	Twisted and Metric tree
árvore VP	Vantage-Point tree
MAE	Método de Acesso Espacial
MAM	Método de Acesso Métrico
MAP	Método de Acesso Pontual
SGBD	Sistema de Gerência de Banco de Dados
XML	eXtensible Markup Language
XPath	XML Path Language
XQuery	XML Query
W3C	World Wide Web Consortium

LISTA DE SÍMBOLOS

d	Função de distância definida como $d : \mathbb{U}^2 \rightarrow \mathbb{R}^+$.
$\mathbb{M}$	Espaço métrico.
$NN_k(q)$	Consulta pelos k vizinhos mais próximos de q (k-Nearest Neighbor Query).
o	Objeto qualquer pertencente a $\mathbb{U}$.
$\mathbb{R}^+$	Conjunto dos números reais não negativos.
RQ(q, r)	Consulta por abrangência (Range Query) onde q é o objeto consulta e r o raio de cobertura da consulta.
t	Função de distorção.
$\mathbb{T}$	Espaço distorcido. É definido pela tripla $< \mathbb{U}, d, t >$.
$\mathbb{U}$	Universo de possíveis objetos.

LISTA DE FIGURAS

RESUMO

O armazenamento de grandes quantidades de informações em bases de dados cria a necessidade de se usar Métodos de Acesso a esses dados de uma forma mais eficiente do que uma busca linear. Dessa forma, diversos Métodos de Acesso vêm sendo propostos há décadas. Desde os mais simples Métodos de Acesso como árvores B até os mais sofisticados Métodos de Acesso Métrico tem-se o mesmo objetivo: a eficiência na consulta.

Para cada tipo de dados, para cada tipo de consulta, existe uma diferente forma de acesso mais adequada. Se os dados puderem ser ordenados, pode-se usar uma árvore B. Na busca por pequenas cadeias de caracteres, pode-se utilizar uma árvore de sufixos.

Com a evolução computacional, não se quer armazenar apenas números ou pequenas seqüências de texto. Já existem diversas bases de dados muito mais complexas, como seqüências de sons, imagens ou até mesmo vídeos armazenados. A complexidade desse tipo de dados e do tipo de consulta feita em cima deles gerou a necessidade de novos Métodos de Acesso.

Os chamados *Métodos de Acesso Métrico* são estruturas capazes de acessar dados bastante complexos, como arquivos multimídia, com uma boa eficiência. Esse tipo de estrutura vem sendo estudada há muitos anos, mas a primeira delas realmente eficaz foi a árvore M. Depois dela, vários outros Métodos de Acesso Métricos surgiram, como a árvore Slim, M^2, M+, DF, DBM aprimorando sua estrutura básica.

Esse trabalho propõe a árvore TM, que inova a forma como os dados são indexados, aprimorando a árvore M. Essa nova estrutura, usa o *espaço métrico* para a busca dos dados, o que é feito por todos Métodos de Acesso Métricos. Mas sua inovação está na forma como os dados são indexados, usando-se um espaço novo também proposto nesse trabalho, o *espaço distorcido*. Experimentos mostram uma melhora significativa na eficiência da consulta tanto em quantidade de acesso a disco quando em custo de processamento.

Palavras-chave: Consulta por similaridade, Árvores métricas, Método de Acesso Métrico, Indexação.

TM-tree: a similarity query Access Method

ABSTRACT

From the storage of great amount of information in databases arises the need of Access Methods more efficient than an ordinary linear query of the whole database. Thus, many Access Methods have been proposed for decades. Simpler Access Methods like B-trees and the more complex, Metric Access Methods have all the same objective: the query efficiency.

For each data type, for each query type, there is a different way to access the data more suitable. If we can order the data, we could use a B-tree to access it. If querying small strings is the case, then maybe a suffix tree should be used.

Today we do not want to store only numbers or small strings. There are already many database storing much more complex data like streams of sound, images, or even videos. The complexity of these type of information stored and the type of query over them have brought forth the need for more sophisticated Access Methods.

Metric Access Methods are capable of efficiently accessing very complex data, such as multimedia files. These Access Methods have been studied for many years, but the first one really effective was the M-tree. Many others Metric Access Methods have been created after it, improving its basic structure, such as the Slim-tree, M^2-tree, M+-tree, DF-tree, DBM-tree.

We propose the TM-tree. This Metric Access Method innovates the way data are indexed, improving the M-tree. This new Access Method uses the *metric space* for querying data as any other Metric Access Method does. But its distinction is in the way it indexes the data, using a new space also proposed in this work, the *twisted space*. Experiments show that the TM-tree has significant improvements concerning disk pages access cost and CPU cost.

Keywords: Similarity query, Metric trees, Metric Access Method, Indexing.

1 INTRODUÇÃO

O uso de XML para armazenamento e principalmente intercâmbio de informação já é bastante grande e tem a tendência de aumentar. Portanto, justifica-se o estudo da consulta sobre essa fonte de dados. Grandes fabricantes de SGBDs (Oracle, IBM, Microsoft) já dão suporte ao intercâmbio com XML (BRAGANHOLO; FEIJÓ, 2002).

Sobre fontes XML, pode-se fazer consultas usando linguagens como XPath (W3C, 1999) ou XQuery (W3C, 2002). No entanto, essas linguagens permitem apenas consultas booleanas, ou seja, o resultado encontrado é exatamente o que foi especificado na consulta, ou ele não será aceito como resultado. Por exemplo, se o nome "Ramez Elmasri" for dado como entrada em uma consulta, mas no banco XML esse nome estiver registrado como "Ramez A. Elmasri" nenhum resultado será retornado, pois nenhum registro é exatamente igual ao dado de entrada da consulta.

Várias pessoas de lugares diferentes, de grupos diferentes podem entrar com dados em uma certa base de dados XML. Juntamente a isso, a flexibilidade da estrutura XML contribui para que os dados em uma base XML estejam bastante heterogêneos. Alguém pode registrar o nome "Ramez Elmasri", e outra pessoa pode registrar o nome "Ramez A. Elmasri". Poderia ainda acontecer de alguém registrar o nome do autor no caminho `book/author` e outra pessoa registrar no caminho `book/authors/author`, o que tornaria a consulta ainda mais complexa.

Uma das formas de consulta sobre bases XML deve ser por similaridade, ou seja, não está sendo procurada a informação que é exatamente igual ao que usuário solicitou, mas sim registros com um certo grau de similaridade. Os registros podem ser mais similares ou menos similares ao que está sendo procurado, ao contrário da consulta booleana, onde há um particionamento dos registros em iguais ou diferentes do que o usuário quer achar.

Para uma pequena base de dados XML, não há necessidade do uso de *Métodos de Acessos* (MAs) (CHAVEZ et al., 2001) para agilizar a consulta. É possível colocar toda a base na memória e percorrê-la inteira com uma boa performance. Contudo, se a base for grande, colocá-la inteira na memória torna-se inviável. Percorrê-la inteira torna-se muito demorado, principalmente, quando a consulta é mais complexa como a consulta por similaridade. Portanto, para bases maiores é indispensável o uso de algum MA.

Árvores B (BAYER; MCCREIGHT, 1972) e outras derivadas são amplamente usadas para o acesso a base de dados tradicionais onde são feitas apenas consultas booleanas. No entanto, para indexar uma base para execução de consulta por similaridade, faz-se necessário outros tipos de Métodos de Acesso.

A consulta por similaridade é bastante comum em bases multimídias (armaze-

namento de imagens, vídeos, sons). Para a consulta por similaridade nessas bases, usa-se Métodos de Acesso Pontuais (MAPs) (GAEDE; GÜNTHER, 1998) e Métodos de Acesso Espaciais (MAEs) (GAEDE; GÜNTHER, 1998) quando é possível representar o objeto a ser indexado como um vetor de características. Métodos de Acesso Métricos (MAMs) (CHAVEZ et al., 2001) são mais sofisticados, permitindo a indexação de objetos simplesmente usando-se uma função de comparação entre eles. Esses Métodos de Acesso serão detalhados no Capítulo 2.

Entre os MAMs, um importante representante é a árvore M (CIACCIA; PATELLA; ZEZULA, 1997). Esse MA foi o primeiro MAM a preocupar-se com importantes questões como acesso a memória secundária, dinamismo e escalabilidade, servindo dessa forma como base para MAMs mais sofisticados que foram desenvolvidos depois como a árvore Slim (TRAINA JÚNIOR et al., 2000), M^2 (CIACCIA; PATELLA, 2000) e DBM (VIEIRA, 2004).

Com o objetivo de indexar bases XML, analisou-se o desempenho do MAM árvore M sobre pequenos campos textuais. No entanto, verificou-se que ele pode ter sua estrutura melhorada otimizando-se a maneira como a árvore insere objetos.

A árvore M e árvores derivadas baseiam sua estrutura no agrupamento de objetos semelhantes. Esses grupos devem se manter o menor possível para evitar ao máximo a sobreposição deles. Quando não se consegue fazer isso, o desempenho da busca pode ser bastante prejudicado.

A proposta desse trabalho é um novo MAM derivado da árvore M, a árvore TM, que consegue agrupar os objetos mais adequadamente nos casos em que a árvore M apresentou deficiência. Isso é feito com o uso do *espaço distorcido*, também apresentado nesse trabalho, com o qual consegue-se fazer uma análise melhor de como se deve agrupar os objetos. Verificou-se com experimentos que o desempenho da busca com a árvore TM realmente apresentou melhoras em relação à árvore M.

1.1 Organização do Documento

- No Capítulo 2 é apresentado o conceito de consulta por similaridade e são apresentados diversos Métodos de Acesso. Eles são divididos em três grupos, Métodos de Acesso Pontuais (MAPs), Métodos de Acesso Vetoriais (MAVs) e Métodos de Acesso Métricos (MAMs). Esse último é onde esse trabalho se concentra.

- No Capítulo 3 são apresentados experimentos que foram feitos usando umas dessas MAMs, a Árvore M, aplicando-a a bases XML. Aponta-se então algumas falhas da Árvore M em determinadas situações.

- No Capítulo 4 é apresentada a árvore TM, que é um aprimoramento da árvore M, solucionando-se as deficiências apresentadas no Capítulo 3.

- No Capítulo 5 são feitas as considerações finais e indicações para trabalhos futuros.

2 TRABALHOS RELACIONADOS

2.1 Consulta por Similaridade e Espaço

Em SGBDs tradicionais, a consulta aos dados é feita por comparação de igualdade, ou seja, existe um particionamento claro dos objetos do banco em iguais ou diferentes do que o usuário requisitou. No entanto, para alguns tipos de objetos, esse tipo de consulta não é adequada.

Por exemplo, em um banco de imagens, para se fazer uma consulta, poderia se fornecer uma imagem e pedir imagens semelhantes. Esse tipo de consulta difere bastante de uma consulta tradicional, porque o sistema de consulta terá que retornar para o usuário objetos que estejam dentro de um parâmetro aceitável de similaridade para o usuário. Calcular se um objeto pertence ou não ao conjunto de objetos que o usuário deseja é muito mais complexo do que em uma consulta por igualdade.

Em uma consulta por similaridade o usuário deve fornecer um *objeto consulta* (CHAVEZ et al., 2001), indicando que quer recuperar objetos semelhantes a esse que forneceu. Além disso, o usuário deve fornecer um parâmetro indicando o grau de similaridade que os objetos recuperados devem ter em relação ao objeto consulta.

Existem basicamente duas formas de o usuário indicar o grau de similaridade que quer, definindo-se assim dois principais tipos de consulta por similaridade (CHAVEZ et al., 2001):

Consulta por Abrangência (*Range Query*): $RQ(q, r)$ recupera todos objetos que estão a uma distância máxima r do objeto consulta q.

Consulta pelos k Vizinhos Mais Próximos (*k-Nearest Neighbor Query*): $NN_k(q)$ recupera os k objetos mais próximos ao objeto consulta q.

A figura 2.1 mostra graficamente como estas duas consultas são feitas.

Para se fazer uma consulta por abrangência, é necessário comparar dois objetos (o objeto consulta e cada um dos objetos da base) e computar a distância entre eles. Se essa distância é aceitável, retorna-se o objeto do banco para o usuário. A consulta pelos vizinhos mais próximos é equivalente, basta ajustar o tamanho da distancia máxima r na $RQ(q, r)$ para que esta retorne exatamente o número de objetos necessários.

Uma forma de comparar objetos de uma base de dados é extrair características de cada objetos gerando assim um vetor de características (*feature vector*) para cada objeto. A comparação é feita em cima desse vetor de característica (PATELLA, 1999). Ou seja, os objetos estão sendo imergidos em um espaço de n dimensões (n é

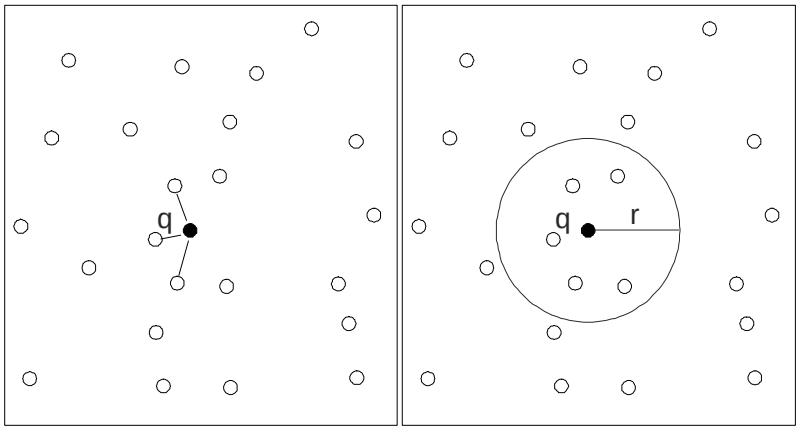


Figura 2.1: Exemplo de um consulta $NN_k(q)$ (esquerda) e $RQ(q, r)$ (direita).

o número de características do vetor) e a comparação na verdade é feita nesse espaço vetorial. Por exemplo, se esse espaço é de 2 dimensões poderia se usar a conhecida distância euclidiana para tal comparação. É nessa idéia que os Métodos de Acesso Pontuais e Espaciais (GAEDE; GÜNTHER, 1998) se baseiam.

Essa comparação, usando-se um vetor de características, pressupõe que se pode representar os objetos da base em um vetor de características, imergindo-os no espaço vetorial. No entanto, pode ser bastante difícil, ou mesmo impossível, representar determinados objetos em um espaço vetorial. Um exemplo bem simples de um caso em que isso falha é a comparação de palavras em um texto. Não se consegue representar palavras com números que eficientemente representem a distância entre elas. Portanto, não se consegue imergir as palavras em um espaço vetorial para se poder usar uma função da geometria analítica para compará-las. Para resolver problemas desse tipo, usa-se uma função que compara diretamente um objeto com outro, retornando a distância entre eles. Essa função é específica para cada domínio de objetos.

Seja $\mathbb{U}$ o conjunto de todos possíveis objetos de um determinado domínio, então

$$d : \mathbb{U}^2 \rightarrow \mathbb{R}^+$$

é a função de distância que compara os objetos de $\mathbb{U}$. Se o valor retornado for zero (0), significa que os dois objetos são idênticos; quanto maior o valor retornado maior é a diferença entre os dois objetos.

Qualquer função de comparação deve obedecer as seguintes propriedades (CHAVEZ et al., 2001):

$\forall o_1, o_2, o_3 \in \mathbb{U}$,

positividade: $d(o_1, o_2) \geq 0$

simetria: $d(o_1, o_2) = d(o_2, o_1)$

reflexividade: $d(o_1, o_1) = 0$

positividade estrita: $o_1 \neq o_2 \Rightarrow d(o_1, o_2) > 0$

Se a função de distância d também satisfaz a propriedade da desigualdade triangular:

desigualdade triangular: $d(o_1, o_2) + d(o_2, o_3) \geq d(o_1, o_3)$

então ela é dita *métrica* e o par $(\mathbb{U}, d)$ define um *espaço métrico* $\mathbb{M}$.

Espaços métricos são úteis para comparação de objetos que não podem ser imergidos no espaço vetorial.

2.2 Métodos de Acesso

Imagens podem ter suas propriedades (cor, textura, forma) mapeadas em pontos em um espaço multidimensional. Nesse caso, cada propriedade seria representada em uma dimensão. Para cada figura, é atribuído um valor para cada uma de suas propriedades, de forma que a figura teria vários valores associados. Por exemplo, se uma determinada figura tivesse sua cor mapeada ao valor 5, sua textura mapeada ao valor 1 e sua forma mapeada ao valor 2, a figura como um todo poderia ser mapeada para o ponto $(5, 1, 2)$ num espaço cujas dimensões são cor, textura e forma como mostra a figura 2.2. Dessa forma, pode-se usar métricas de distância como Euclidiana, Minkowski, Manhattan, etc. para calcular o quão similar uma imagem é a outra (BIMBO, 1999).

Esse tipo de modelagem do objeto em ponto é usado por *Métodos de Acesso Pontual* (Point Access Methods - PAMs). Entre esses métodos, estão as árvores k-d (BENTLEY, 1975, 1979), k-d-B (ROBINSON, 1981), BSP (FUCHS; KEDEM; NAYLOR, 1980; FUCHS; ABRAM; GRANT, 1983), quadtree (FINKEL; BENTLEY, 1974), LSD (HENRICH; SIX; WIDMAYER, 1989), hB (LOMET; SALZBERG, 1989, 1990), BV (FREESTON, 1995).

Entretanto, em alguns casos, mapear objetos em um ponto pode não ser uma boa alternativa. Por exemplo, bancos de dados geográficos contêm polígonos em geral, e dados de sistemas CAD contêm poliedros. Nesses casos que se quer indexar imagens vetoriais, se usa *Métodos de Acesso Espacial* (Space Access Methods - SAMs) (GAEDE; GÜNTHER, 1998). Exemplos de árvores desse método são: R (GUTTMAN, 1984), R+ (SELLIS; ROUSSOPOULOS; FALOUTSOS, 1987), R* (BECKMANN et al., 1990), X (BERCHTOLD; KEIM; KRIEGEL, 1996), P (JAGADISH, 1990; SCHIWIETZ, 1993), SKD (OOI, 1987), GBD (OHSAWA; SAKAUCHI, 1990).

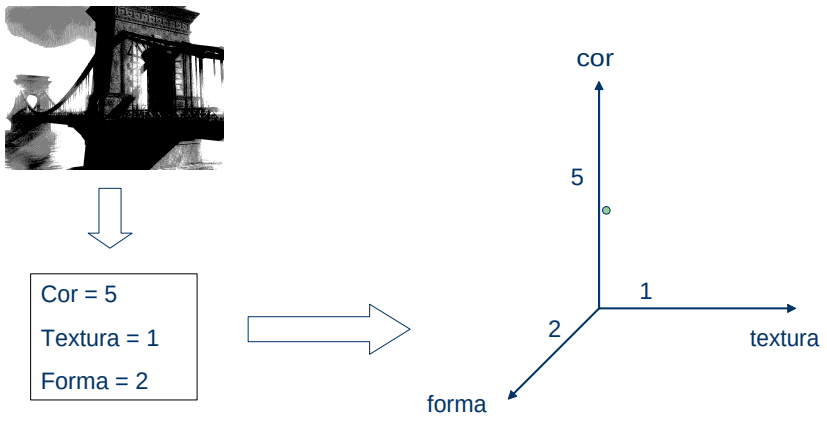


Figura 2.2: Mapeamento de uma figura em um ponto no espaço.

Existem ainda casos em que não é adequado mapear o objeto em um espaço ve-

torial de características, em uma posição absoluta. Mapear objetos em uma posição absoluta significa que existe um ponto "zero" no espaço, um ponto de referência absoluto. Muitos conjuntos de objetos simplesmente não apresentam esse ponto "zero". Mesmo assim, existe uma noção de similaridade entre os objetos do universo. Para esses casos, usa-se *Métodos de Acesso Métricos* (Metric Access Methods - MAMs), onde cada objeto é indexado relativamente aos outros objetos do universo e não em uma posição absoluta no espaço (BÖHM; BERCHTOLD; KEIM, 2001). Esse método pode usar apenas uma função de distância para classificar os objetos a serem indexados. Portanto, ao contrário dos métodos anteriores, não se pode fazer operações primitivas como adição, subtração ou qualquer operação geométrica (TRAINA JÚNIOR et al., 2000). Representantes desse Método de Acesso são árvores Burkhard-Keller (BURKHARD; KELLER, 1973), VP (UHLMANN, 1991), GH (UHLMANN, 1991), M (CIACCIA; PATELLA; ZEZULA, 1997), Slim (TRAINA JÚNIOR et al., 2000).

2.2.1 Métodos de Acesso Pontual

2.2.1.1 Árvore k-d

A árvore k-d (BENTLEY, 1975, 1979) (figura 2.3) é uma árvore binária. Dado um certo espaço com alguns pontos, é escolhido um dos pontos pelo qual passará uma linha ortogonal a um dos eixos do espaço que dividirá o espaço em duas partições. Tenta-se escolher o ponto que criará partições equilibradas, ou seja, com a quantidade de pontos parecida. Esse ponto torna-se a raiz da árvore. Os pontos de uma partição ficam na subárvore esquerda e os pontos da outra partição na subárvore direita. Esse processo se repetirá para cada uma das partições até que todos pontos tenham sido usados para particionar o espaço. As linhas que dividem o espaço são ortogonais alternadamente aos eixos do espaço. Por exemplo, se a primeira linha é ortogonal ao eixo x, então a próxima linha será ortogonal ao eixo y.

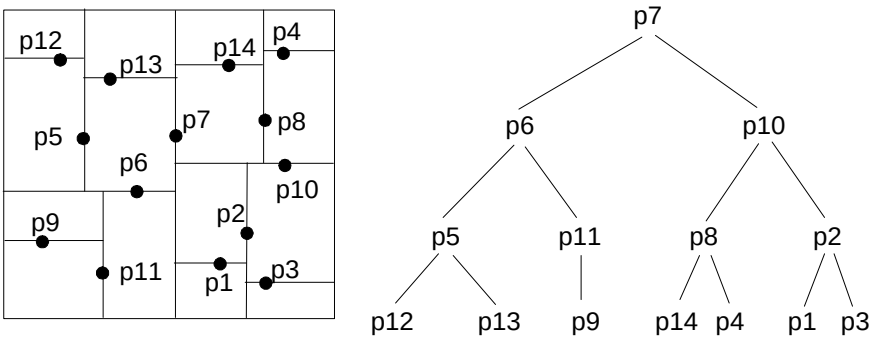


Figura 2.3: Exemplo de uma árvore k-d.

2.2.1.2 Árvore BSP

As árvores BSP (FUCHS; KEDEM; NAYLOR, 1980; FUCHS; ABRAM; GRANT, 1983) (figura 2.4), assim como as k-d, são árvores binárias que dividem o espaço em subespaços. Entretanto, a BSP permite que a linha divisória do espaço não seja ortogonal a qualquer eixo. Os subespaços são divididos recursivamente até que se tenha uma quantidade máxima de objetos estipulada em cada um deles.

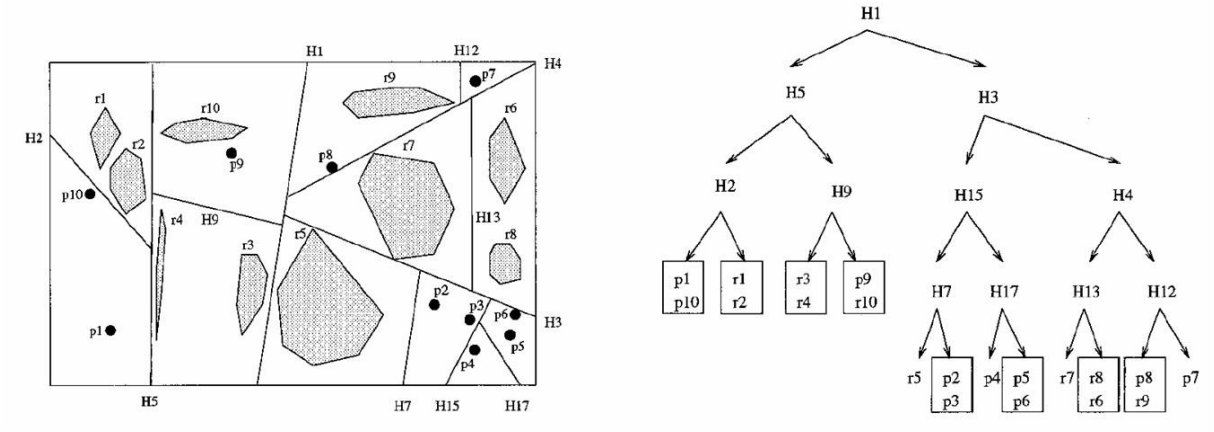


Figura 2.4: Exemplo de uma árvore BSP.

As árvores BSP adaptam-se bem a diferentes distribuições de dados, mas, muitas vezes, elas são desbalanceadas apresentando subárvores bastante profundas, o que pode acarretar em uma performance ruim (GAEDE; GÜNTHER, 1998). Além disso, elas requerem maior espaço, pois sua estrutura (de divisões não ortogonais aos eixos) é mais complexa em relação a k-d, ocupando mais espaço.

2.2.1.3 Quadtree

A quadtree (FINKEL; BENTLEY, 1974) (figura 2.5) é bastante semelhante a k-d. O espaço é subdividido em 2^d subespaços onde d é o número de dimensões. As linhas divisórias devem ser ortogonais aos eixos. Portanto, cada nodo tem 2^d nodos filhos ao contrário da árvore k-d que é binária.

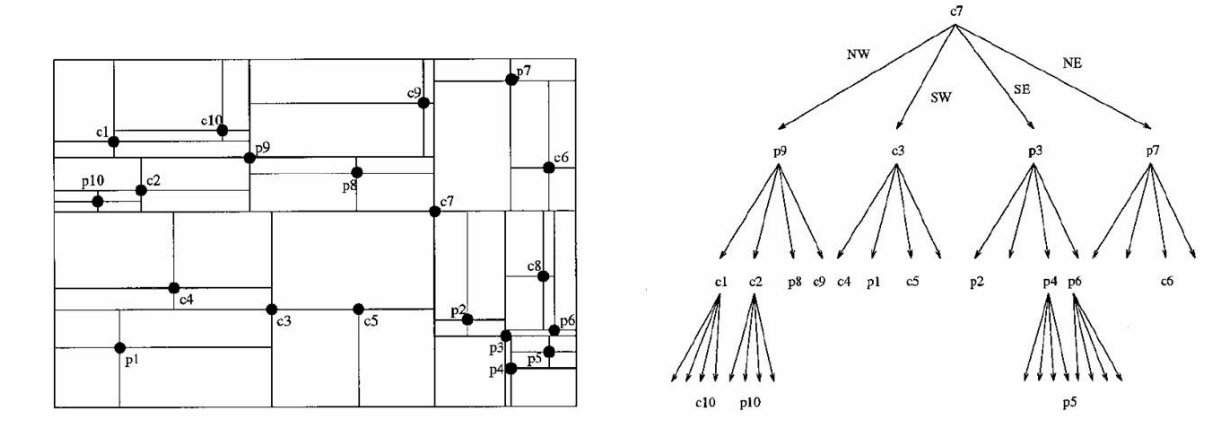


Figura 2.5: Exemplo de uma quadtree.

Existe uma variação da quadtree (*region quadtree* (SAMET, 1984)) (figura 2.6) que impõe que os 2^d subespaços devem ter o mesmo tamanho, o que facilita bastante a consulta na árvore.

Existe ainda outra variação que permite indexar polígonos, chamada de PM quadtree (SAMET; WEBBER, 1985).

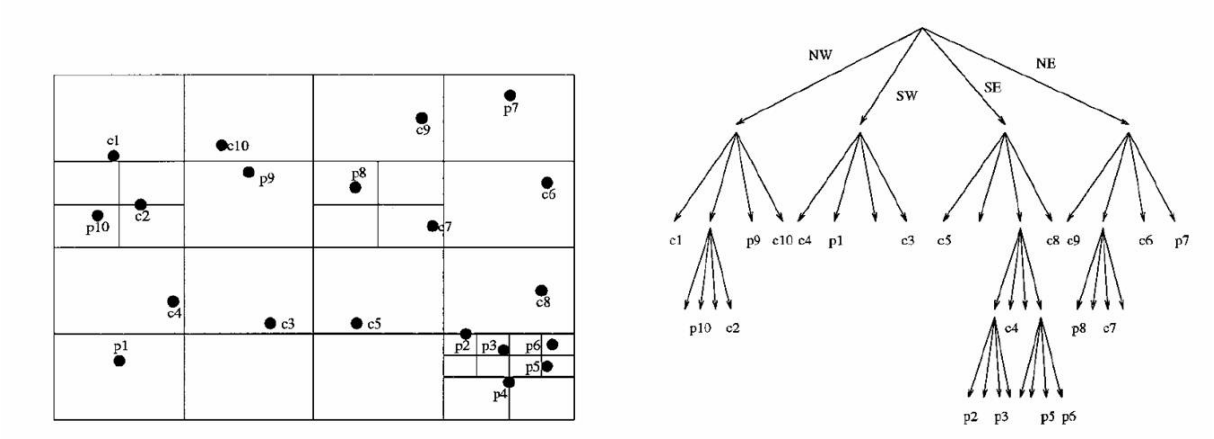


Figura 2.6: Exemplo de uma *region quadtree*.

2.2.2 Métodos de Acesso Espacial

2.2.2.1 Árvore R

A árvore R (GUTTMAN, 1984) (figura 2.7) divide o espaço em retângulos n-dimensionais que vão envolver outros desses retângulos (criando uma hierarquia de retângulos) ou objetos do espaço propriamente (polígonos, pontos, etc). Esses retângulos devem ser os menores possíveis que envolvam o objeto desejado. Além disso, a intersecção deles pode não ser vazia, mas um objeto só está envolvido por um retângulo se este o cobrir completamente, ou seja, se um retângulo estiver cobrindo apenas parte de um objeto, esse objeto não pertence ao retângulo. As arestas dos retângulos são ortogonais aos eixos.

A raiz da árvores é composta pelo primeiro nível de retângulos que cobrem o espaço. Cada retângulo do nodo aponta para um nodo que contém todos retângulos envolvidos por esse. Esse processo se repete até se chegar as folhas que contêm os objetos do espaço.

Todas folhas da árvore R ficam no mesmo nível, e elas contêm apenas objetos do espaço, não retângulos. Nodos não-folha contêm apenas retângulos, não objetos do espaço. Cada nodo da árvore, com exceção da raiz, deve conter uma quantidade mínima e máxima de retângulos definidas previamente.

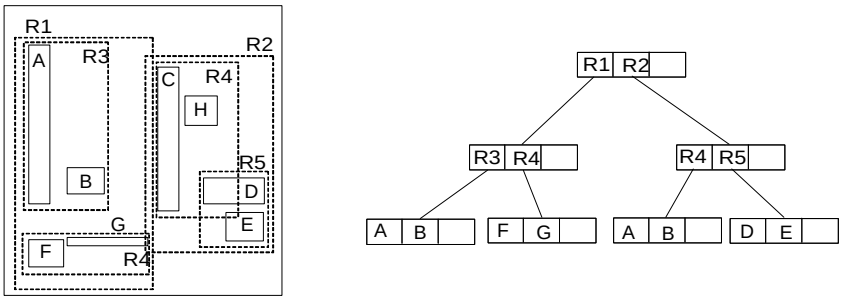


Figura 2.7: Exemplo de uma árvore R.

2.2.2.2 Árvore R+

Como os retângulos na árvore R podem estar interseccionados, quando um determinado objeto é procurado, pode-se ter que verificar mais de um retângulo até que o objeto seja encontrado. Para evitar esse problema, a árvore R+ (SELLIS; ROUSSOPOULOS; FALOUTSOS, 1987) proíbe a intersecção de retângulos e não impõe que um objeto deve estar contigo por inteiro em um retângulo para que pertença a ele. Portanto, um mesmo objeto pode estar contido em mais de um retângulo, facilitando a consulta.

No entanto, para garantir que os retângulos não se interseccionem, a árvore R+ força algumas divisões de retângulos que seriam desnecessárias, o que causa um maior número de nodos e, conseqüentemente, maior espaço ocupado do índice devido a espaços não utilizados nos nodos.

*2.2.2.3 Árvore R**

A árvore R* (BECKMANN et al., 1990) tenta melhorar o problema de performance da árvore R devido a intersecção de retângulos, e da R+ de espaço não utilizado. Assim como a árvore R, a R* permite intersecção de retângulos, mas aplica uma política de inserção de novos objetos que minimiza a quantidade intersecção. Quando um nodo é sobrecarregado, ao invés dele ser dividido imediatamente como ocorre na árvore R, são removidas algumas entradas desse nodo e reinseridas na árvore. Esse algoritmo baseia-se em (PREPARATA; SHAMOS, 1985).

2.2.2.4 Árvore X

A árvore X (BERCHTOLD; KEIM; KRIEGEL, 1996) é mais uma variação da árvore R. Ela é apropriada para indexar espaços de várias dimensões, o que pode ser feita em árvores R, mas elas foram projetadas tendo em mente um espaço de duas dimensões. A árvore X introduz o conceito de *supernodo*. Quando um nodo está cheio, sua divisão é adiada aumentando-se seu tamanho, transformando-o em um supernodo. Além disso, é mantido um histórico das divisões em uma árvore binária para melhorar futuras divisões.

2.2.3 Métodos de Acesso Métrico

2.2.3.1 Árvore Burkhard-Keller

As árvores Burkhard-Keller (BURKHARD; KELLER, 1973) (figura 2.8) foram as primeiras árvores métricas (BÖHM; BERCHTOLD; KEIM, 2001). Para a raiz da árvore, é escolhido aleatoriamente algum objeto pivô. Então, é usada uma função de distância que retorna um valor discreto para particionar os demais objetos do universo. Essas partições formariam os primeiros ramos da árvore. Esse processo é repetido para todas partições não vazias, construindo-se assim, a árvore.

Existem variações dessa árvore. Por exemplo, pode-se fazer a restrição de que todos pivôs de um mesmo nível na árvore sejam o mesmo objeto (BAEZA-YATES et al., 1994).

2.2.3.2 Árvore VP

A árvore VP (vantage-point) (UHLMANN, 1991) (figura 2.9) é uma árvore binária que, semelhante à árvore Burkhard-Keller, seleciona um elemento pivô que

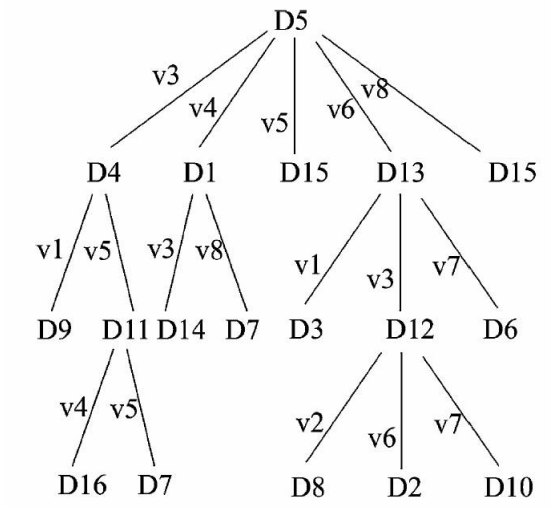


Figura 2.8: Exemplo de uma árvore Burkhard-Keller. D: dados; v: valor da distância com relação ao pivô.

será a raiz da árvore e particiona os demais baseados na sua distância em relação ao pivô, dividindo os elementos em dois subconjuntos. Esse processo é repetido para os subconjuntos da mesma forma que na árvore Burkhard-Keller. Mas, ao contrário da árvore Burkhard-Keller, a função de distância retorna um valor contínuo, e não um valor discreto. Essa regra também se aplica aos demais MAMs apresentadas nessa seção.

Existem variações dessa árvore como a árvore VP otimizada (CHIUEH, 1994), árvore VP múltipla (BOZKAYA; OZSOYOGLU, 1997) e a floresta VP (YIANILOS, 1999).

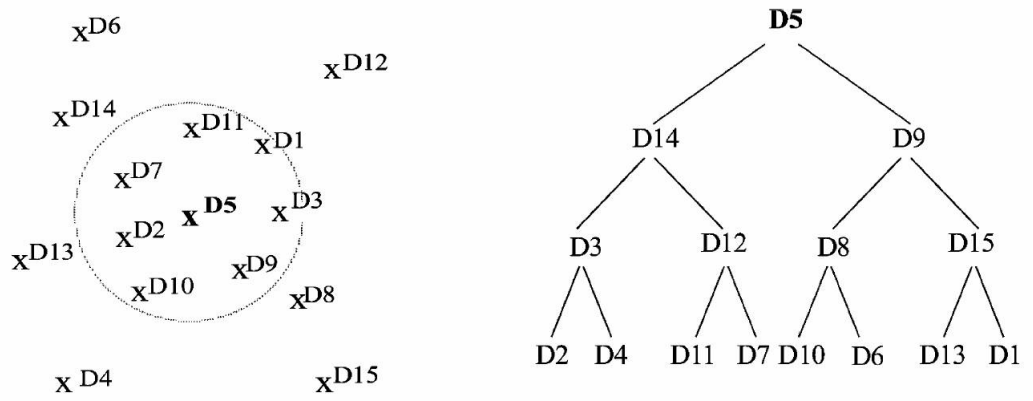


Figura 2.9: Exemplo de uma árvore VP.

2.2.3.3 Árvore GH

A árvore GH (generalized hyperplane) (UHLMANN, 1991) (figura 2.10) é uma árvore binária que usa dois pivôs em cada nível da árvore. Os elementos mais próximos ao pivô da esquerda são alocados para ele, da mesma forma os elementos mais similares ao pivô da direita são alocados a ele.

A árvore de acesso geométrico ao vizinho próximo (geometric near-neighbor access tree - GNAT) (BRIN, 1995) é uma variação da árvore GH. No entanto a GNAT

não é binária, é n-ária, ou seja, são usados n pivôs em cada nível da árvore.

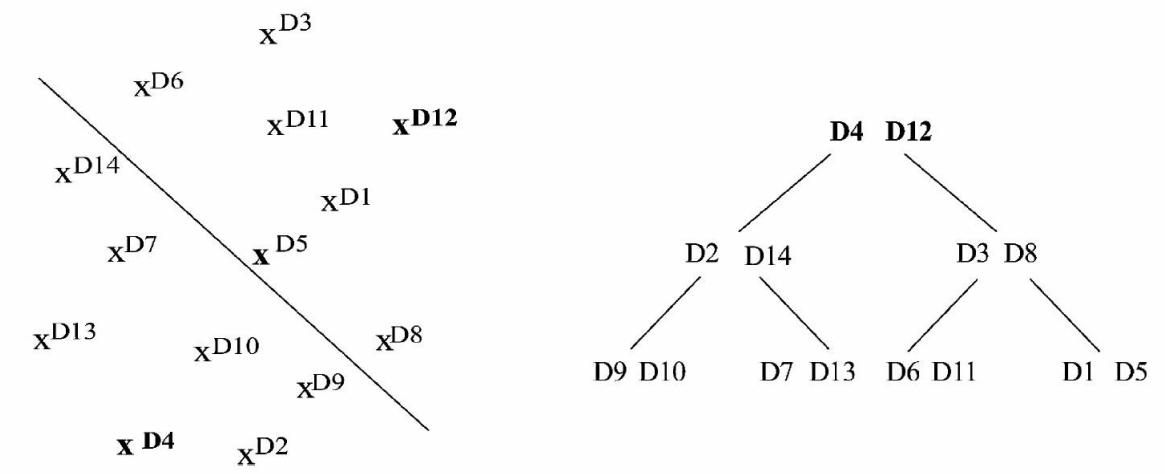


Figura 2.10: Exemplo de uma árvore gh.

2.2.4 Árvore M

A árvore M (CIACCIA; PATELLA; ZEZULA, 1997) é um MAM que introduziu importantes características se destacando de seus antecessores:

Uso de memória secundária Essa árvore usa nodos de tamanho fixo permitindo que eles sejam armazenados como páginas de disco, facilitando assim o uso da memória secundária.

Dinamismo É possível inserir e remover objetos da árvore com facilidade, sendo que esse processo não implica em uma reorganização drástica e custosa da estrutura.

Escalabilidade A árvore M é construída de baixo para cima (*bottom-up*) de forma que ela sempre está balanceada. Por essa razão, o custo da busca não cresce explosivamente a medida que a quantidade de objeto aumenta.

A estrutura da árvore M é bastante parecida com a da árvore R, apesar dessa última ser um MAE e não um MAM. Os nodos folha da árvore contêm ponteiros para os objetos que estão sendo indexados (poderiam conter os próprios objetos ao invés dos ponteiros). Cada entrada do nodo não-folha contêm um ponteiro para uma subárvore, o representativo dessa subárvore, o raio de cobertura desse representativo e a distância métrica do representativo para seu pai. Representativos (*representatives*), também chamados de objetos de rota (*routing objects*), representam um grupo de objetos. Eles podem ser qualquer objeto do espaço que foi escolhido como centróide entre os objetos de um determinado nodo. Esse grupo de objetos está contigo na subárvore para o qual esse representativo aponta. Além disso, a distância métrica (Seção 2.1) desse representativo para qualquer objeto contido em sua subárvore não deve ser maior do que o raio de cobertura do representativo. Portanto, sempre que um objeto é inserido, o raio do representativo pode ter que ser aumentado para cobrir esse novo objeto.

Cada nodo da árvore M é armazenado em uma página de disco. Portanto, cada nodo que é acessado representa um acesso a disco. Além disso, quando o usuário

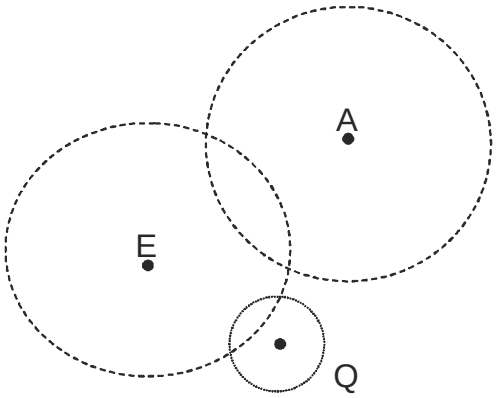


Figura 2.11: Descarte de nodos na árvore M

faz requisição de uma busca, ele fornece o *objeto consulta* (veja Seção 2.1) que será comparado (usando a função de distância métrica) com os objetos da árvore. Essa comparação pode ser extremamente custosa se, por exemplo, o que se está comparando forem vídeos. Portanto, é muito importante evitar tanto os acessos a nodos como os cálculos de distância. A estrutura da árvore M usa a propriedade da desigualdade triangular da função de distância (Seção 2.1) para isso.

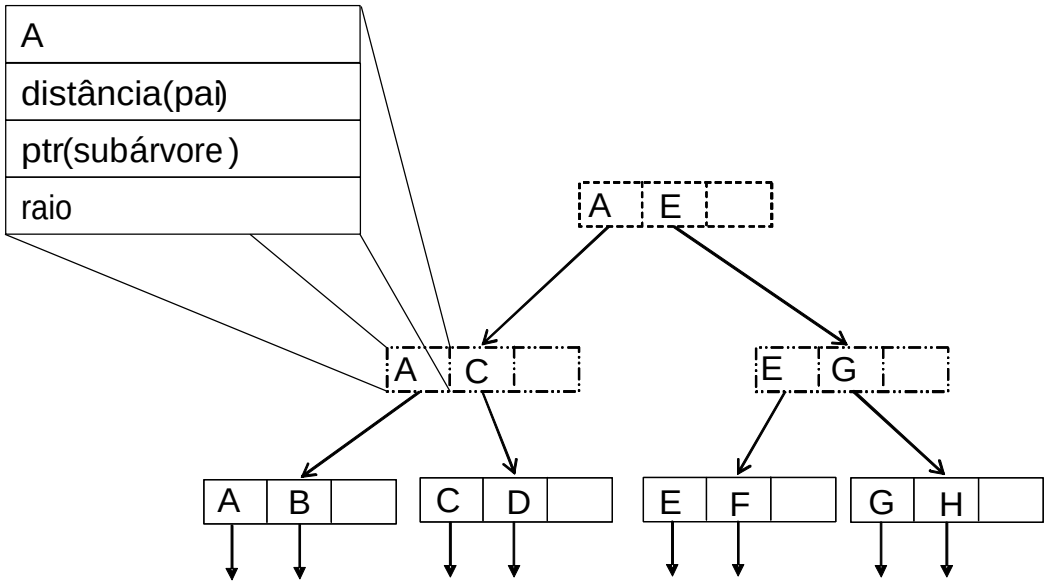
Supondo a função de distância d, uma consulta por abrangência $RQ(Q, R_Q)$, um representativo O e seu raio de cobertura R_O. É possível se descartar a subárvore de O se $d(Q, O) > R_Q + R_O$ porque qualquer objeto que esteja na subárvore de O está necessariamente dentro do raio de cobertura de O, e portanto não satisfaz o critério de busca. No exemplo da figura 2.11, Q é o objeto consulta, A e E são representativos de uma árvore M. A subárvore de A pode ser descartada, pois $d(Q, A) > R_Q + R_A$; enquanto que a subárvore de E deve ser percorrida, pois $d(Q, E) \leq R_Q + R_E$.

A figura 2.12(a) apresenta uma possível estrutura de uma árvore M e os objetos do espaço métrico (figura 2.12(b)) que estão sendo indexados (veja a Seção 2.1 para a definição de espaço métrico). No nodo 'AE' os objetos A e E são representativos dos nodos 'AC' e 'EG' respectivamente. O nodo 'AC' contém os representativos A e C. Esse último é representativo do nodo 'CD' que é um nodo folha e portanto não tem representativos. Os objetos C e D desse nodo apontam diretamente para entradas na base de dados.

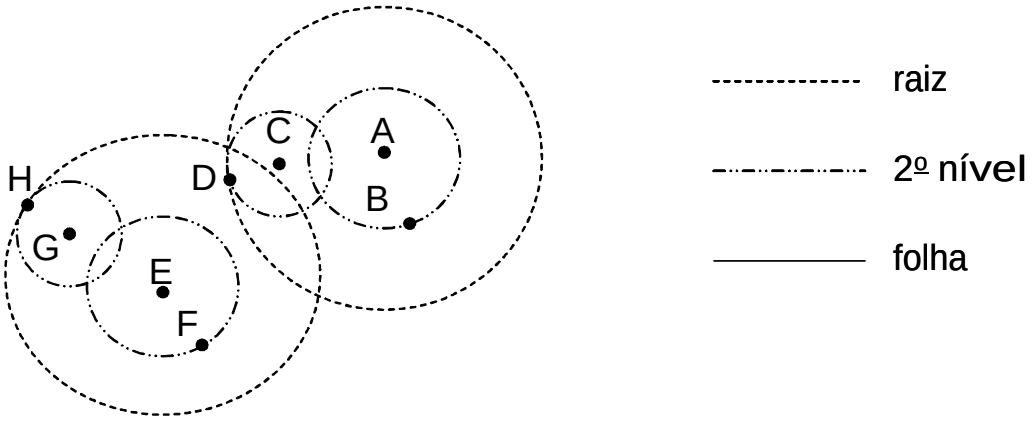
Não é permitida a inserção de um mesmo objeto em dois nodos diferentes da árvore. No entanto, um mesmo objeto pode aparecer como representativo em diversos nodos da árvore. Em cada nível, seu raio será diferente, diminuindo a medida que vai ficando mais próximo das folhas.

Novos objetos sempre são inseridos nos nodos folhas. Quando o nodo folha atinge sua capacidade máxima, é criado um novo nodo, e os objetos são divididos entre os dois. Em seguida, é escolhido um representativo para os dois nodos e esses representativos são inseridos no nodo acima. O nodo acima pode já estar em sua capacidade máxima e nesse caso ele também se divide, sendo esse um processo recursivo. Quando o nodo raiz precisa se dividir, não existe um nó acima para se inserir os dois novos representativos, portanto é criado um novo nodo que será a nova raiz da árvore. É dessa forma que a altura da árvore aumenta.

Quando um nodo é dividido, deve-se particionar os objetos dele em dois grupos que serão distribuídos entre os dois nodos. Existem diversos algoritmos para tal



(a) Estrutura de uma árvore M



(b) Objetos em um espaço métrico

Figura 2.12: Uma árvore M

fim que são detalhados em (PATELLA, 1999). O mimMax particiona os objetos de uma forma ideal, mantendo dois menores grupos possíveis. Esse algoritmo é extremamente lento, sendo de ordem n^3. A árvore Slim (TRAINA JÚNIOR et al., 2000) propõe um algoritmo quase tão eficaz (MST), mas com um custo mais baixo (ordem $n^2 \log(n)$).

O algoritmo de inserção de novos objetos da árvore M escolhe um representativo que não precisará aumentar seu raio para poder englobar o novo objeto. Se isso não for possível, o representativo que tiver o menor aumento de raio será escolhido. Essa tática visa manter os raios dos representativos o menor possível, para evitar intersecções de raios. Quanto maior for o número de intersecções, pior será a busca, já que será necessário acessar mais de um nodo quando o objeto consulta estiver exatamente na região de intersecção. A princípio essa parece uma boa tática, mas como será visto adiante (Seção 4.3) ela pode prejudicar a estrutura da árvore. O trabalho apresentado aqui foca nesse ponto. Melhorando a escolha de qual representativo será escolhido para a inserção, pode se ter aumentos significativos de desempenho na busca.

2.2.5 Outras árvores baseadas na árvore M

Após o surgimento da árvore M, surgiram árvores também métricas baseadas nela. A principal dessas novas árvores é a Slim (TRAINA JÚNIOR et al., 2000).

A árvore Slim tem sua estrutura básica semelhante à arvore M, armazenando os dados nas folhas e usando uma hierarquia de clusters nos nodo intermediários. Assim como a árvore M, a árvore Slim permite sobreposição das regiões. No entanto, ela tem sua arquitetura voltada para a redução dessa sobreposição. Isso é importante, pois a sobreposição das regiões torna a consulta no índice mais lenta (BERCHTOLD; KEIM; KRIEGEL, 1996). A árvore Slim, da mesma forma que a árvore M, preocupa-se com o acesso à memória secundária. No entanto, a árvore Slim apresenta uma performance superior aos outros MAMs, diferenciando-se pelos seguintes fatores:

- É usado um novo algoritmo de divisão baseado na *Minimal Spanning Tree* (MST) que é mais rápido do que outros algoritmos de divisão não prejudicando a performance da consulta no MAM

- É usado um novo algoritmo que guia a inserção de objetos em nodos internos nas subárvores adequadas. Esse algoritmo leva a uma maior ocupação de espaço em disco.

- É usado o algoritmo *slim-down* que torna a árvore métrica mais estreita e mais rápida em um pós-processamento. Esse algoritmo foi usado pois os autores concluíram que a sobreposição de regiões em árvores métricas acarreta uma grande ineficiência.

Além da árvore Slim, várias outras árvores foram criadas baseadas na estrutura inicial da árvore M ou da árvore Slim:

Árvore M^2 (CIACCIA; PATELLA, 2000) Ela usa um espaço métrico multidimensional, permitindo consultas por mais de uma característica ao mesmo tempo.

Árvore M+ (ZHOU et al., 2003) Combina as idéias da árvore M e MVP (BOZKAYA; OZSOYOGLU, 1997) criando uma árvore métrica com mais de um representativo por nó, aumentando assim o poder de poda.

Árvore DF (TRAINA JÚNIOR et al., 2002) Essa árvore aplica a técnica Omni (SANTOS FILHO et al., 2001) à árvore Slim, criando uma árvore Slim com representativos globais, aumentando também o poder de poda.

Árvore DBM (VIEIRA, 2004) Essa árvore usa mais nós em regiões do espaço mais densas, do que em regiões mais esparsas, o que diminui a sobreposição de nós e conseqüentemente aumenta a velocidade da consulta. Para isso, ela não faz uso da restrição do balanceamento da árvore, e mostra que não perde escalabilidade com isso.

2.2.6 Conclusão

À medida que a complexidade e volume dos dados que são armazenados aumenta, a necessidade por sistemas de consulta mais sofisticados também cresce. Diversos sistemas de busca vêm sendo propostos e melhorados cada vez mais. Os MAMs permitem eficientemente que se faça consultas sobre dados bastante complexos de serem manipulados, e que estão presentes cada vez mais nas bases de dados. A importância desse tipo de índice é, portanto, crescente também.

Esses sofisticados Métodos de Acesso, os MAMs, além de servir como indexadores para dados extremamente complexos como vídeo, podem ser muito bem aproveitados para dados bastante mais simples como os campos textuais de bases XML usadas nos experimentos desse trabalho. Apesar de dados textuas serem bem mais simples do que vídeos ou imagens por exemplo, eles podem apresentar o mesmo problema de representação em um espaço vetorial. Ou seja, a mesma dificuldade que pode se ter em extrair vetores de características de algum vídeo para representá-lo em um espaço vetorial, pode se ter com textos ou outros tipos de dados. Para esses casos, os MAMs se mostram eficazes na indexação.

Baseado na árvore R, a árvore M foi um marco no desenvolvimento de MAMs, servindo como ponto de partida para vários outros MAMs, inclusive para o MAM proposto aqui. Por isso, a árvore M serviu como referencial nos experimentos apresentados adiante.

3 APLICANDO ÁRVORE M A BASES TEXTUAIS

Esse capítulo apresenta experimentos feitos sobre bases XML usando-se o MAM árvore M. Os experimentos com a árvore M aqui apresentados preocupam-se apenas com a indexação e recuperação dos dados. O problema da qualidade da busca, ou seja, se a consulta representa realmente o que o usuário quer não é tratado aqui. O problema da consulta por similaridade em bases XML é tratado por (DORNELES; LIMA; HEUSER, 2003). Como qualquer consulta por similaridade, o usuário deve fornecer um objeto consulta (veja Seção 2.1). Nesse trabalho são propostas diversas métricas para que cada campo de uma fonte XML seja comparado com o campo correspondente no objeto consulta.

3.1 Fontes

Para os experimentos, fontes de dados reais foram usadas: referências bibliográficas de arquivos BibTeX. As fontes foram extraídas de "The Collection of Computer Science Bibliographies" (`http://dblp.uni-trier.de/xml/`) e de arquivos BibTeX de membros do Grupo de Banco de Dados da UFRGS (`http://www.inf.ufrgs.br/gbd/`) totalizando mais de 16 mil entradas bibliográficas. Como essas bases estavam no formato BibTeX, elas foram convertidas para XML usando bib2XML (`http://www.cs.duke.edu/~sprenkle/bibtex2html/`). Essas fontes XML estão em `http://www.inf.ufrgs.br/~nadvorny/bibs_database.zip`.

3.2 Funções de distância

Para cada tipo de campo XML, deve-se usar uma função de distância adequada. Os campos indexados nesse trabalho contêm pequenas seqüências de caracteres. Para medir a distância entre eles, usou-se a função de Levenshtein (LEVENSHTEIN, 1965) (edit distance) e outra baseada em grams (CAVNAR, 1993).

A função de Levenshtein compara duas seqüências de caracteres contando o número de caracteres que devem ser substituídos, removidos, ou incluídos para transformar uma palavra na outra. Por exemplo, a distância de Levenshtein entre "carro" e "caro" é 1, pois basta incluir (ou remover) um caractere para transformar uma na outra. A distância entre "raptar" e "raspar" é 2. Essa função é normalizada dividindo-se a disância pelo tamanho da maior palavra, de forma que o resultado é um número entre 0 e 1.

A função baseada em grams utilizada conta quantos grams (pedaços de palavra)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<XML>
   <REFERENCES>
      <REF>
         <TYPE>INBOOK</TYPE>
         <AUTHORS>
            <AUTHOR>
               <LAST>Lawrence</LAST>
               <PREF>Steve</PREF>
            </AUTHOR>
        </AUTHORS>
         <DATE>
            <YEAR>2000</YEAR>
         </DATE>
         <TITLE>Overfitting and Neural...</TITLE>
         <PAGES>
            <START>114</START>
            <END>119</END>
         </PAGES>
         <PUBLISHER>IEEE Press</PUBLISHER>
         <BOOKTITLE>Proceedings of the IEEE...</BOOKTITLE>
         <REFNUM>lawrence00overfitting</REFNUM>
      </REF>
   </REFERENCES>
</XML>
```

Figura 3.1: Exemplo de um documento XML

existem em comum entre duas palavras. O tamanho do gram pode ser variado. Nesse trabalho foram utilizados grams de tamanho 2 (ou seja, 2 caracteres), pois esse tamanho se mostrou mais adequado para esse domínio de objetos. Para se ter um valor normalizado, dividi-se a quantidade de grams iguais pela quantidade total de grams, de forma que o resultado é um número entre 0 e 1.

3.3 Experimentos

Nos experimentos foram usadas bases com um subconjunto do total de fontes que se tem. Foram usadas bases de 200, 500 e 1000 objetos. Cada documento XML tem campos sobre a publicação (nome do autor, título, ano, etc.). Um exemplo de fonte é mostrado na Figura 3.1.

Os documentos XML foram indexados pelo campo de título da publicação, representado pelo campo `<TITLE>` e sobrenome do primeiro autor, representado pelo campo `<LAST>`. Foram gerados, portanto, dois índices.

Para se testar o desempenho da consulta nesses índices, foram executadas 14 consultas por abrangência usando-se os raios de consulta $0; 0,1; 0,2...1,0$. Portanto, $14 \times 11 = 154$ consultas para cada um dos índices gerados. Os gráficos 3.2, 3.3 e 3.4 apresentam a média dessas 14 consultas para cada um dos raios (distâncias). Como a distância máxima entre dois objetos é 1 (já que a distância está normalizada), uma

consulta por abrangência de raio=1 irá recuperar todos objetos da base, percorrendo todo o índice.

O custo de IO mostrado, representa o número de páginas de disco que foram acessados para se fazer a consulta. Cada nodo da árvore é armazenado em uma página de disco. Portanto, cada vez que o algoritmo varre um nodo diferente, está sendo requisitada uma nova página.

O custo de CPU representa o número de cálculos de distância que foram feitos nas consultas. Quando uma consulta por similaridade está sendo processada, é necessário comparar o objeto consulta do usuário com os objetos da base. Essa comparação é feita usando uma função de distância adequada para o tipo de objeto (nesse caso está se usando Levenshtein e grams). Esse custo de CPU é relevante em relação ao IO porque esse cálculo de distância pode ser extremamente dispendioso. Por exemplo, para calcular a distância de um trecho de filme para outro pode ser necessário analisar milhares de imagens e trechos de som. Por isso, mesmo sendo cálculos de CPU, eles podem se tornar relevantes frente ao custo de IO.

3.4 Análise do índice sobre campo do nome do autor

A figura 3.2 compara o desempenho da consulta no índice pelo campo <LAST> usando-se os tamanhos de páginas 512Bytes, 1024Bytes, 4096Bytes.

Em termos de acesso a disco, claramente o aumento do tamanho da página tem influência direta para a redução de páginas requisitadas. Aumentando-se o tamanho da página, pode-se armazenar mais elementos por página e consequentemente o índice de forma geral usa menos páginas, o que faz com que a consulta também precise requisitar menos páginas.

Em termos do uso de CPU, pode-se observar que o tamanho da página também exerce um certa influência, embora não tão forte. Como os cálculos de distância são feitos para cada elemento dentro do nodo, não importa tanto quantos nodos existem, o que mais influencia o desempenho é o número de elementos total para o qual serão feitos os cálculos de distância. Se esses elementos estão todos num nodo, ou se estão distribuídos em 200, a quantidade de cálculo será a mesma, se os 200 nodos forem percorridos. Claro que, ao evitar o acesso a um nodo, está se evitando vários cálculos de distância. Portanto, quanto menor o número de nodos acessados, provavelmente, menor será também o número de cálculo de distância. Mas isso depende também da ocupação de cada nodo. Uma consulta que percorre 2 nodos com 100 elementos cada um, terá um custo de CPU bem maior do que um consulta que acesse 20 nodos com 2 elementos em cada um.

Por isso, nos gráficos apresentados, a curva do custo de CPU é influenciada pela curva de IO, mas não é completamente fiel à ela.

3.5 Análise do índice sobre campo de título da publicação

A figura 3.3 compara o desempenho da consulta no índice pelo campo <TITLE> usando-se os tamanhos de páginas 2048Bytes, 4096Bytes, 8192Bytes.

Pode-se observar que a curva do custo de CPU é mais fiel à curva de IO do que na comparação anterior, embora ainda não seja completamente fiel.

É interessante de se observar que, o índice com tamanho de página de 8kB teve um desempenho superior aos demais. No entanto, o índice com tamanho de página

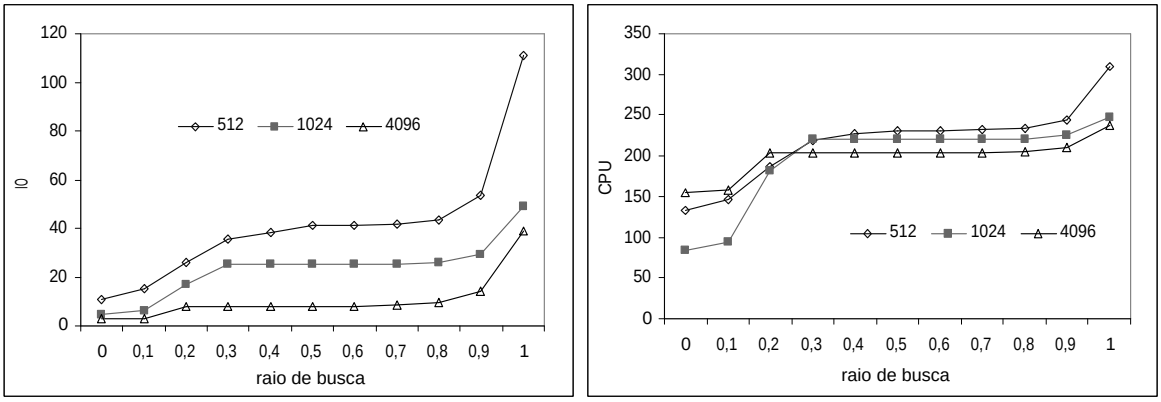


Figura 3.2: Comparação de tamanho de bloco para o campo <LAST>.

de 4kB, não teve um desempenho constantemente superior ao índice de 2kB. A quantidade de requisições de páginas é muito influenciada pelo número de nodos da árvore, que é muito influenciado pelo tamanho do nodo (quanto maior o tamanho, menor será a quantidade de nodos necessária). Mas a quantidade de requisições de páginas também é muito influenciada pela organização da árvore. Quanto melhor os elementos dela estiverem agrupados, melhor será o desempenho. Provavelmente, o índice de 2kB ficou melhor agrupado do que o índice de 4kB, conseguindo um desempenho superior muitas vezes, mesmo tendo um número total de nodos maior.

A figura 3.4 mostra os mesmos gráficos apresentados pela figura 3.3, mas com enfoque na comparação do desempenho do índice para diferentes quantidades de elementos. Pode-se observar que, tanto o custo de CPU quanto o custo de IO, aumentam mais ou menos linearmente em relação ao número de elemento. Isto é, dobrando-se o número de elementos indexados, dobra o custo de CPU e IO pra se fazer uma consulta.

3.6 Agrupamentos gerados pela árvore M

Observando-se os índices gerados, constatou-se que os elementos nele não estavam muito bem agrupados, acontecendo muitos casos em que a ocupação do nodo era muito baixa (muitas vezes com apenas um elemento). Também se observou que era comum existirem representativos com raio=1, o que significa que o nodo desse representativo sempre será acessado, porque raio 1 significa que qualquer elemento pode estar na subárvore daquele representativo (a distância máxima entre quaisquer dois elementos, com a métrica utilizada é 1). Um exemplo desse tipo de agrupamento não adequado pode ser observado na figura 3.5. Cada um dos 4 círculos representa um grupo ou, mais especificamente, o raio do representativo da subárvore. Note que um grupo está bastante grande englobando vários elementos, enquanto que outros grupos estão pequenos e englobando poucos elementos, havendo desperdício de espaço nesses nodos.

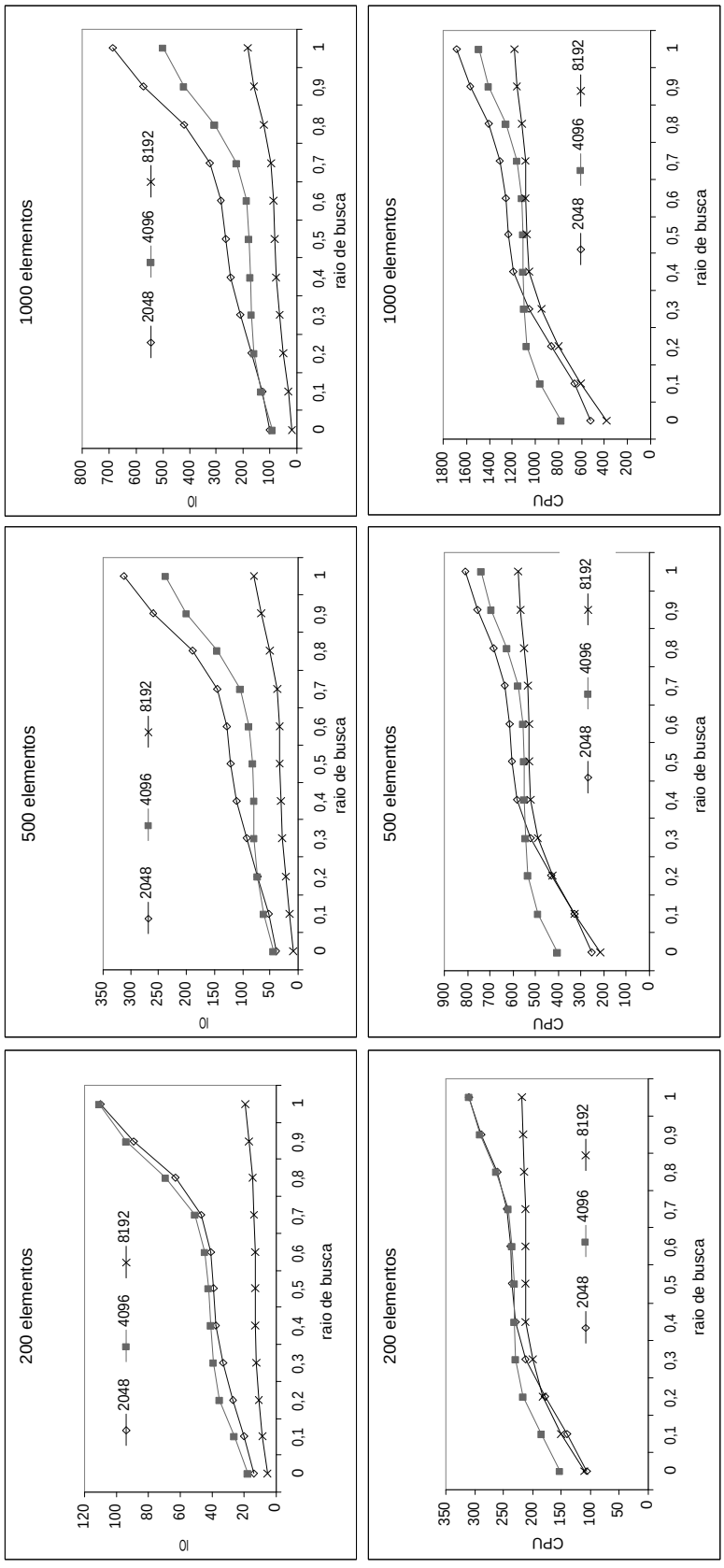


Figura 3.3: Comparação de tamanho de bloco para o campo <TITLE>.

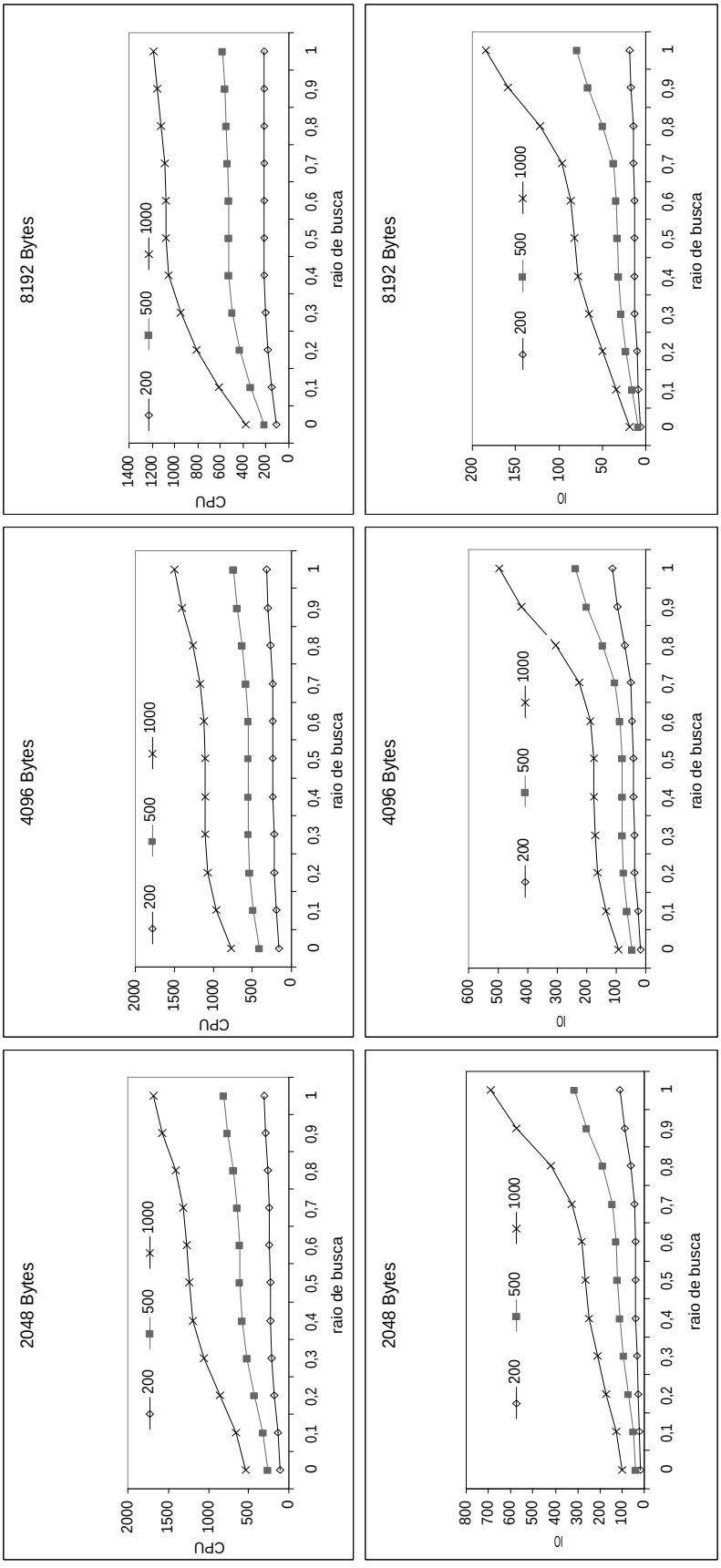


Figura 3.4: Comparação do número de objetos indexados para o campo <TITLE>.

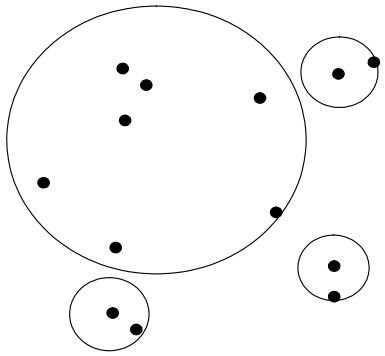


Figura 3.5: Agrupamento não adequado de objetos

4 ÁRVORE TM

4.1 Introdução

Como foi visto no capítulo anterior, dependendo da distribuição espacial, a árvore M pode criar grupos muito maiores do que outros, o que pode gerar um atraso em todas as consultas. Se a distribuição dos objetos fosse diferente, o agrupamento deles poderia ser melhor (mais homogêneo). Pode-se então tentar redistribuir os objetos no espaço para ajudar o algoritmo de agrupamento da árvore M a achar grupos melhores, mais homogêneos, de forma que essa estrutura de acesso tenha um desempenho melhor. A figura 4.4 mostra o que seria um agrupamento homogêneo ao contrário do agrupamento bem desproporcional mostrado no capítulo anterior na figura 3.5.

Esse novo espaço onde os objetos estão redistribuídos foi chamado de *espaço distorcido*. Para se obter um espaço distorcido, é necessário aplicar uma *função de distorção* sobre o espaço métrico. Esse novo espaço salienta a distância entre os objetos, de forma que o algoritmo de agrupamento de árvore M é induzido a criar grupos mais homogêneos automaticamente.

Nesse espaço distorcido, no entanto, não se pode fazer consultas ao objeto, como será visto em detalhes nesse capítulo. Isso porque o espaço distorcido não garante que os objetos contidos nele serão encontrados, já que a função de distorção altera a distância entre os objetos. No espaço métrico (veja Seção 2.1), esse problema não acontece. Mas o espaço métrico pode levar a agrupamentos não homogêneos, o que pode significar um baixo desempenho, como visto no capítulo anterior.

A solução para esse problema foi a chamada árvore TM (*Twisted and Metric tree*). Essa nova árvore apresentada neste capítulo usa o espaço métrico para fazer a consulta a objetos e usa o espaço distorcido para fazer a indexação dos objetos. Dessa forma, soluciona-se o problema de agrupamentos não homogêneos com a introdução do espaço distorcido, e evita-se o problema do próprio espaço distorcido (falha na consulta) usando complementarmente o espaço métrico.

4.2 Função de distorção e espaço distorcido

Um espaço métrico é definido pelo par universo $\mathbb{U}$ e função de distância d. A função de distância atribui um valor a quaisquer dois objetos pertencentes ao universo. Portanto, essa função está definindo como os objetos estão distribuídos no espaço métrico. Alterando essa função de distância, é possível obter uma distribuição espacial que induza o algoritmo de agrupamento da árvore M a criar grupos mais homogêneos, agilizando a consulta.

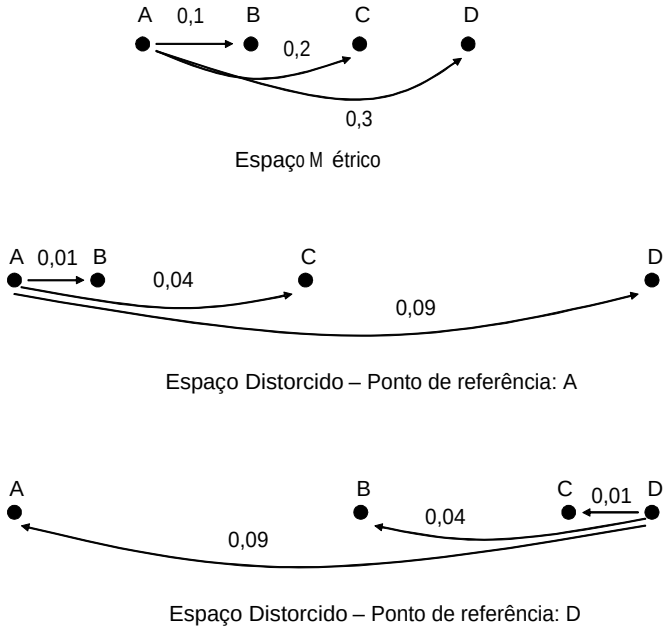


Figura 4.1: Relatividade das distâncias no espaço distorcido.

Baseando-se nessa idéia, se define o *espaço distorcido*. Um espaço distorcido $\mathbb{T}$ é definido pela tripla $< \mathbb{U}, d, t >$ onde $\mathbb{U}$ é o conjunto de todos objetos possíveis, d é uma função de distância e t é uma *função de distorção*.

Por exemplo, se fosse aplicada a função $y = x^2$ sobre a função de distância entre dois objetos, ter-se-ia uma distribuição espacial completamente diferente do espaço métrico original, como mostra a figura 4.1. Essa função, aproxima todos objetos do espaço (x está entre 0 e 1). No entanto, objetos mais próximos vão se aproximar mais do que objetos que estão mais longe como mostra a figura 4.1. É interessante também observar que as distâncias nesse espaço distorcido, são relativas a cada objeto. Nesse mesmo exemplo da figura 4.1, se o ponto de referência é o objeto A, o objeto C está a uma distância de 0,05 $(0,09 - 0,04)$ do objeto D. No entanto, se o ponto de referência for o objeto D, a distância entre o próprio D e C é $0,01$, já que a distância entre eles no espaço métrico é $0,1$ $(0,1^2 = 0,01)$. Portanto, cada objeto desse espaço distorcido, "vê" todos outros elementos aproximarem-se.

4.3 Crescimento homogêneo das subárvores

Quando um objeto é inserido em uma árvore M, se ele não está contido em nenhum dos agrupamentos já existentes, esse objeto é inserido na subárvore que causará o menor aumento de raio do representativo (Seção 2.2.4). Para ilustrar essa situação veja a figura 4.2. O objeto A e B são representativos de raios 10 e 1 respectivamente[1]. Deseja-se inserir o objeto C, o qual está a uma distância 12 de A e 5 de B. Portanto, se o objeto C fosse inserido na subárvore de A, o raio de A teria que aumentar de 10 para 12 para englobar C, um aumento de 2 unidades. No entanto, ao inserir C na subárvore de B, seu raio teria que ser aumentado de 1 para 5, um aumento de 4 unidades. Como $2 < 4$, o algoritmo da árvore M (e de qualquer

[1]Usualmente a distância ficaria normalizada entre 0 e 1, mas, para que o exemplo fique mais simples, se usou números inteiros.

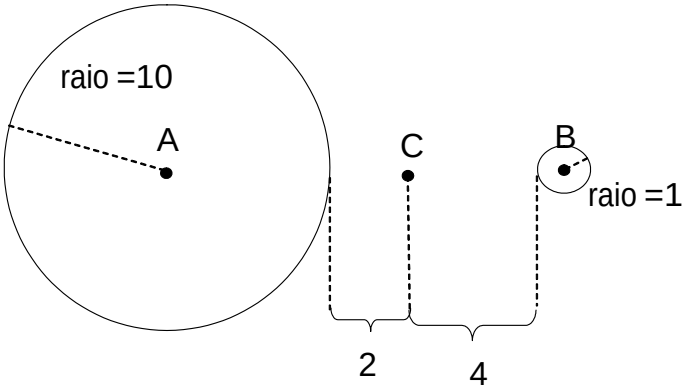


Figura 4.2: Crescimento de raio desequilibrado no espaço métrico.

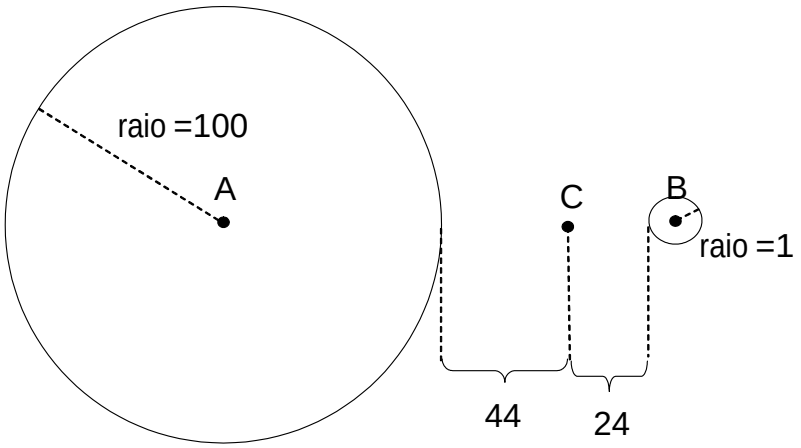


Figura 4.3: Crescimento homogêneo de raio no espaço distorcido.

outra árvore derivada de M até então) escolhe inserir C na subárvore de A.

Em princípio essa parece uma boa estratégia, sempre escolhendo o menor aumento de raio, ter-se-ia uma soma total de raios no espaço menor possível. Entretanto, essa estratégia causa um problema: alguns grupos podem crescer demasiadamente, enquanto outros grupos não aumentam e ficam com uma ocupação muito baixa. Quando se tem representativos de raios muito grande, a probabilidade desse representativo ser acessado em uma consulta começa a ficar muito grande, prejudicando o desempenho geral do Método de Acesso. Imagine por exemplo, um representativo de raio=0,9 em um espaço de tamanho 1. Esse representativo será acessado em quase todas as consultas. Foi exatamente esse o problema identificado nos experimentos com a árvore M descritos no Capítulo 3.

Usando a técnica proposta nesse trabalho, aplica-se uma função de distorção, por exemplo, $y = x^2$ à função de distância. No exemplo da figura 4.2, o raio de A que é 10 no espaço métrico fica sendo 100 no espaço distorcido. A distância de A até C passa de 12 para 144, o raio de B continua sendo 1, e a distância de B para C passa de 5 para 25. Portanto, no espaço distorcido, para que o raio de A englobe C, ele deve aumentar 44 unidades (144 - 100 = 44). Já o raio de B precisa aumentar apenas 24 unidades (25 - 1 = 24). Essa nova situação é ilustrada na figura 4.3.

Nesse novo cenário, o representativo B é escolhido para englobar C, ao contrário do que acontece quando se usa o espaço métrico. Dessa forma, a função de distorção

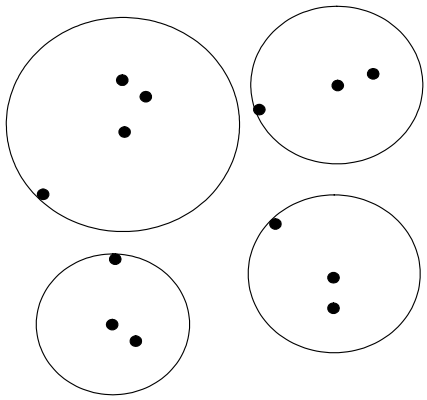


Figura 4.4: Agrupamento homogêneo dos objetos

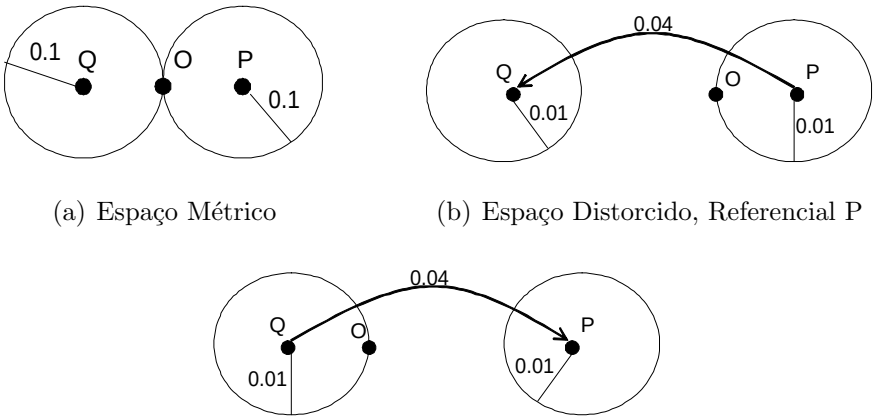
está impedindo que representativos que estão com um raio muito *grande* cresçam ainda mais, prejudicando o acesso na consulta. É claro que "grande" é muito relativo. Dependendo do espaço 10 pode ser muito grande ou muito pequeno. Quem define esse limiar é a função de distorção, e por isso ela deve ser bem escolhida. Nesse exemplo, a função $y = x^2$ "decidiu" que 10 era grande demais, e deu preferência ao aumento do raio do representativo B, mesmo esse aumento sendo maior do que seria o aumento de A. Dessa forma, evita-se agrupamentos desproporcionais como ilustrados na figura 3.5 obtendo-se agrupamentos muito mais homogêneos como mostra a figura 4.4. Esse agrupamento vantajoso foi validado pelos experimentos apresentados a seguir nesse capítulo.

Poderia ser interessante identificar funções ideiais de distorção para cada espaço métrico. Mas nesse trabalho não se definiu uma forma de se chegar a uma função de distorção ideal. De fato, não se sabe se isso é realmente possível, sendo isso inclusive um dos trabalhos futuros indicados. Nos experimentos apresentados, escolheu-se funções que intuitivamente pareciam boas, mas nenhum método formal foi usado para isso.

4.4 Consulta no espaço distorcido

O fato dos objetos estarem a uma distância que varia dependendo da referência causa um problema. Um representativo que tem um certo raio, indica que qualquer objeto em sua subárvore está dentro desse raio. No entanto, isso é verdade apenas se a referência for o próprio representativo, mas quando se está fazendo uma consulta, a referência é o objeto consulta e não o representativo. Portanto, pode acontecer de um objeto estar fora do raio do representativo, do ponto de vista do objeto consulta. A figura 4.5 mostra essa situação. Se a referência no espaço é o objeto P, o objeto O está dentro de seu raio e fora do raio de consulta de Q. Mas se a referência for o objeto Q, o objeto O está fora do raio do P, e dentro do seu raio de consulta. Na prática, isso significa que, se uma consulta estivesse sendo feita procurando-se elementos com uma distância de até 0,01 de Q, o nodo do representativo P não seria acessado, e portanto, não seria encontrado o objeto O, mesmo ele estando respeitando os critério da consulta.

O espaço métrico é definido pelo par $< \mathbb{U}, d >$, sendo que a função d, para que possa ser usada em árvores M, deve respeitar as propriedades de positividade, sime-



(a) Espaço Métrico

(b) Espaço Distorcido, Referencial P

(c) Espaço Distorcido, Referencial Q

Figura 4.5: Problema da Desigualdade Triangular

tria, reflexividade, positividade estrita e desigualdade triangular que estão detalhadas na Seção 2.1. Aplicando-se a função $y = x^2$ sobre as distâncias, a desigualdade triangular deixa de ser garantida. Por exemplo, $0,1 + 0,2 >= 0,3$. No entanto, $0,1^2 + 0,2^2 (= 0,05)$ é menor do que $0,3^2 (= 0,09)$.

Portanto, a função de distância não é mais dita métrica, de forma que o espaço também não pode mais ser considerado métrico. No entanto, a função usada nas consultas em árvores M exige a propriedade da desigualdade triangular. Se essa não é mais respeitada, a consulta não retorna resultados corretos, como visto no exemplo anterior.

4.5 Usando o espaço métrico e o distorcido: a árvore TM

Para resolver esse problema, usa-se o raio no espaço distorcido apenas para a indexação da base. Quando for feita uma consulta, usa-se o raio do espaço métrico. Dessa forma, se tem um bom agrupamento dos objetos porque o espaço está distorcendo as distâncias para favorecer isso, mas ao mesmo tempo, a consulta funciona, porque está se mantendo o raio do espaço métrico também. Ou seja, se está trabalhando nos dois espaços.

A mudança estrutural que se deve fazer na árvore M para se obter uma árvore TM é bastante simples. De fato, basta apenas incluir um campo a mais em cada entrada de nodo para armazenar, além do raio no espaço métrico, o raio no espaço distorcido. A figura 4.6 ilustra bem a simplicidade da alteração.

4.6 Restrição da função de distorção

Apesar de não ter sido definido um formalismo para definir possíveis funções de distorção neste trabalho, podem ser identificadas algumas regras para a definição da função de distorção.

A função deve ser não-linear, pois uma função de distorção linear iria simplesmente aproximar ou afastar todos elementos no espaço, mas a distribuição, continuaria a mesma, e é justamente a distribuição que se deseja mudar com uma função de distorção.

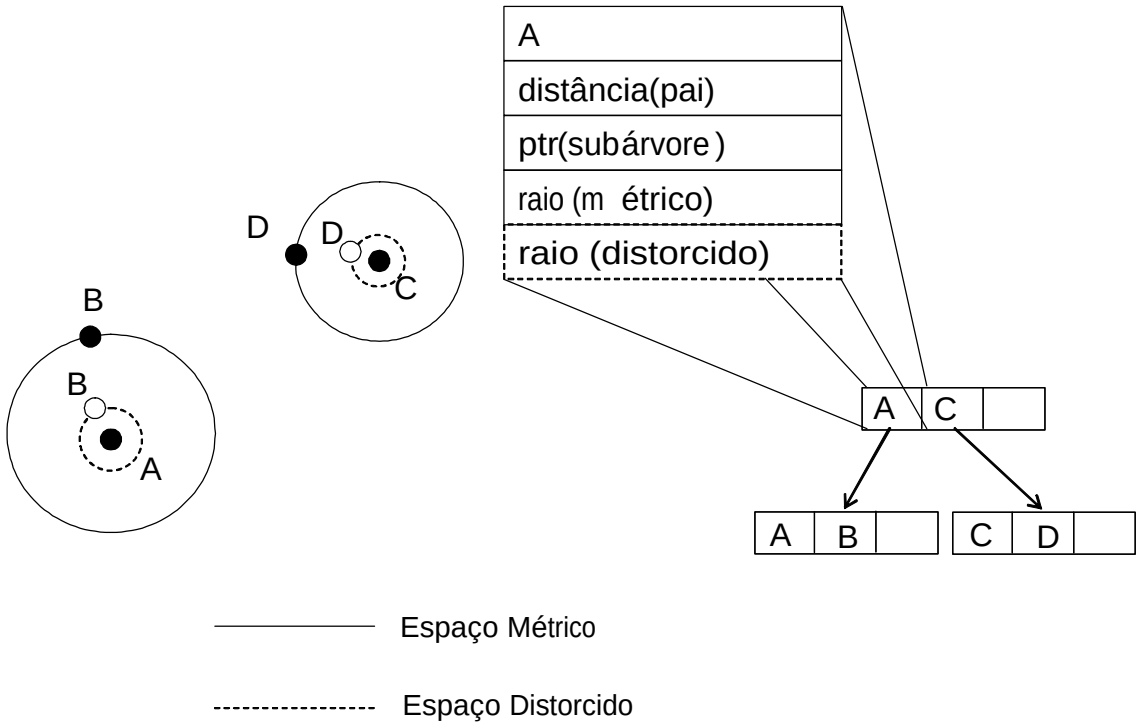


Figura 4.6: Uma árvore TM

Além disso, deve-se garantir que qualquer objeto que está dentro de um raio de consulta no espaço métrico também está dentro do raio de consulta no espaço distorcido, não havendo, dessa forma, falsos negativos. Ou seja, dada uma função de distância métrica d, uma função de distorção t e três objetos o_1, o_2, o_3 no espaço métrico, se $d(o_1, o_2) > d(o_1, o_3)$ então $t(d(o_1, o_2)) > t(d(o_1, o_3))$. Para que isso seja garantido, basta que a função t não seja decrescente em nenhum ponto. Portanto a função de distorção deve ser não-linear e não-decrescente, ou seja, deve ser crescente.

4.7 Experimentos

Foram feitos experimentos usando a árvore TM para verificar sua eficiência. Indexou-se 200 elementos da base XML de bibliografia pelo elemento <TITLE> usando 4kB de tamanho de página, e as funções de distância de Levenshtein e grams (veja Seção 3.2). Vários índices foram construídos usando-se diversas funções para distorcer o espaço. As funções usadas foram:

Arcotangente hiperbólico: $atanh(x) = \frac{1}{2} \cdot ln(\frac{1+x}{1-x})$

potência2: x^2

potência30: x^{30}

divX: $\frac{-1}{x-1} - 1$

potênciaX: $4^x - 1$

suas representações gráficas estão na figura 4.7. Todas elas respeitam as restrições propostas acima.

A comparação do desempenho usando a função de distância baseada em grams é mostrada na figura 4.8. Por questões de escala, o gráfico mostra o desempenho da busca até um raio de $0,6$, para que fiquem mais claras as diferenças (um gráfico com

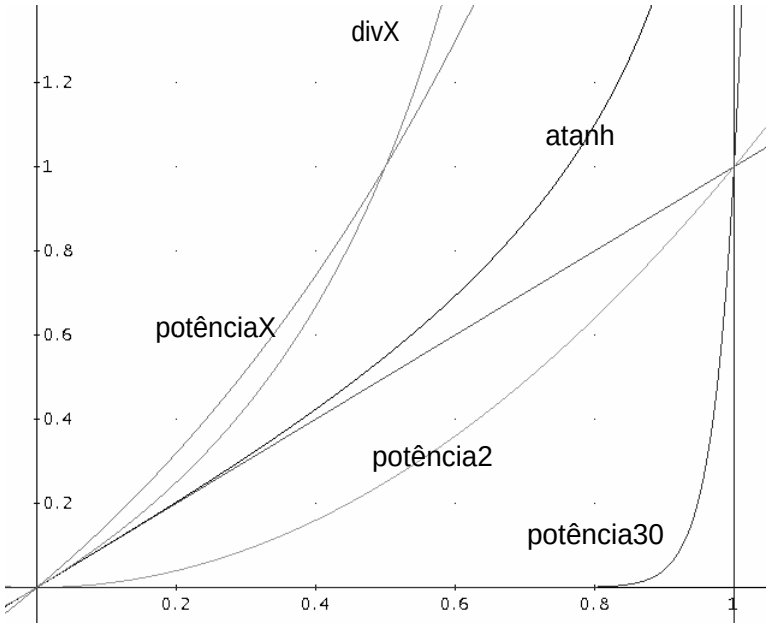


Figura 4.7: Gráfico das funções usadas.

os raios de busca até 1 podem ser vistos na figura 4.9). Claramente, a aplicação dessa técnica de distorção espacial (árvore TM) teve um bom desempenho comparado com a função linear (nenhuma distorção no espaço, ou seja, a árvore M) tanto em relação a cálculos de distância necessários (CPU) quanto a páginas de disco solicitadas (IO).

A figura 4.10 mostra a comparação de desempenho usando-se a função de Levenshtein sem distorção (linear) e com a função de distorção "potência30". O uso da função de distorção para a função de distância de Levenshtein teve uma melhora bastante tímida comparada com a melhora proporcionada na função baseada em grams. Mas pode-se observar no gráfico que o desempenho da busca com a função de Levenshtein já está bastante superior ao desempenho da busca usando-se a função grams. Isso é um indício de que os agrupamentos gerados pela árvore M usando a função de Levenshtein já estavam mais homogêneos e portanto não precisavam ser tão melhorados como no caso da função de grams.

Analisando-se o volume das árvores (figura 4.11), as funções *atanh*, *potência2*, *potência30* e *divX* geraram árvores menores do que a técnica tradicional. A função *potênciaX* gerou árvores maiores, e por isso teve o desempenho prejudicado quando a consulta tem um raio muito grande. Mesmo assim, teve um bom desempenho para raios de buscas pequenos, que é o caso mais comum.

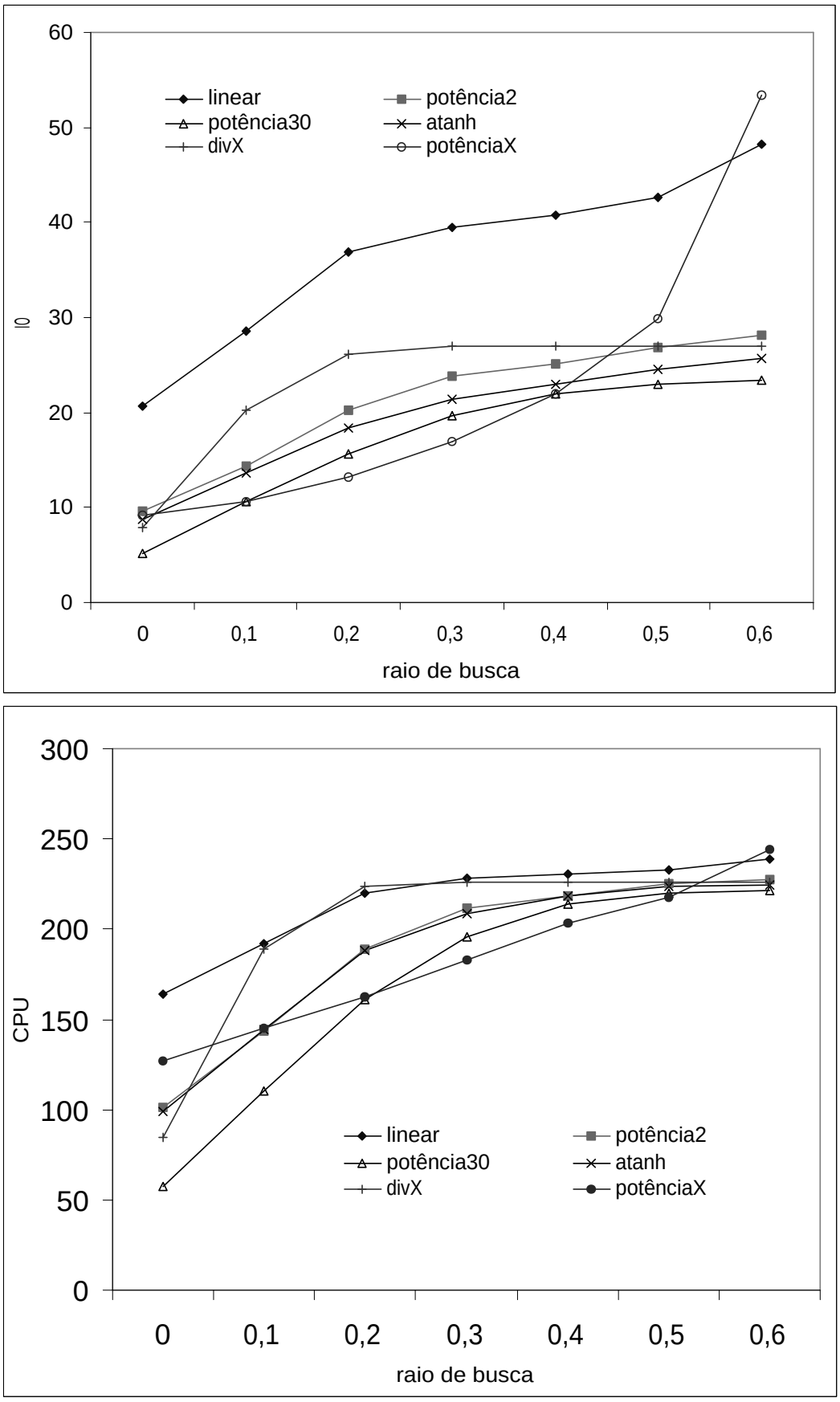
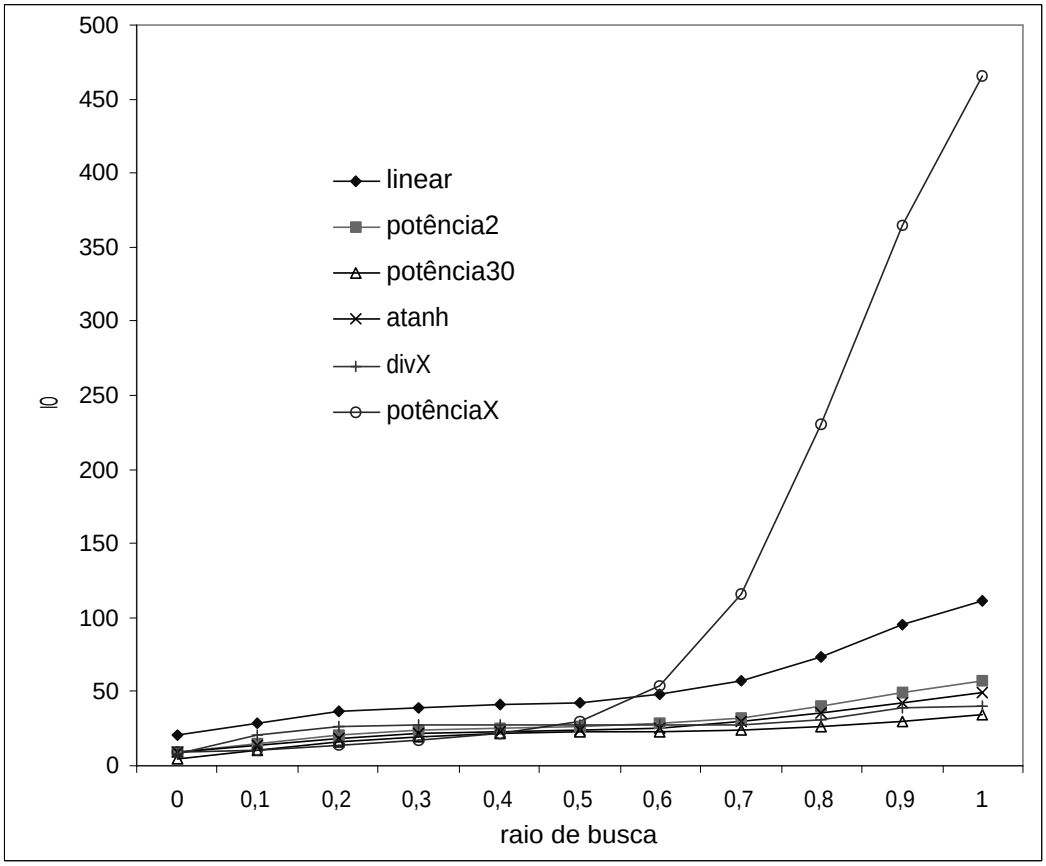


Figura 4.8: Comparação do desempenho da árvore M e a árvore TM, indexando o campo <TITLE>, usando grams.



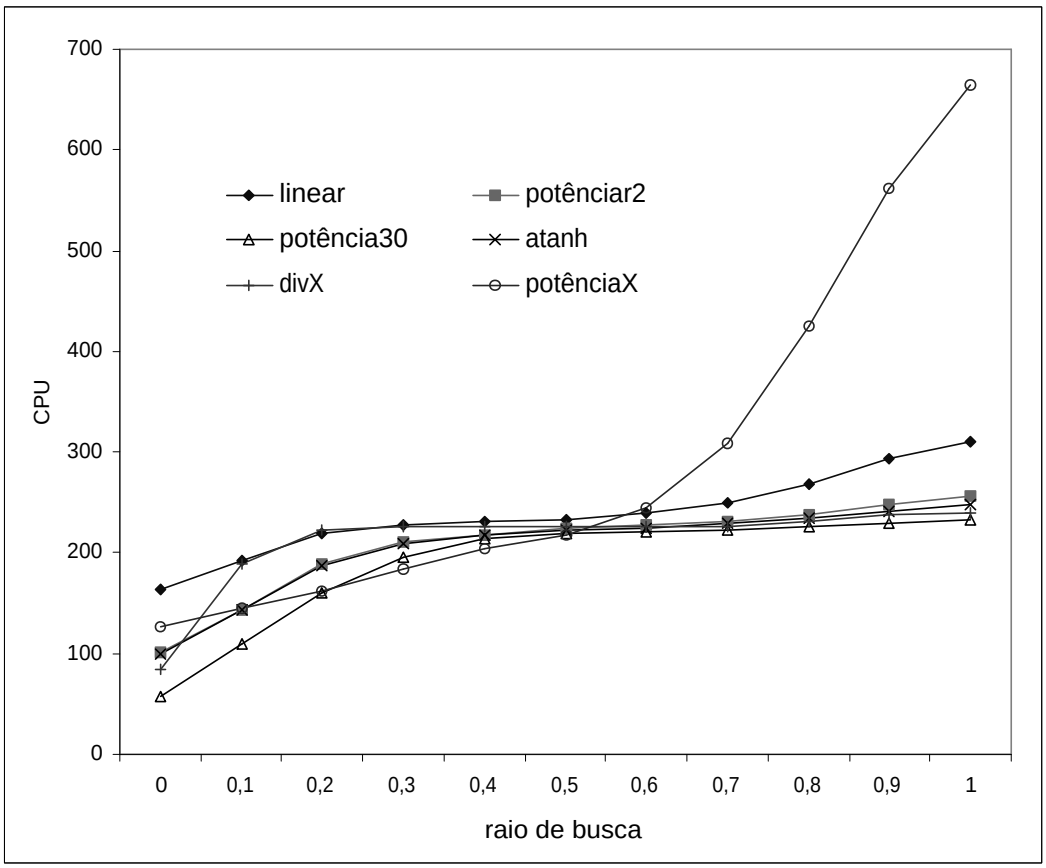


Figura 4.9: Comparação do desempenho da árvore M e a árvore TM, indexando o campo <TITLE>, usando grams até raio de busca 1

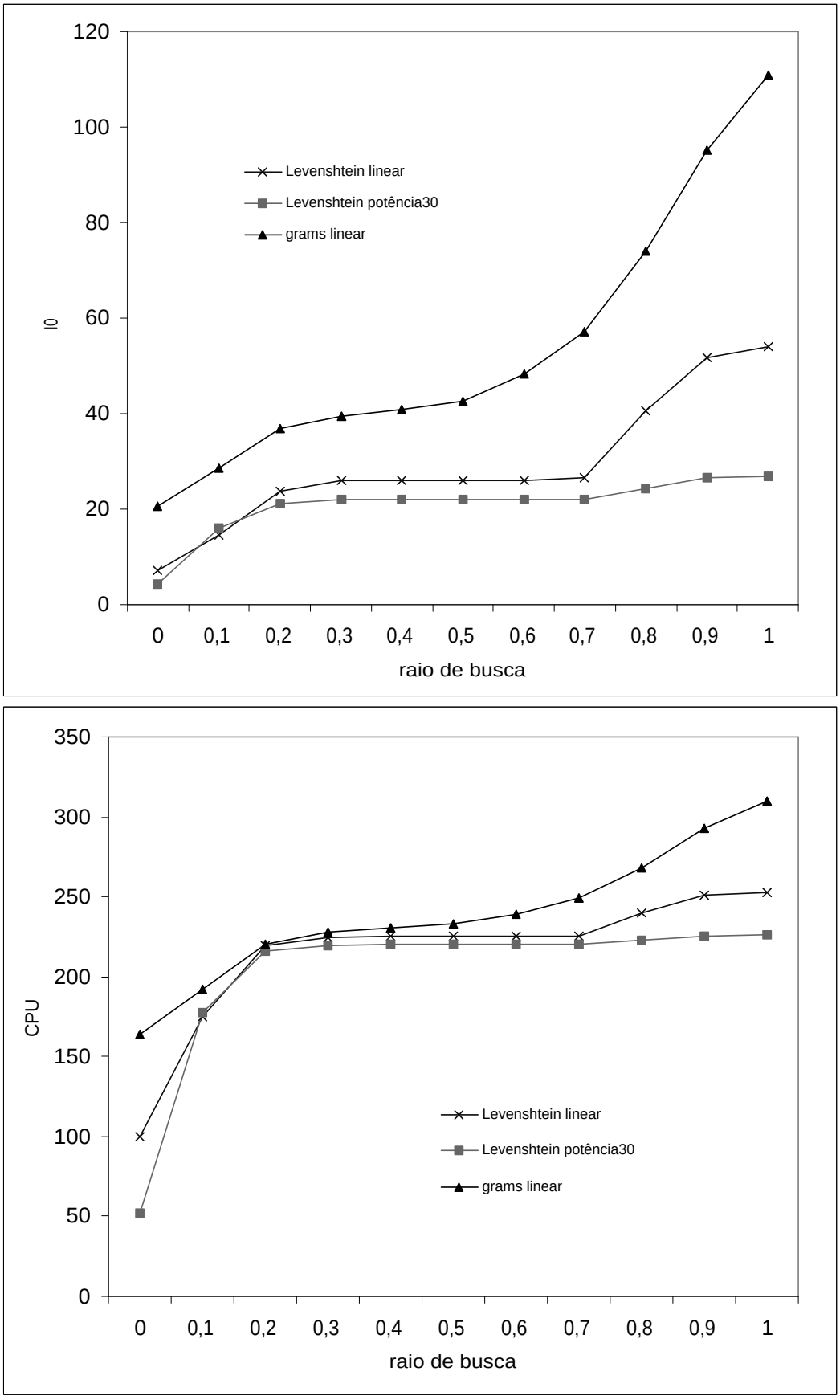


Figura 4.10: Comparação do desempenho da árvore M e a árvore TM, indexando o campo <TITLE>, usando a distância de Levenshtein.

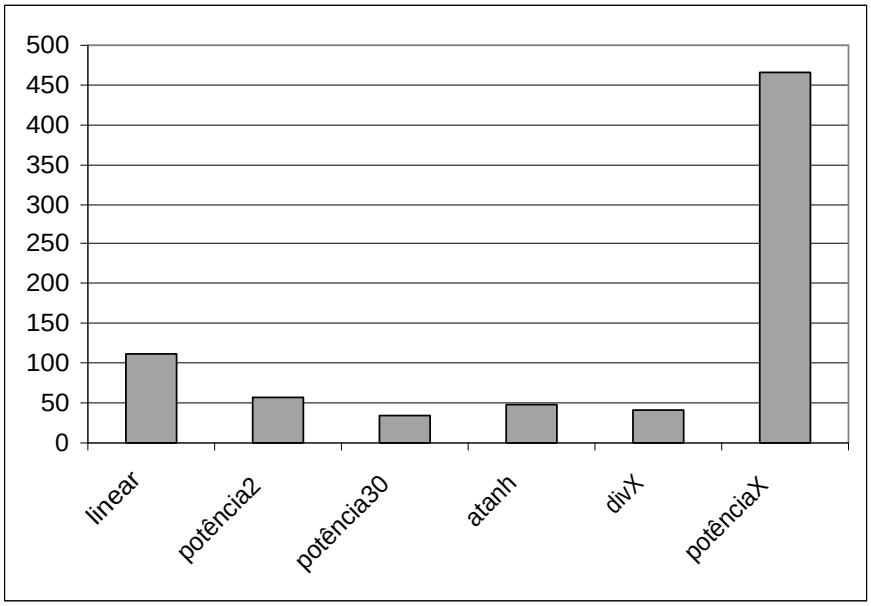


Figura 4.11: Comparação do volume dos índices.

5 CONCLUSÃO

Existem vários tipos de dados como imagem, sons, vídeos, que requerem Métodos de Acesso mais sofisticados. Para suprir essa necessidade, Métodos de Acesso Métricos vêm sendo usados. A árvore M é um importante representante desse grupo, sendo pioneira em importantes características como acesso a memória secundária e dinamismo, servindo de base para Métodos de Acesso mais sofisticados que vieram depois.

No entanto, identificou-se um problema nesse Método de Acesso. Esse Método de Acesso se baseia na construção de grupos representados em uma árvore. Em alguns casos, esse agrupamento pode ficar bastante prejudicado devido a distribuição dos objetos no espaço indexado.

Esse problema foi solucionado introduzindo-se o *espaço distorcido*, onde os objetos são redistribuídos de forma que o algoritmo de agrupamento dos objetos pode identificar melhor em que grupo inserir novos objetos gerando uma árvore cujo acesso ficou mais rápido: a árvore TM.

5.1 Contribuições

São contribuições desse trabalho:

- Experimentos com a árvore M que mostraram o problema de agrupamentos desequilibrados, de forma que pode haver representativos com um grande raio em oposição a outros representativos com raios pequenos e agrupando poucos elementos, desperdiçando espaço de armazenamento.

- A proposta de uma função de distorção, definindo um novo espaço chamado de distorcido.

- A idéia de se usar dois espaços em um Método de Acesso. Até então, sempre se trabalhou em apenas um espaço. Usando-se mais de um espaço, pode-se ter mais de uma visão de um mesmo universo. No caso desse trabalho, aproveitou-se o espaço métrico para fazer a consulta, e o espaço distorcido para indexação dos objetos.

- A solução para o problema identificado na arvore M: definindo-se o espaço distorcido e usando ele em conjunto com o espaço métrico, se criou a árvore TM, que apresentou desempenho no acesso bastante superior a árvore M

As seguintes publicações resultaram desse trabalho:

- NADVORNY, C. F.; HEUSER, C. A. Twisting the Metric Space to Achieve Better Metric Trees. In: SIMPÓSIO BRASILEIRO DE BANCO DE DADOS, SBBD, 19., 2004, Brasília. **Anais...**, 2004. p.178–190.

- NADVORNY, C. F.; HEUSER, C. A. Árvore TM: melhorando Árvores métricas agrupando melhor os objetos. In: ESCOLA REGIONAL DE BANCO DE DADOS, ER-BD, 1., 2005, Porto Alegre. **Anais...**, 2005. p.1–6.

5.2 Trabalhos futuros

Como mencionado no Capítulo 4, não foi definida uma forma de se encontrar a função de distorção ideal para determinado espaço métrico. A análise do espaço métrico determinando a função de distorção ideal para aquele espaço é um trabalho em aberto.

As funções utilizadas para a distorção do espaço foram todas não métricas. Apenas com funções não métricas se conseguiu o efeito de distorção espacial necessário para o melhor agrupamento dos objetos. Mas isso cria o problema na falha da busca, sendo necessário o uso do espaço métrico também. Encontrar uma função métrica que gere uma distribuição espacial adequada ao agrupamento seria, portanto, bastante útil, sendo outro trabalho futuro.

O espaço distorcido definido nesse trabalho foi aplicado apenas a árvore M. No entanto, existe uma série de outros Métodos de Acesso mais sofisticados do que a árvore M que também devem se beneficiar do uso do espaço distorcido. Portanto, a implementação e experimentação com esses outros Métodos de Acesso também está entre os trabalhos futuros.

Os experimentos apresentados aqui foram todos com bases em XML. A árvore TM pode ser usada para indexação de qualquer tipo de objeto imersível no espaço métrico, como som, imagem, texto. Seria interessante ter um base de testes mais abrangente, contendo vários tipos diferentes de objetos indexados e fazer experimentos usando consultas pelos vizinhos mais próximos (NN_k) além da consulta por abrangência (RQ). Além disso, devido a dificuldades técnicas, o volume da base não foi muito grande. Experimentos com bases maiores poderiam mostrar estatísticas mais precisas.

REFERÊNCIAS

BAEZA-YATES, R. A.; CUNTO, W.; MANBER, U.; WU, S. Proximity Matching Using Fixed-Queries Trees. In: ANNUAL SYMPOSIUM ON COMBINATORIAL PATTERN MATCHING, 5., 1994, London, UK. **Proceedings...** [S.l.]: Springer-Verlag, 1994. p.198–212.

BAYER, R.; MCCREIGHT, E. M. Organization and Maintenance of Large Ordered Indexes. **Acta Informatica**, Berlin, v.1, n.3, p.173–189, Feb. 1972.

BECKMANN, N.; KRIEGEL, H.-P.; SCHNEIDER, R.; SEEGER, B. The R*-Tree: an efficient and robust access method for points and rectangles. In: ACM SIGMOD INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA, 1990, Atlantic City, NJ. **Proceedings...** [S.l.: s.n.], 1990. p.322–331.

BENTLEY, J. L. Multidimensional Binary Search Trees Used for Associative Searching. **Communications of the ACM**, [S.l.], v.18, n.9, p.509–517, Sept. 1975.

BENTLEY, J. L. Multidimensional Binary Search Trees in Data Base Applications. **IEEE Transactions on Software Engineering**, [S.l.], v.SE-5, n.4, p.333–340, July 1979.

BERCHTOLD, S.; KEIM, D. A.; KRIEGEL, H.-P. The X-tree: an index structure for high-dimensional data. In: INTERNATIONAL CONFERENCE ON VERY LARGE DATA BASES, VLDB, 22., 1996, Mumbai (Bombay), India. **Proceedings...** [S.l.]: Morgan Kaufmann, 1996. p.28–39. Disponível em <http://www.vldb.org/conf/1996/P028.PDF>. Acesso em: mar. 2003.

BIMBO, A. del. **Visual Information Retrieval**. San Francisco, California: Morgan Kaufmann, 1999.

BÖHM, C.; BERCHTOLD, S.; KEIM, D. A. Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases. **ACM Computing Surveys**, [S.l.], v.33, n.3, p.322–373, Sept. 2001.

BOZKAYA, T.; OZSOYOGLU, M. Distance-Based Indexing for High-Dimensional Metric Spaces. **SIGMOD Record**, New York, v.26, n.2, p.357–368, June 1997. Trabalho apresentado na ACM SIGMOD International Conference on Management of Data; SIGMOD, 1997.

BRAGANHOLO, V. d. P.; FEIJÓ, D. d. V. **Estudo de soluções comerciais para integração relacional/XML**. 2002. Disponível em

<http://www.mycgiserver.com/~diegofeijo/disciplinas/cmp303/ProjetoPesquisa2.pdf>. Acesso em: jan. 2003.

BRIN, S. Near Neighbor Search in Large Metric Spaces. In: INTERNATIONAL CONFERENCE ON VERY LARGE DATA BASES, VLDB, 21., 1995, Zurich, Switzerland. **Proceedings...** San Francisco: Morgan Kaufmann, 1995. p.574–584.

BURKHARD, W. A.; KELLER, R. M. Some Approaches to Best-Match File Searching. **Communications of the ACM**, [S.l.], v.16, n.4, p.230–236, Apr. 1973.

CAVNAR, W. B. N-gram-based text filtering for TREC-2. In: TEXT RETRIEVAL CONFERENCE, TREC, 2., 1993. **Proceedings...** [S.l.: s.n.], 1993.

CHAVEZ, E.; NAVARRO, G.; BAEZA-YATES, R. A.; MARROQUIN, J. L. Searching in metric spaces. **ACM Computing Surveys**, [S.l.], v.33, n.3, p.273–321, 2001.

CHIUEH, T. Content-Based Image Indexing. In: INTERNATIONAL CONFERENCE ON VERY LARGE DATA BASES, VLDB, 20., 1994, Santiago, Chile. **Proceedings...** Hove: Morgan Kaufmann, 1994. p.582–593.

CIACCIA, P.; PATELLA, M. The M2-tree: processing complex multi-feature queries with just one index. In: DELOS NETWORK OF EXCELLENCE WORKSHOP, 1., 2000. **Proceedings...** [S.l.: s.n.], 2000.

CIACCIA, P.; PATELLA, M.; ZEZULA, P. M-tree: an efficient access method for similarity search in metric spaces. In: INTERNATIONAL CONFERENCE ON VERY LARGE DATA BASES, VLDB, 23., 1997. **Proceedings...** [S.l.: s.n.], 1997. p.426–435.

DORNELES, C. F.; LIMA, A. E. N.; HEUSER, C. A. **Experiments on similarity query over XML data sources**. Porto Alegre: Universidade Federal do Rio Grande do Sul, 2003. (RP-337).

FINKEL, R. A.; BENTLEY, J. L. Quad Trees: A data structure for retrieval on composite keys. **Acta Informatica**, Berlin, v.4, n.1, p.1–9, Nov. 1974.

FREESTON, M. A General Solution of the n-dimensional B-tree Problem. In: ACM SIGMOD INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA, 1995, San Jose, California. **Proceedings...** [S.l.: s.n.], 1995. p.80–91.

FUCHS, H.; ABRAM, G. D.; GRANT, E. D. Near real-time shaded display of rigid objects. **Computer Graphics**, Detroit, MI, v.17, n.3, p.65–72, July 1983.

FUCHS, H.; KEDEM, Z. M.; NAYLOR, B. F. On Visible Surface Generation by a Priori Tree Structures. **Computer Graphics**, [S.l.], v.14, n.3, p.124–133, July 1980.

GAEDE, V.; GÜNTHER, O. Multidimensional access methods. **ACM Computing Surveys**, [S.l.], v.30, n.2, p.170–231, June 1998.

GUTTMAN, A. R-trees: a dynamic index structure for spatial searching. **SIGMOD Record (ACM Special Interest Group on Management of Data)**, [S.l.], v.14, n.2, p.47–57, 1984.

HENRICH, A.; SIX, H.-W.; WIDMAYER, P. The LSD tree: spatial access to multidimensional point and nonpoint objects. In: INTERNATIONAL CONFERENCE ON VERY LARGE DATA BASES, 15., 1989, Amsterdam, The Netherlands. **Proceedings...** Palo Alto: Morgan Kaufmann, 1989. p.45–53.

JAGADISH, H. V. Spatial Search with Polyhedra. In: IEEE INTERNATIONAL CONFERENCE ON DATA ENGINEERING, 1990, Los Angeles, CA. **Proceedings...** [S.l.: s.n.], 1990. p.311–319.

LEVENSHTEIN, V. I. Binary codes capable of correcting spurious insertions and deletions of ones. **Russian Problemy Peredachi Informatsii**, [S.l.], v.1, p.12–25, Jan. 1965.

LOMET, D. B.; SALZBERG, B. A Robust Multi-Attribute Search Structure. In: INTERNATIONAL CONFERENCE ON DATA ENGINEERING, 5., 1989, Los Angeles, CA. **Proceedings...** [S.l.: s.n.], 1989. p.296–304.

LOMET, D. B.; SALZBERG, B. The hB-Tree: A multiattribute indexing method with good guaranteed performance. **ACM Transactions on Database Systems**, [S.l.], v.15, n.4, p.625–658, Dec. 1990.

OHSAWA, Y.; SAKAUCHI, M. A New Tree Type Data Structure with Homogeneous Nodes Suitable for a Very Large Spatial Database. In: INTERNATIONAL CONFERENCE ON DATA ENGINEERING, 6., 1990. **Proceedings...** Los Alamitos: IEEE Computer Society Press, 1990.

OOI, B. C. Spatial kd-Tree: an indexing mechanism for spatial databases. In: IEEE COMPUTER SOFTWARE AND APLICATIONS CONFERENCE, 1987. **Proceedings...** [S.l.: s.n.], 1987. p.433–438.

PATELLA, M. **Similarity Search in Multimedia Databases**. 1999. Ph.D. in Electronic and Computer Engineering — Università degli Studi di Bologna, Bologna.

PREPARATA, F. P.; SHAMOS, M. I. **Computational Geometry**. New York: Springer-Verlag, 1985.

ROBINSON, J. T. The K-D-B-Tree: A search structure for large multidimensional dynamic indexes. In: ACM SIGMOD INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA, 1981, Ann Arbor, Michigan. **Proceedings...** New York: ACM Press, 1981. p.10–18.

SAMET, H. The Quadtree and Related Hierarchical Data Structures. **ACM Computing Surveys**, [S.l.], v.16, n.2, p.187–260, June 1984.

SAMET, H.; WEBBER, R. E. Storing a collection of polygons using quadtrees. **ACM Transactions on Graphics**, [S.l.], v.4, n.3, p.182–222, July 1985.

SANTOS FILHO, R.; TRAINA, A. J. M.; TRAINA JÚNIOR, C.; FALOUTSOS, C. Similarity Search without Tears: the OMNI-family of all-purpose access methods. In: INTERNATIONAL CONFERENCE ON DATA ENGINEERING, ICDE, 17., 2001, Heidelberg, Germany. **Proceedings...** Washington: IEEE, 2001. p.623–632.

SCHIWIETZ, M. **Speicherung und anfragebearbeitung komplexer geo-objekte**. 1993. Tese (Doutorado em Ciência da Computação) — Ludwig-Maximilians-Universität München, Germany.

SELLIS, T. K.; ROUSSOPOULOS, N.; FALOUTSOS, C. The R+-Tree: a dynamic index for multi-dimensional objects. In: INTERNATIONAL CONFERENCE ON VERY LARGE DATA BASES, VLDB, 13., 1987, Brighton, England. **Proceedings...** [S.l.]: Morgan Kaufmann, 1987. p.507–518.

TRAINA JÚNIOR, C.; TRAINA, A. J. M.; SANTOS FILHO, R.; FALOUTSOS, C. How to improve the pruning ability of dynamic metric access methods. In: INTERNATIONAL CONFERENCE ON INFORMATION AND KNOWLEDGE MANAGEMENT, CIKM, 11., 2002. **Proceedings...** New York: ACM Press, 2002. p.219–226.

TRAINA JÚNIOR, C.; TRAINA, A. J. M.; SEEGER, B.; FALOUTSOS, C. Slim-Trees: high performance metric trees minimizing overlap between nodes. In: INTERNATIONAL CONFERENCE ON EXTENDING DATABASE TECHNOLOGY, EDBT, 7., 2000, Konstanz, Germany. **Proceedings...** [S.l.]: Springer, 2000. p.51–65.

UHLMANN. Satisfying General Proximity / Similarity Queries with Metric Trees. **IPL: Information Processing Letters**, [S.l.], v.40, 1991.

VIEIRA, M. R. **DBM-Tree**: Método de acesso métrico sensível à densidade local. 2004. Dissertação (Mestrado em Ciência da Computação) — Universidade de São Paulo.

W3C. **XML Path Language (XPath) 1.0**. Disponível em <http://www.w3.org/TR/xpath>. Acesso em: dez. 2002.

W3C. **XQuery 1.0: an xml query language** . Disponível em <http://www.w3.org/TR/xquery>. Acesso em: dez. 2002.

YIANILOS, P. N. Excluded Middle Vantage Point Forests for Nearest Neighbor Search. In: DIMACS IMPLEMENTATION CHALLENGE WORKSHOP: NEAR NEIGHBOR SEARCHES, 6., 1999, Baltimore, Maryland. **Proceedings...** [S.l.: s.n.], 1999.

ZHOU, X. et al. A New Dynamical Multidimensional Index for Metric Spaces. In: AUSTRALIAN DATABASE CONFERENCE, 4., 2003, Adelaide, Australia. **Proceedings...** [S.l.: s.n.], 2003. p.1–8.