Operação de busca exata aos *k*-vizinhos mais próximos reversos em espaços métricos

Willian Dener de Oliveira

SERVIÇO DE PÓS-GRADUAÇÃO DO ICMC-USP

Data de Depósito:

Assinatura:_____

Operação de busca exata aos *k*-vizinhos mais próximos reversos em espaços métricos

Willian Dener de Oliveira

Orientador: ***Prof. Dr. Caetano Traina Júnior***

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

USP – São Carlos
Fevereiro/2010

Dedicatória

À minha esposa Fernanda, com amor, admiração
e gratidão por sua compreensão, carinho,
presença e incansável apoio ao longo do período
de elaboração deste trabalho.
Aos meus pais, Gilmar e Eliana, que sempre me
motivaram e apoiaram.

Agradecimentos

À Deus.

Ao meu orientador, Prof. Dr. Caetano Traina Júnior, agradeço o apoio e a confiança que dispensou durante a elaboração deste trabalho, mesmo nos momentos que eu não acreditava. À Profª Drª Agma Juci Machado Traina, devido aos conselhos, sugestões e confiança, principalmente no período em que trabalhei no Grupo como Especialista de Laboratório.

A todos do Grupo de Base de Dados e Imagens, que nos anos de convivência, muito me ensinaram, contribuindo para meu crescimento pessoal, científico e intelectual.

A minha esposa Fernanda e meus pais, para os quais dedico este trabalho, pelo incentivo e apoio incondicional, me fortalecendo nos momentos difíceis.

Aos amigos de longa data e também aqueles conquistados no Grupo de Base de Dados e Imagens que, direta ou indiretamente, contribuíram com seu apoio.

À FAPESP (Fundação de Amparo à Pesquisa do Estado de São Paulo) e CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior), pelo apoio financeiro à realização deste trabalho.

Sumário

Dedicatória **iii**

Agradecimentos **v**

Lista de Algoritmos **xv**

Lista de Abreviaturas e Siglas **xvii**

Lista de Símbolos **xix**

Resumo **xxi**

Abstract **xxiii**

1 Introdução **25**

1.1 Motivação 26

1.2 Objetivos do trabalho 27

1.3 Organização do documento 27

2 Conceitos fundamentais **29**

2.1 Considerações iniciais 29

2.2 Espaços métricos 29

2.2.1 Métricas 30

2.3 Consultas por similaridade 32

2.3.1 Consulta por abrangência 32

2.3.2 Consulta aos k-vizinhos mais próximos 33

2.3.3 Consulta aos k-vizinhos mais próximos reverso 34

2.4 Métodos de acesso espaciais 34

2.4.1 R-tree e algumas variantes 35

2.4.2 Algoritmos de consultas por similaridade em métodos de acesso espaciais . . . 37
2.5 Métodos de acesso métrico . . . 38
2.5.1 Técnicas de Burkhard e Keller . . . 39
2.5.2 *Fixed queries tree* . . . 40
2.5.3 *Vantage point tree* . . . 40
2.5.4 *Generalized hyperplane tree* . . . 41
2.5.5 *AESA* . . . 42
2.5.6 *Geometric near-neighbor access tree* . . . 43
2.5.7 *M-tree* . . . 43
2.5.8 *Slim-tree* . . . 45
2.5.9 Família OMNI . . . 47
2.6 Imersão de espaços métricos e sua aplicação em MAMs . . . 48
2.7 Considerações finais . . . 49

3 k-vizinhos mais próximos reversos 51
3.1 Considerações iniciais . . . 51
3.2 Conceitos sobre consultas $RkNN$ . . . 51
3.2.1 Variantes . . . 52
3.3 Algoritmos RNN euclidianos com pré-computação . . . 54
3.3.1 RNN-tree . . . 54
3.3.2 RdNN-tree . . . 55
3.3.3 Outro método baseado em pré-computação . . . 56
3.3.4 Considerações sobre os algoritmos com pré-computação . . . 56
3.4 Algoritmos RNN euclidiano com propriedades para espaços de duas ou três dimensões . . . 56
3.4.1 A abordagem de Stanoi et al. (2000) . . . 58
3.4.2 A abordagem de Stanoi et al. (2001) para o caso bicromático . . . 58
3.5 Algoritmos $RkNN$ com aproximação . . . 59
3.5.1 Abordagem de Singh et al. (2003) . . . 59
3.5.2 Abordagem ERkNN . . . 61
3.6 Abordagem TPL . . . 62
3.7 $RkNN$ em espaços métricos . . . 63
3.7.1 Abordagem de Tao et al. (2006) . . . 63
3.7.2 Abordagem MRkNNCoP-tree . . . 64
3.7.3 Abordagem AMRkNN-tree . . . 65
3.8 A abordagem *k-counting* e *k-browsing* . . . 66
3.9 Abordagem AKKRZ . . . 66
3.10 Considerações finais . . . 67

4 Algoritmo $RkNN$-MG e podas utilizando lei dos cossenos **69**
4.1 Considerações iniciais 69
4.2 Visão geral do algoritmo $RkNN$-MG 70
4.3 Conceito min_κ utilizando geometria métrica 71
4.4 Algoritmo $RkNN$-MG 82
4.5 Considerações finais 85

5 Experimentos **87**
5.1 Conceituação inicial 87
5.2 Experimentos com o conjunto de dados *Eigen Faces* 89
5.3 Experimentos com o conjunto de dados *Cidades Americanas* 89
5.4 Experimentos com o conjunto de dados *Currency* 90
5.5 Experimentos com o conjunto de dados *Histograma* 91
5.6 Experimentos com o conjunto de dados *Faces (SISAP)* 92
5.7 Experimentos com o conjunto de dados *EnglishWords* 92
5.8 Teste de escalabilidade 92
5.9 Considerações Finais 94

6 Conclusões e trabalhos futuros **97**
6.1 Principais contribuições 98
6.2 Trabalhos futuros 98

Lista de Figuras

1.1 Conjunto de dados com três agrupamentos e duas anomalias 27

2.1 Representação das formas geométricas geradas para as métricas L_1, L_2 e L_∞ para os pontos equidistantes à distância ξ a partir do elemento central s_q 32
2.2 Exemplos de consultas por similaridade sobre o elemento s_q num espaço bidimensional euclidiano 33
2.3 Exemplo de uma R-tree: representação estrutural e representação espacial com os MBRs (Guttman, 1984) 36
2.4 MinDist, MaxDist e MinMaxDist 38
2.5 Exemplo da VP-tree com raiz s_r: representação bidimensional e a correspondente árvore 41
2.6 Exemplo do primeiro nível de uma *Generalized Hyperplane Tree* 42
2.7 Exemplo do primeiro nível de uma *Geometric Near-neighbor Access Tree* . 43
2.8 Exemplo de particionamento de nó usando MST (Traina Júnior et al, 2000) 47
2.9 Exemplo de utilização do conceito OMNI com dois focos 48

3.1 $kNN(s_q, 2)$ e $RkNN(s_q, 2)$ exemplificando que o relacionamento entre $RkNN$ e kNN é assimétrico 52
3.2 Exemplo de $RkNN$ bicromático e monocromático para $k = 1$ 53
3.3 Ilustração do método de Stanoi et al (2000) baseado na propriedade 3.1 . . 57
3.4 Exemplo de uma consulta RNN em duas fases do método de Stanoi et al (2001) 59
3.5 Algoritmo proposto para RNN por Singh et al (2003, p. 93) 60
3.6 Ilustração da poda por semi-espaços (Tao et al, 2004) 63
3.7 Ilustração das regras de poda (Tao et al, 2006) 64
3.8 Ilustração do uso de distâncias kNN conservativa e progressiva (Achtert et al, 2006b) 65

4.1 Visão geral do algoritmo *RkNN-MG* 70
4.2 Ilustração do cálculo da distância ao hiperplano 71
4.3 Ilustração de $height(s_q, s_i; s_j)$ 72
4.4 Ilustração da propriedade 4.2 73
4.5 Ilustração de propriedade 4.3 73
4.6 Exemplo de como encontrar os pontos de tangência 75
4.7 Ilustração do lema 4.1 76
4.8 Ilustração do lema 4.2 77
4.9 Ilustração do lema 4.3 79
4.10 Exemplo de min_κ 79

5.1 Gráficos de desempenho do algoritmo *RkNN-MG* sobre o conjunto de dados *Eigen Faces*. 90
5.2 Gráficos de desempenho do algoritmo *RkNN-MG* sobre o conjunto de dados *Cidades Americanas* 90
5.3 Gráficos de desempenho do algoritmo *RkNN-MG* sobre o conjunto de dados *Currency* 91
5.4 Gráficos de desempenho do algoritmo *RkNN-MG* sobre o conjunto de dados *Color Histrogram*. 91
5.5 Gráficos de desempenho do algoritmo *RkNN-MG* sobre o conjunto de dados *Faces (SISAP)* 92
5.6 Gráficos de desempenho do algoritmo *RkNN-MG* sobre o conjunto de dados *EnglishWords* 93
5.7 Gráficos de desempenho do algoritmo *RkNN-MG* sobre o conjunto de dados *SC*. 94
5.8 Gráficos de desempenho do algoritmo *RkNN-MG* sobre o conjunto de dados *SU*. 94

Lista de Tabelas

3.1 Resumo dos métodos para implementação da consulta $RkNN$ apresentados neste capítulo 67

5.1 Conjunto de dados utilizados nos experimentos 88
5.2 Porcentagem de redução do conjunto SC 93
5.3 Porcentagem de redução do conjunto SU 95

Lista de Algoritmos

3.1 Inserir 55
3.2 Remover 55
3.3 $ERkNN(T, s_q, k)$ 61
4.1 FindRknn 83
4.2 DequeueWithUpdate 84

Lista de Abreviaturas e Siglas

AESA	*Approximating eliminating search algorithm*
FHQT	*Fixed-height fixed queries tree*
FQT	*Fixed queries tree*
GHT	*Generalized hyperplane tree*
GNAT	*Geometric near-neighbor access tree*
kNN	k-vizinhos mais próximos
MA	Método de acesso
MAE	Método de acesso espacial
MAEP	Método de acesso espacial pontual
MAENP	Método de acesso especial não-pontual
MAM	Método de acesso métrico
MBR	Retângulo envolvente mínimo
MST	Minimum spanning tree
NN	vizinho mais próximo
$RkNN$	k-vizinhos mais próximos reversos
RNN	Vizinho mais próximo reverso
$RRNN$	Vizinho mias próximo reverso ranqueado
R_q	Consulta por abrangência
SGBD	Sistema gerenciador de banco de dados
SIG	Sistema de informações geográficas
VPT	*Vantage Point Tree*

Lista de Símbolos

n	Número de elementos num conjunto de dados
$\mathbb{M}$	Espaço métrico
d	Função de distância ou métrica
S	Conjunto de dados
S_i	Subconjunto de dados i
s_i	Elemento i
$\mathbb{S}$	Domínio do elementos válidos
ξ	Raio de consulta
E	Dimensão
L_p	Métricas de Minkowski
$\mathbb{R}^+$	Domínio dos número reais positivos
c_i	cadeia de caracteres i
s_q	elemento central da consulta
s_{r_i}	elemento representativo i
R_i	retângulo envolvente mínimo i
N_i	nó i de uma árvore

Resumo

OLIVEIRA, W. D. **Operação de Busca Exata aos k-Vizinhos mais Próximos Reversos.** 2010. 107f. Dissertação (Mestrado) - Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Paulo, 2010.

A complexidade dos dados armazenados em grandes bases de dados aumenta cada vez mais, criando a necessidade de novas operações de consulta. Uma classe de operações que tem apresentado interesse crescente são as chamadas Consultas por Similaridade, sendo as mais conhecidas as consultas por Abrangência (R_q) e por k-Vizinhos mais Próximos (kNN), sendo que esta última obtém quais são os k elementos armazenados mais similares a um dado elemento de referência. Outra consulta que é interessante tanto para consultas diretas quanto como parte de operações de análises mais complexas é a operação de consulta aos k-Vizinhos mais Próximos Reversos ($RkNN$). Seu objetivo é obter todos os elementos armazenados que têm um dado elemento de referência como um dos seus k elementos mais similares. Devido à complexidade de execução da operação de $RkNN$, a grande maioria das soluções existentes restringem-se a dados representados em espaços multidimensionais euclidianos (nos quais estão definidas também operações cardinais e topológicas, além de se considerar a similaridade como sendo a distância Euclidiana entre dois elementos), ou então obtém apenas respostas aproximadas, sujeitas à existência de falsos negativos. Várias aplicações de análise de dados científicos, médicos, de engenharia, financeiros, etc. requerem soluções eficientes para o problema da operação de $RkNN$ sobre dados representados em espaços métricos, onde os elementos não podem ser considerados estar em um espaço nem Euclidiano nem multidimensional. Num espaço métrico, além dos próprios elementos armazenados existe apenas uma função de comparação métrica entre pares de objetos. Neste trabalho, são propostas novas podas de espaço de busca e o algoritmo $RkNN$-MG que utiliza essas novas podas para solucionar o problema de consultas $RkNN$ exatas em espaços métricos sem limitações. Toda a proposta supõe que o conjunto de dados está em um espaço métrico imerso isometricamente em espaço

euclidiano e utiliza propriedades da geometria métrica válida neste espaço para realizar podas eficientes por lei dos cossenos combinada com as podas tradicionais por desigualdade triangular. Os experimentos demonstram comparativamente que as novas podas são mais eficientes que as tradicionais podas por desigualdade triangular, tendo desempenhos equivalente quando comparadas em conjuntos de alta dimensionalidade ou com dimensão fractal alta. Assim, os resultados confirmam as novas podas propostas como soluções alternativas eficientes para o problema de consultas $RkNN$.

Palavras-chaves: Vizinhos Mais Próximos Reversos. Consultas por Similaridade. $RkNN$. Espaço Métrico. Indexação.

Abstract

OLIVEIRA, W. D. **Answering Exact Reverse k-Nerarest Neighbors Queries in Metric Spaces.** 2010. 107f. Dissertação (Mestrado) - Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Paulo, 2010.

Data stored in large databases present an ever increasing complexity, pressing for the development of new classes of query operators. One such class, which is enticing an increasing interest, is the so-called Similarity Queries, where the most common are the similarity range queries (R_q) and the k-nearest neighbor queries (kNN). A k-nearest neighbor query aims at retrieving the k stored elements nearer (or more similar) to a given reference element. Another important similarity query is the reverse k-nearest neighbor ($RkNN$), useful both for queries posed directly by the analyst and for queries that are part of more complex analysis processes. The objective of a reverse k-nearest neighbor queries is obtaining the stored elements that has the query reference element as one of their k-nearest neighbors. As the $RkNN$ operation is a rather expensive operation, from the computational standpoint, most existing solutions only solve the query when applied over Euclidean multidimensional spaces (as these spaces also define cardinal and topological operations besides the Euclidean distance between pairs of elements) or retrieve only approximate answers, where false negatives can occur. Several applications, like the analysis of scientific, medical, engineering or financial data, require efficient and exact answers for the $RkNN$ queries over data which is frequently represented in metric spaces, that is where no other property besides the similarity measure exists. Therefore, for applications handling metrical data, the assumption of Euclidean metric or even multidimensional data cannot be used. In this work, we propose new pruning rules based on the law of cosines, and the $RkNN$-MG algorithm, which uses them to solve $RkNN$ queries in a way that is exact, faster than the existing approaches, that is not limited for any value of k, and that can be applied both over static and over dynamic datasets. The new pruning rules assume that the data set is in a metric space that can be embedded

into an Euclidean space and use metric geometry properties valid in this space to perform effective pruning based on the law of cosines combined with the traditional pruning based on the triangle inequality property. The experiments show that the new pruning rules are alkways more efficient than the traditional pruning rules based solely on the triangle inequality. The experiments show that for high high dimensionality datasets, or for metric datasets with high fractal dimensionality, the performance improvement is smaller than for for lower dimensioinality datasets, but it's never worse. Thus, the results confirm that the our pruning rules are efficient alternative to solve $RkNN$queries in general.

Keywords: Reverse k-nearest neighbor. Similarity query. $RkNN$. Metric space. Access Method.

Capítulo

1

Introdução

Os sistemas de gerenciamento de banco de dados (SGBDs) foram criados para armazenar e recuperar grandes volumes de dados, garantindo que possam ser recuperados de maneira eficiente, sempre fornecendo respostas precisas às consultas. Por muito tempo, estes sistemas trataram apenas consultas envolvendo números e pequenas cadeias de caracteres. No entanto, com a contínua evolução da computação, dados mais complexos passaram a ser armazenados e sua organização em bases de dados passou a ser um objetivo importante. Dentre os dados complexos que são armazenados atualmente estão, por exemplo: dados multimídia (como imagens, áudio e vídeo), informações geo-referenciadas (como coordenadas geográficas), séries temporais e sequências de proteínas.

Para realizar consultas sobre dados complexos, a comparação de elementos baseada em igualdade ($=$ e $\neq$) tem pouca utilidade prática, pois a existência de dois elementos exatamente iguais é muito rara. As operações de comparação relacionais (baseadas na precedência entre qualquer par de elementos, usualmente expressas pelos símbolos $<$, $\leq$, $\geq$ e $>$), geralmente também não são aplicáveis, pois os dados complexos são usualmente representados em domínios que não atendem à propriedade da relação de ordem total. Decorrente dessa situação, as operações de comparação usadas entre elementos em conjuntos de dados complexos são geralmente baseadas em similaridade. As consultas por similaridade retornam os elementos mais semelhantes ao elemento de referência da consulta, sendo que as operações de consultas por similaridade mais conhecidas são a consulta por abrangência (R_q) e a consulta por k-vizinhos mais próximos (kNN), sendo que esta última obtém quais são os k elementos armazenados mais similares a um dado elemento de referência.

Outra consulta por similaridade interessante tanto para consultas diretas quanto como parte de operações de análises mais complexas é a consulta aos k-vizinhos mais próximos reversos ($RkNN$). Seu objetivo é obter todos os elementos armazenados que têm um dado elemento de referência como um dos seus k elementos mais similares.

1.1 Motivação

Admitindo que seja n o número de elementos no conjunto de dados, uma solução simples para o algoritmo da operação $RkNN$ tem complexidade $O(n^2)$. Devido à essa complexidade, a maioria das soluções existentes restringem-se a trabalhar com dados representados em espaços multidimensionais euclidianos, nos quais são definidas também operações cardinais e topológicas. As soluções nesses espaços normalmente consideram a dissimilaridade como sendo a distância euclidiana entre dois elementos ou obtêm apenas respostas aproximadas, sujeitas à existência de falsos positivos. Poucas são as soluções para espaços métricos, nos quais existem apenas os elementos armazenados e uma função de comparação entre os pares de elementos. Além disso, essas soluções possuem restrições quanto ao limite superior do parâmetro k ou apresentam resultados aproximados.

No entanto, várias aplicações de análise de dados científicos, médicos, de engenharia, financeiros, dentre outras, requerem soluções eficientes para o problema da operação de $RkNN$ sobre dados representados em espaços métricos, no qual os elementos não podem ser considerados estar em um espaço nem euclidiano nem multidimensional. A seguir, são apresentados algumas aplicações que motivam o desenvolvimento de trabalhos sobre essa consulta:

Biologia molecular: Sempre que novas sequências biológicas são descobertas, há a necessidade de testá-las com relação ao interesse e originalidade. Com uma nova sequência, consultas $RkNN$ podem ser aplicadas nas grandes bases de sequências que armazenam sequências de moléculas biológicas com função conhecida. Os resultados dessas consultas são utilizados para verificar a originalidade desta nova sequência (Achtert et al, 2007).

Sistemas de apoio à decisão: Tome-se como exemplo uma rede de farmácias que deseja avaliar um local para abrir um novo estabelecimento. Dadas várias possibilidades, a estratégia é escolher a área que pode atrair o maior número de clientes. A consulta $RkNN$ em uma base de dados de habitantes do município retorna um conjunto de pessoas que tem a nova farmácia como a mais próxima deles, sendo então potenciais clientes do estabelecimento comercial (Korn e Muthukrishnan, 2000).

Detecção de agrupamentos e anomalias: O exemplo da figura 1.1 é apresentado usando um espaço euclidiano bidimensional, pois sua visualização em espaços métricos é difícil, porém sua discussão continua válida para espaços métricos em geral.

Pontos em dois grupos na parte superior da figura são muito próximos um do outro, enquanto que as distâncias entre os elementos do agrupamento na parte inferior são relativamente grandes. Como sugerido por Nanopoulos et al (2001), a escolha de um valor apropriado de k possibilita encontrar pontos que não têm nenhum $RkNN$, sendo denominado anomalias (um elemento que não pertence a nenhum agrupamento). Por exemplo, para $k = 3$, pontos s_1 e s_2 são anomalias na figura 1.1 (Tao et al, 2006).

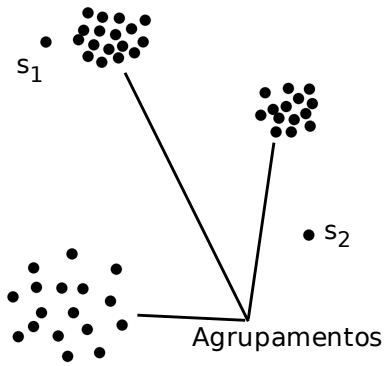


Figura 1.1: Conjunto de dados com três agrupamentos e duas anomalias

1.2 Objetivos do trabalho

Essa dissertação propõe uma nova abordagem para obter a resposta exata de operações de consultas aos k-vizinhos mais próximos reversos em espaços métricos, utilizando métodos de acesso métricos (MAMs) para realizar podas no espaço de busca e permitam uma execução eficiente. Especificamente, assumindo-se que o espaço métrico é imerso isometricamente em um espaço euclidiano, são propostas novas estratégias de podas que utilizam propriedades geométricas válidas neste espaço e obtêm podas mais eficientes do espaço.

1.3 Organização do documento

Além deste capítulo introdutório, esta dissertação é organizada em mais cinco capítulos:

Capítulo 2 Conceitua espaço métrico e espaço multidimensional. Ainda neste capítulo são apresentadas as métricas, as consultas por similaridade mais importantes e os métodos de acesso relevantes na literatura, sendo enfatizados os métodos de acesso métrico.

Capítulo 3 Contém os conceitos e os principais algoritmos já desenvolvidos para a consulta $RkNN$.

Capítulo 4 Apresenta a nova estratégia desenvolvida para a definição de novas podas utilizando lei dos cossenos e o pseudocódigo desenvolvido para demonstrar a eficiência da consulta $RkNN$ utilizando essa nova estratégia.

Capítulo 5 Define o ambiente de testes utilizado para a avaliação da abordagem proposta e descreve os resultados dos experimentos realizados, comparando os resultados obtidos com a abordagem padrão de podas utilizando desigualdade triangular. Em adição, são apresentados alguns experimentos demonstrando empiricamente as vantagens da nova abordagem em relação ao estado da arte em consultas $RkNN$.

Capítulo 6 Conclui a dissertação, apresentando as contribuições e as sugestões de trabalhos futuros.

Capítulo

2

Conceitos fundamentais

2.1 Considerações iniciais

Consultas por similaridade são operações importantes para aplicações em bases de dados que envolvem elementos complexos. Por definição, uma consulta por similaridade busca os elementos de um conjunto de dados que sejam similares a um elemento dado na consulta, baseando-se numa medida de comparação.

Nesse capítulo é realizado um levantamento sobre consultas por similaridade em espaços métricos e sobre métodos de acesso espaciais e métricos. Inicia-se esse levantamento com a conceituação de espaço métrico e de um outro espaço usado com frequência, o espaço multidimensional com função de distância definida. A seguir, apresentam-se as métricas mais importantes, as consultas por similaridade e descrevem-se os métodos de acesso (MAs) mais relevantes na literatura, com enfase nos métodos de acesso métrico.

2.2 Espaços métricos

Um espaço métrico $\mathbb{M}$ é representado por um par $\langle \mathbb{S}, d \rangle$, onde $\mathbb{S}$ é o domínio de elementos válidos e $d : \mathbb{S} \times \mathbb{S} \rightarrow \mathbb{R}^+$ é uma função de distância, chamada métrica, que atende às seguintes condições (Chávez et al, 2001; Copson, 1968):

1. **Identidade**: $d(s_1, s_1) = 0$;
2. **Simetria**: $d(s_1, s_2) = d(s_2, s_1)$;
3. **Não-negatividade:** $d(s_1, s_2) > 0$ para qualquer $s_1 \neq s_2$;

4. **Desigualdade triangular**: $d(s_1, s_2) \leq d(s_1, s_3) + d(s_3, s_2)$;
   $\forall\ s_1,\ s_2,\ s_3\ \in\ \mathbb{S}$.

Espaços métricos podem ser definidos sobre os domínios de espaços multidimensionais, assim como sobre conjuntos de elementos adimensionais, desde que, para isto, seja definida uma métrica adequada.

Intuitivamente, os elementos de um espaço multidimensional de dimensão $E > 1$ podem ser vistos como pontos no espaço, onde a coordenada do ponto em cada dimensão corresponde ao valor de um dos atributos do conjunto de dados. Trabalhos em espaços multidimensionais têm, em sua grande maioria, explorado as propriedades geométricas desses espaços, mas normalmente as técnicas produzidas por esses trabalhos não podem ser estendidas para espaços métricos, nos quais a única informação disponível é a distância entre os elementos.

Num SGBD, um conjunto de elementos $S\ \subseteq\ \mathbb{S}$ representa o conjunto de elementos indexados sobre o qual as consultas são efetuadas. Esse conjunto é frequentemente chamado a *base de dados* representada no espaço métrico.

2.2.1 Métricas

Uma métrica ou função de distância d pode ser interpretada como uma função de cálculo de dissimilaridade entre pares de elementos do conjunto ao qual está associada. Por convenção, adota-se que quanto mais similares dois elementos são entre si, menor o valor dessa função. Consequentemente, valores maiores são retornados para elementos muito distintos. Dessa maneira, as operações de busca por similaridade tornam-se naturais dentro de espaços métricos.

Em geral, as funções de distância são definidas para aplicações específicas ou para domínios de aplicações. Por isso, normalmente, uma função de distância é definida por um especialista da aplicação, porém a definição dessa função não se restringe a um tipo de consulta que possa ser necessária. Nessa seção, são apresentados exemplos de métricas relevantes que podem ser aplicadas a vários tipos de dados e que são importantes para os experimentos apresentados neste trabalho.

Contudo, antes de apresentar essas métricas, há de distinguí-las em dois grupos com relação ao valor de retorno (contradomínio):

- **função de distância discreta**: o conjunto de valores de retorno da função é um conjunto relativamente pequeno de valores;

- **função de distância contínua**: o conjunto de valores de retorno é infinito ou muito grande.

A métrica de *Levenshtein* (descrita a seguir) aplicada em palavras de um dicionário é um exemplo de métrica discreta, enquanto que a métrica euclidiana é um exemplo de métrica contínua.

Métrica de Levenshtein

A métrica de *Levenshtein* (Levenshtein, 1966), conhecida também como L_{edit} ou distância de edição, é comumente utilizada para avaliar a similaridade entre duas cadeias de caracteres. A similaridade é calculada como o menor número de operações de edições elementares (inserções, remoções e substituições de caracteres) necessário para transformar uma cadeia de caracteres c_1 numa outra cadeia c_2. Uma generalização dessa métrica altera o cálculo definindo custos diferentes para cada tipo de operação de edição elementar e também permite definir custos diferentes para cada par de caracteres (Zezula et al, 2006).

Apesar dessa generalização alterar o resultado final do cálculo da função de distância, uma característica da métrica de *Levenshtein* e de suas derivações é que o menor valor do cálculo é zero e no máximo, em muitas línguas e dicionários reais, dificilmente representa um valor grande, ou seja, a capacidade de discriminação oferecida pela métrica tende a ser pequena.

Métricas de Minkowski

As métricas de *Minkowski* (Beyer et al, 1999), conhecidas como L_p, são empregadas em espaços multidimensionais. Deste modo, é possível utilizar distâncias geométricas da família L_p para comparar dados de dimensão fixa. Suponha-se que $s_1 = \{s_{1_1}, s_{1_2}, ..., s_{1_E}\}$ e $s_2 = \{s_{2_1}, s_{2_2}, ..., s_{2_E}\}$ sejam dois elementos de um espaço multidimensional de dimensão E. A família de distâncias L_p é definida como:

$$L_p(s_1, s_2) = (\sum_{i=1}^{E} |s_{1_i} - s_{2_i}|^p)^{1/p} \tag{2.1}$$

Os seguintes membros da família L_p são bem conhecidos:

- *City-Block*: $L_1(s_1, s_2) = \sum_{i=1}^{E} |s_{1_i} - s_{2_i}|$
- *Euclidiana*: $L_2(s_1, s_2) = (\sum_{i=1}^{E} |s_{1_i} - s_{2_i}|^2)^{1/2}$
- *Chebychev*: $L_\infty(s_1, s_2) = max_{i=1}^{E} |s_{1_i} - s_{2_i}|$

A métrica L_1, também conhecida como *City-Block*, corresponde ao somatório do módulo das diferenças entre as coordenadas. Nesse caso, o conjunto de pontos de mesma distância ξ a partir de um centro forma um losango — Figura 2.1(a). A métrica L_2, mais conhecida como distância euclidiana, corresponde à função usual para distância entre vetores. O conjunto de pontos a mesma distância de um centro segundo a distância

L_2 forma uma circunferência — Figura 2.1(b). Calculando-se o limite da equação 2.1 quando p tende ao infinito, obtém-se a métrica L_∞, também conhecida como *Chebychev* ou $L_{infinity}$, na qual o conjunto de pontos a distância de um centro forma um quadrado — Figura 2.1(c). A figura 2.1(d) apresenta as formas geométricas em relação aos pontos equidistantes à distância ξ a partir do elemento central s_q para esses três membros.

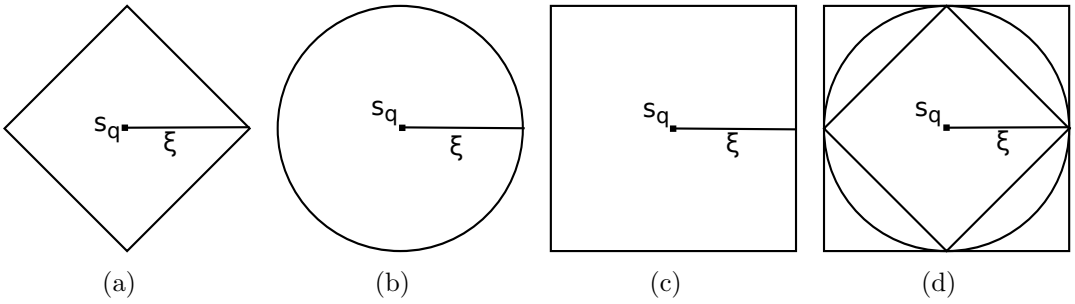


Figura 2.1: Representação das formas geométricas geradas para as métricas L_1, L_2 e L_∞ para os pontos equidistantes à distância ξ a partir do elemento central s_q

2.3 Consultas por similaridade

Os operadores de comparação por similaridade se aplicam a muitos tipos de dados complexos, desde que seja definida uma função de distância que respeite as propriedades de uma métrica. Por exemplo, sistemas de informações geográficas (SIGs) adotam por definição a função de distância euclidiana.

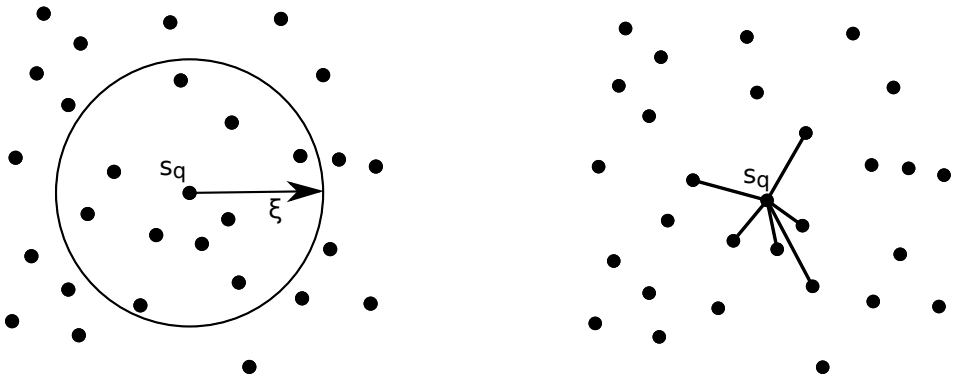
Assim, com a emergência do suporte de dados multimídia em SGBDs, operadores de consulta baseados em comparação por similaridade vêm despertando muito interesse. Um operador de consulta por similaridade recupera os elementos que atendem a um determinado critério de dissimilaridade, expresso com referência a um elemento do domínio de dados $s_q \in \mathbb{S}$, chamado *elemento central da consulta*. Em geral, são tratados pelo menos dois critérios de seleção por similaridade: os que expressam as consultas por abrangência e as consultas aos k vizinhos mais próximos.

2.3.1 Consulta por abrangência

Uma consulta por abrangência $R_q(s_q, \xi)$ recebe como parâmetros o elemento central da consulta s_q e um grau de dissimilaridade máximo ξ. Sua execução recupera todos os elementos da base de dados que diferem do elemento central da consulta por no máximo a dissimilaridade indicada por ξ. De modo formal, dado um sub-conjunto de dados $S = \{s_1, s_2, \cdots, s_n\}$ contudo em $\mathbb{S}$ e uma função de distância d, uma consulta por abrangência centrada em um elemento de consulta $s_q \in \mathbb{S}$ e com uma distância máxima ξ, retorna:

$$R_q(s_q, \xi) = \{s_i \mid s_i \in S, d(s_i, s_q) \leq \xi\}$$

A figura 2.2(a) mostra um exemplo de consulta por abrangência num espaço bidimensional com função de distância euclidiana. Outro exemplo de consulta por abrangência em uma base de dados de sequências genéticas é "*Selecione as sequências de DNA que difiram desta sequência s_q dada por até 5 bases purínicas*", representada como $R_q(s_q, 5)$.



(a) Consulta por abrangência com raio ξ

(b) Consulta aos k vizinhos mais próximos com $k = 6$

Figura 2.2: Exemplos de consultas por similaridade sobre o elemento s_q num espaço bidimensional euclidiano

2.3.2 Consulta aos k-vizinhos mais próximos

A consulta aos k-vizinhos mais próximos $kNN(s_q, k)$ recebe como parâmetros o elemento central da consulta s_q e uma quantidade k. Sua execução recupera os k elementos mais similares ao elemento central da consulta. Formalmente, dado um sub-conjunto de dados $S = \{s_1, s_2, \cdots, s_n\}$ contido em $\mathbb{S}$ e uma função de distância d, uma consulta aos k-vizinhos mais próximos centrado em um elemento de consulta $s_q \in \mathbb{S}$ e, com um número de objetos a ser retornado igual a k, o retorno da busca é dado por:

$$kNN(s_q, k) = A = \{s_j \mid s_j \in S, \forall s_i \in S - A : d(s_j, s_q) \ \leq \ d(s_j, s_i), \mid A \mid = k\}$$

Um exemplo de consulta aos k-vizinhos mais próximos em uma base de dados de sequências genéticas é "*Selecione as 3 proteínas mais semelhantes a esta proteína s_q dada*", representada por $kNN(s_q, 3)$. A figura 2.2(b) ilustra uma consulta aos k-vizinhos mais próximos em um domínio bidimensional empregando uma função de distância euclidiana.

2.3.3 Consulta aos k-vizinhos mais próximos reverso

As consultas por abrangência e por k vizinhos mais próximos são sem dúvidas as mais comuns e podem ser consideradas elementares para compor o repertório de consultas por similaridade. No entanto, outras consultas podem ser necessárias em determinadas situações e, embora possam ser definidas em função daquelas duas elementares, é importante reconhecer outras operações de seleção por similaridade, importantes em determinadas situações. Neste contexto, outra consulta por similaridade importante é a consulta aos k-vizinhos mais próximos reversos, representada como $RkNN(s_q, k)$. Ela recebe como parâmetros o elemento central da consulta s_q e uma quantidade k, e recupera os elementos do conjunto que têm o elemento central da consulta como um de seus k vizinhos mais próximos. Como exemplo desta consulta em uma base de dados de sequências genéticas, pode-se solicitar "*Selecione as proteínas que têm esta proteína s_q dada como uma de suas 3 mais semelhantes*", representada como $RkNN(s_q, 3)$.

Por ser a consulta abordada nesse trabalho, uma melhor descrição da consulta $RkNN$ é feita no capítulo 3, o qual também contém os trabalhos presentes na literatura sobre ela.

2.4 Métodos de acesso espaciais

É possível realizar consultas por similaridade em dados multidimensionais, e os métodos de acesso espaciais (MAEs) são estruturas projetadas para indexar dados multidimensionais permitindo uma execução eficiente das consultas por similaridade.

Em geral, um conjunto de elementos é dito multidimensional, ou E-dimensional, se todo elemento desse conjunto pode ser localizado por um conjunto de E coordenadas ($E > 1$). Os MAEs têm como premissa o fato de os dados manipulados pertencerem ao domínio dos dados multidimensionais e, portanto, podem se utilizar das propriedades geométricas para realizar buscas eficientes.

Uma classificação dos MAEs diferencia-os em 2 tipos: *método de acesso espacial para dados pontuais*, ou simplesmente *método de acesso espacial pontual* (MAEP), e *método de acesso espacial para dados não-pontuais*, ou simplesmente *método de acesso espacial não-pontual* (MAENP). Os MAEPs consideram que os dados são pontos num espaço, enquanto que os MAENPs consideram os dados como regiões, possuindo, portanto, extensão espacial (Gaede e Günther, 1998).

Uma avaliação completa dos MAEs pode ser encontrada em Gaede e Günther (1998) e em Böhm et al (2001). Esta dissertação se limita a apresentar o MAE *R-tree* (Guttman, 1984), por ser o mais referenciado e utilizado para indexação de dados multidimensionais, estando inclusive disponível para uso em diversos SGBDs de domínio públicos (como exemplo PostgreSQL) e comerciais (como exemplo Oracle). A *R-tree* é importante para

este trabalho também por sempre ser utilizada como base de comparação com outros MAEs, em vista de demonstrar e validar as melhorias propostas. Além disso, praticamente todos os algoritmos da consulta $RkNN$ propostos em MAEs são implementados na R-tree ou uma variante dela, como poderá ser observado no capítulo 3.

2.4.1 R-tree e algumas variantes

Considerada uma extensão da *B-tree* para elementos multidimensionais, a *R-tree* (Guttman, 1984) é um MAE para indexar dados multidimensionais não necessariamente pontuais e que se assume que todas as coordenadas são homogêneas (pertencem ao mesmo domínio e têm a mesma extensão de variação) e contínuas. Usualmente, considera-se que os dados estão no espaço $\mathbb{R}^E$. Essa árvore é balanceada e usa o conceito de retângulo envolvente mínimo (MBR) para definir regiões, de maneira que mesmo os elementos são representados na estrutura por MBRs.

Dado um espaço multidimensional $\mathbb{R}^E$, um MBR $R_i \subseteq \mathbb{R}^E$ é o hiper-retângulo ortogonal mínimo que envolve um conjunto de elementos. Toda superfície de um MBR R_i contém ao menos um elemento, sendo que para a *R-tree*, o particionamento realizado pelos MBRs é não completo e não disjunto. Logo, os MBRs permitem sobreposição, embora isto possa reduzir a eficiência da estrutura de índice.

Os nós na *R-tree* contém entre m e M elementos, com $m \leq M/2$, exceto para o nó raiz, o qual pode conter entre 2 e M elementos, sendo que M é a capacidade máxima de armazenamento em um nó, isto é o maior número de elementos que podem ser armazenados no nó. Nós internos contêm registros da forma (ptr_i, R_i), em que ptr_i é o ponteiro para um nó filho e R_i é o MBR que cobre todos os retângulos do nó filho. Os nós folhas contêm registros na forma (oid_i, R_i), em que oid_i é um ponteiro para o elemento e R_i é o MBR do elemento, isto é, cada nó folha armazena a representação do elemento em forma de um MBR.

A figura 2.3 (Guttman, 1984) contém um exemplo de uma *R-tree* num espaço bidimensional, apresentando sua estrutura e ilustrando os dados indexados. Para este exemplo, $M = 3$ e $m = 1$ e cada MBR é denotado por R_i, $1 \leq i \leq 19$.

Na *R-tree*, a inserção de novos dados é similar a inserção numa *B-tree*, isto é, novos registros de indexação são adicionados em nós folhas, os nós que sofrem *overflow* são particionados e as partições são propagadas para cima. Resumidamente, o algoritmo de inserção realiza as seguintes verificações: Se o novo registro está em exatamente um nó folha, então é inserido num registro deste nó. Se está contido em várias regiões, aquela com menor volume é selecionada. Se nenhuma região contém o dado, a região com menor aumento de volume é selecionada. Em caso de empate, o algoritmo seleciona a região com menor volume. Quando o nó sofre *overflow* depois da inserção, ele deve ser particionado em dois, assim novos MBRs devem ser criados e o pai dos nós devem ser atualizados. Se

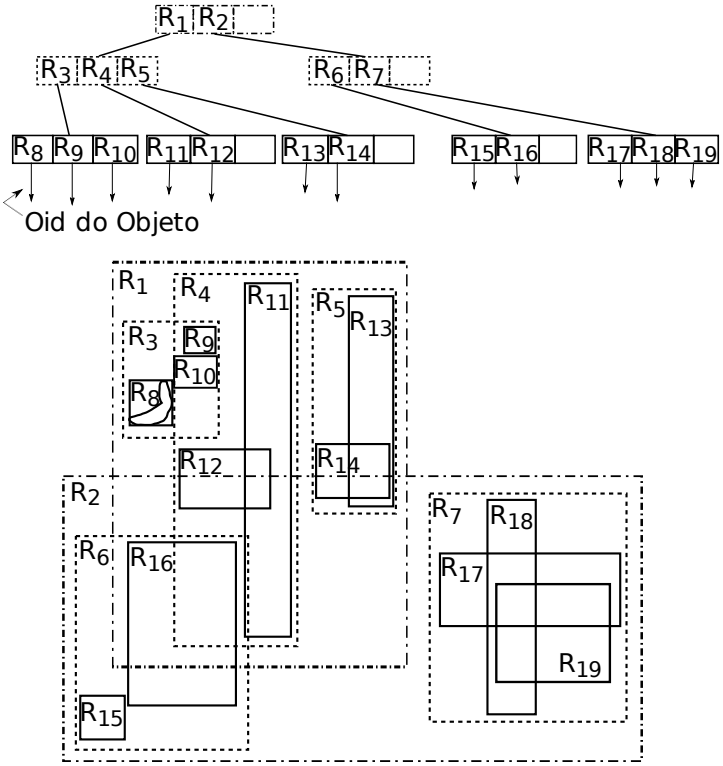


Figura 2.3: Exemplo de uma R-tree: representação estrutural e representação espacial com os MBRs (Guttman, 1984)

o pai sofre *overflow*, o processo deve ser aplicado recursivamente. No fim, se a raiz sofre *overflow*, ela é particionada e uma nova raiz é criada. Todo esse particionamento em caso de *overflow* ocorre baseado em 2 políticas: um algoritmo quadrático e um algoritmo linear. Guttman (1984), baseado em resultados experimentais, recomenda o uso do algoritmo linear.

Um extensão da *R-tree* destacada nesta dissertação é a R^**-tree* (Beckmann et al, 1990). Sua estrutura de dados básica é a mesma da *R-tree*, mas o algoritmo para particionamento de regiões e inserção de novos elementos são melhorados e é apresentado o conceito de *reinserção forçada*.

O algoritmo de inserção da R^**-tree* distingue nós folhas de nós internos. Quando nenhuma região contém o novo registro, o algoritmo diferencia os critérios de seleção dos nós: o critério para escolher um nó folha é, o que causa, pela ordem, o menor aumento da sobreposição de regiões, o menor aumento do volume coberto pelo nó e o menor volume do espaço coberto pelo nó. Já para nós internos, a região com menor aumento é a escolhida e em caso de empate o algoritmo seleciona a região com menor volume.

A R^**-tree* também contém uma nova política de particionamento que objetiva minimizar a sobreposição entre regiões e a cobertura de espaço vazio. Contudo, a árvore procura evitar o particionamento utilizando o conceito de *reinserção forçada* — reinserir o registro de um nó que sofre *overflow*. Numa reinserção, os elementos selecionados são aqueles mais distantes do centro da região. Esse processo de reinserção propicia dois resultados

benéficos: uma melhor utilização do espaço e uma estrutura reorganizada dinamicamente, evitando o particionamento dos nós e amenizando decisões ruins feitas no início, quando a árvore ainda tem poucos registros.

2.4.2 Algoritmos de consultas por similaridade em métodos de acesso espaciais

Nesta seção, descreve-se um algoritmo de consulta por abrangência canônico e dois algoritmos de consultas aos k-vizinhos mais próximos que se destacam para MAEs.

O algoritmo canônico para executar consultas por abrangência em estruturas de indexação hierárquica foi apresentado por Böhm et al (2001) e pode ser descrito do seguinte modo: iniciando da raiz, o algoritmo realiza um percurso em profundidade na árvore de indexação. Se o nó atual é um nó folha, verifica-se se os dados neste nó estão dentro da dissimilaridade permitida na consulta em relação ao elemento de consulta, retornando-o então como resultado. Se a página é um nó interno, o algoritmo continua a busca em todos os nós filhos dele caso a região desse nó intersecte com a área da consulta.

A implementação de consultas ao k-vizinhos mais próximos foi inicialmente proposta para resolver a especialização *consulta ao único vizinho mais próximo*, sendo depois generalizada para o caso de $k \geq 1$. A primeira abordagem de implementação, conhecida como algoritmo *RKV* (Roussopoulos et al, 1995), percorre a árvore em profundidade como o algoritmo de consulta por abrangência, mas neste caso não há um critério fixado de filtragem de subárvores. O critério ótimo seria a distância real do vizinho mais próximo, contudo este critério não é conhecido no começo da consulta. Por isso, o algoritmo utiliza estimativas das distâncias para filtrar as subárvores. Conforme o algoritmo percorre a árvore, mais informações são adquiridas e as estimativas podem ser melhoradas. Uma estimativa para a distância ao vizinho mais próximo é a menor distância entre um ponto já visitado e s_q, mas o algoritmo também realiza estimativas com relação às regiões visitadas até o momento.

As estimativas com relação às regiões já visitadas são calculadas utilizando alguns conceitos importantes: *MinDist*, *MaxDist* e *MinMaxDist*. *MinDist* é a distância mínima entre uma região e s_q. Por exemplo, se as regiões são esferas envolventes mínimas, então *MinDist* é igual a distância de s_q ao centroide menos o raio da esfera (a formula no caso de MBRs pode ser encontrada em Böhm et al (2001)). *MaxDist* é a distância máxima entre a região e s_q. Por fim, *MinMaxDist* é a distância máxima de s_q a uma região em que um ponto é garantido ser encontrado. Por exemplo, ao menos um ponto deve ser localizado na superfície de um MBR, assim o menor limitante máximo de uma distância de s_q a um ponto na região de *MaxDist* pode ser computado. A figura 2.4 ilustra estes 3 conceitos. Observa-se na figura que a quina superior esquerda é mais distante de s_q que a

quina inferior direita, assim a quina inferior direita é o ponto que determina *MinMaxDist* para este exemplo.

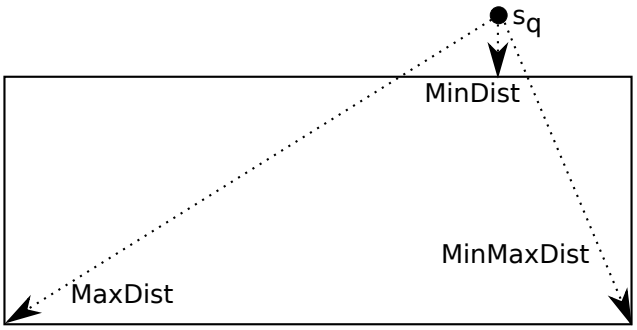


Figura 2.4: MinDist, MaxDist e MinMaxDist

O valor estimado utilizado no algoritmo RKV é o mínimo entre todas as distâncias computadas para os pontos e todos os valores de *MinMaxDist* das regiões processadas até o momento. Uma subárvore pode ser seguramente podada se seu *MinDist* é maior que o valor estimado que foi computado.

Para estender este algoritmo para consultas kNN, Roussopoulos et al (1995) propõem no descarte o uso de *MinMaxDist* em que o novo valor estimado é a k-ésima menor distância dentre todos as distâncias.

O segundo algoritmo de consulta ao vizinho mais próximo é conhecido como algoritmo *HS* (Hjaltason e Samet, 1999, 1995). Neste algoritmo, percorre-se o índice considerando que o melhor vem primeiro, isto é, as regiões são acessadas em ordem crescente de distância do limitante menor à s_q. Uma fila de prioridade é utilizada para armazenar os nós das árvores cujo pais já foram processados. A fila retorna sempre o nó com menor *MinDist* até s_q.

No início, a fila com prioridade contém apenas o nó raiz. Um nó N_i é retirado da fila e processado. Se N_i é um nó folha, todos os seus registros são examinados para ver se um deles é mais próximo de s_q que a distância do ponto mais próximo. Se este é o caso, o candidato a vizinho mais próximo e a distância do ponto mais próximo é atualizado. Se N_i é um nó interno, então *MinDist* é computado para todos seus filhos e eles são inseridos na fila. A busca pára quando o *MinDist* do primeiro nó na fila é maior que a distância ao vizinhos mais próximo atual, o que significa que nenhum outro ponto pode ser mais próximo do elemento de consulta que o atual candidato a vizinho mais próximo.

Para implementar consultas kNN, é necessária uma segunda fila com prioridade contendo a lista de candidatos. Como no caso do algoritmo RKV, o novo valor estimado é a k-ésima menor distância dentre todos as distância computadas para os pontos, o que é igual à distância máxima de s_q a um dos pontos armazenados na segunda fila.

2.5 Métodos de acesso métrico

Os métodos de acesso métricos (MAMs) são estruturas de dados especializadas para indexar dados em espaços métricos. Eles exploram propriedades métricas como a desigualdade triangular para possibilitar acesso eficiente aos dados. Descrições detalhadas dos MAMs são apresentadas por Chávez et al (2001), Hjaltason e Samet (2003) e Zezula et al (2006). Nesta seção é apresentado um breve resumo dos principais MAMs presentes na literatura.

2.5.1 Técnicas de Burkhard e Keller

A primeira proposta de indexação de dados complexos com base em distância entre pares de elementos foi proposta por Burkhard e Keller (1973). O trabalho apresenta três técnicas para indexar os dados, nas quais uma métrica discreta é assumida. O processo de indexação particiona recursivamente o conjunto de dados, criando uma árvore. As três técnicas são apresentadas a seguir:

- Na primeira técnica, cada nó possui uma entrada. Assim, um elemento qualquer do conjunto é selecionado e definido como raiz da árvore em um nó. Todo elemento restante no conjunto é dividido em subconjuntos, sendo cada subconjunto um grupo S_i de elementos distanciados em i unidades da raiz. Estes subconjuntos são recursivamente particionados até que reste somente um elemento no subconjunto.

- Na segunda técnica, o conjunto de dados é particionado em um número fixo de subconjuntos e, para cada um, são definidos um elemento representante e um raio máximo. Os demais elementos são distribuídos de forma que sua distância ao representante seja menor ou igual ao respectivo raio. Esse procedimento é repetido recursivamente e os representantes são agrupados em nós.

- A terceira técnica é similar à segunda, com o requisito adicional que a distância máxima entre quaisquer dois elementos em um mesmo conjunto não seja maior que uma dada constante c, cujo valor é diferente em cada nível da estrutura. Os conjuntos satisfazendo esse critério são chamados de *clique*. A escolha do valor de c tem que garantir que todos os elementos do espaço estejam em pelo menos um dos *cliques*, podendo um elemento aparecer em mais que um *clique*. Em seguida, um elemento arbitrário de cada *clique* é escolhido como seu representante.

Os representantes selecionados nas três técnicas são usados para podar elementos e subárvores durante uma consulta.

A consulta por abrangência $R_q(s_q, \xi)$ inicia da raiz e percorre toda a árvore recursivamente comparando o elemento representante do nó s_c, também conhecido como *pivô*, com o elemento de consulta s_q. Se a distância $d(s_c, s_q) < \xi$, o elemento s_c é retornado.

Além disso, recursivamente, o algoritmo acessa todos os nós filhos s_i caso valha a seguinte desigualdade:

$$max\{d(s_c, s_q) - \xi, 0\} \leq i \leq d(s_c, s_q) + \xi$$

2.5.2 Fixed queries tree

A *Fixed Queries Tree* (FQ-tree) foi proposta por Baeza-Yates et al (1994) como uma modificação das técnicas de Burkhard e Keller (1973). A modificação consiste em utilizar um único elemento representante para todos os nós do mesmo nível. Todos os elementos na FQ-tree são armazenados em nós folhas e os nós internos são utilizados para navegação durante uma busca ou inserção. O algoritmo de consulta R_q é o mesmo apresentado anteriormente para as técnicas de Burkhard e Keller. Entretanto, a FQ-tree têm como vantagem a redução no número de computação de distâncias devido ao fato que se mais de uma subárvore tiver que ser acessada, apenas uma distância deve ser computada entre o elemento de consulta e o representante daquele nível.

Uma variante, conhecida como *Fixed-Height Fixed Queries Tree* (FHQT), foi proposta com uma modificação na estrutura da FQ-tree. Essa modificação define que todos os nós folhas estejam em uma mesmo nível, isto é, caminhos menores na FQ-tree são expandidos para que todos os nós folhas fiquem numa altura h. Essa abordagem melhora o desempenho de consultas, pois o processo da consulta pode parar antes de alcançar o nó folha, não implicando em custos extras (Baeza-Yates et al, 1994; Baeza-Yates, 1997).

2.5.3 Vantage point tree

Os MAMs explicados até aqui foram projetados com base em métricas discretas. A árvore denominada *Vantage Point Tree* (VP-tree), proposta por Yianilos (1993), foi o primeiro MAM projetado para métricas contínuas, apesar de atender também as métricas discretas.

A VP-tree divide o conjunto S através do princípio de *particionamentos por bolas*, em que o conjunto S é dividido em subconjuntos S_1 e S_2 baseado em algum elemento $s_r \in S$, chamado de ponto de vantagem (elemento representante), e o raio ξ_r, definido como a mediana das distâncias de cada elemento ao ponto de vantagem. A definição do ponto de vantagem e do raio do particionamento é escolhido por um algoritmo que considera todos os elementos da árvore, para que ela seja balanceada. Por essa razão, a árvore é estática. Assim, o conjunto é dividido em cada subconjunto, segundo o seguinte critério:

$$S_1 = \{s_i \in S \mid d(s_i, s_r) \leq \xi_r\}$$

$$S_2 = \{s_i \in S \mid d(s_i, s_r) > \xi_r\}$$

O mesmo princípio deve ser aplicado em ambos os subconjuntos, formando uma hierarquia, como pode ser visto na figura 2.5.

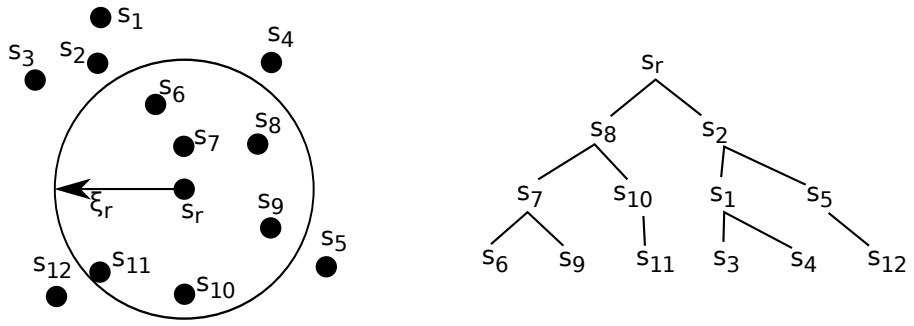


Figura 2.5: Exemplo da VP-tree com raiz s_r: representação bidimensional e a correspondente árvore

O algoritmo R_q percorre a VP-tree da raiz para as folhas. Durante o percurso, o algoritmo avalia o nó interno com relação à distância entre o ponto de vantagem e o elemento de consulta $d(s_{r_1}, s_q)$. Se essa distância é igual ou inferior ao raio de consulta, então o ponto de vantagem s_r deve ser retornado. Além disso, o algoritmo deve verificar quais subárvores ele deve acessar. Em resumo, ele deve acessar a subárvore esquerda (S_1) se $max\{d(s_q, s_r) - \xi_r, 0\} \leq \xi_q$ e deve acessar a subárvore direita (S_2), se $max\{\xi_r - d(s_q, s_r), 0\} \leq \xi_q$ (Zezula et al, 2006).

Uma variante da VP-tree proposta no mesmo artigo é a *VP^s-tree* . A ideia é aproveitar as distâncias calculadas durante a construção da árvore para produzir algoritmos de buscas mais eficientes. As distâncias calculadas durante a inserção de elementos na árvore é armazenada na VP^s-tree e então utilizada nas consultas R_q, como a seguir:

- Se $\mid d(s_q, s_r) - d(s_r, s_i) \mid > \xi_q$, o elemento s_i pode ser descartado sem a necessidade de calcular a distância $d(s_q, s_i)$
- Se $d(s_q, s_r) + d(s_r, s_i) \leq \xi_q$, o elemento s_i deve ser incluído no conjunto de respostas da consulta.

Outra variante da VP-tree é a *VP^{sb}-tree* (Yianilos, 1993), uma extensão da VP^s-tree em que os nós folhas formam *buckets*. Nessa extensão, são armazenados mais elementos num nó folha, propiciando economia de espaço e resultando em menos acessos a disco para as consultas.

2.5.4 Generalized hyperplane tree

Em Uhlmann (1991), é proposta a *Generalized Hyperplane Tree* (GH-tree) que usa o princípio de particionamento por hiperplano generalizado para dividir o espaço. Cada particionamento divide o conjunto S em dois subconjunto S_1 e S_2, de forma que:

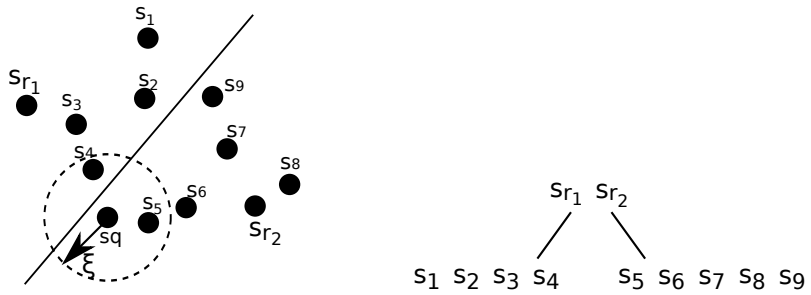
$$S_1 = \{s_i \in S \mid d(s_{r_1}, s_i) \leq d(s_{r_2}, s_i)\}$$

$$S_2 = \{s_i \in S \mid d(s_{r_1}, s_i) > d(s_{r_2}, s_i)\}$$

sendo s_{r_1} e s_{r_2} elementos do conjunto S escolhidos como representantes.

A GH-tree é uma árvore binária construída da seguinte maneira: dois elementos representantes s_{r_1} e s_{r_2} são selecionados do conjunto de dados e aplica-se o particionamento. Os elementos mais próximos a s_{r_1} formam a subárvore esquerda e os elementos mais próximos a s_{r_2} formam a subárvore direita. A seguir, o processo é recursivamente aplicado em ambos os subespaços. Salienta-se que a escolha dos representantes considera todos os elementos da base para que a árvore fique balanceada. Na figura 2.6 apresenta-se um exemplo de uma GH-tree: na figura 2.6(a) é indicado o particionamento e é especificada uma consulta R_q. A figura 2.6(b) apresenta a estrutura da árvore correspondente particionada no primeiro nível.

Num processo de busca para uma R_q, a subárvore esquerda é acessada caso $d(s_{r_1}, s_q) - \xi \leq d(s_{r_2}, s_q) + \xi$, enquanto que a subárvore direita é acessada caso $d(s_{r_1}, s_q) + \xi \geq d(s_{r_2}, s_q) - \xi$.



(a) Uma consulta por abrangência que acessa ambas as partições

(b) Estrutura correspondente ao particionamento no primeiro nível da árvore

Figura 2.6: Exemplo do primeiro nível de uma *Generalized Hyperplane Tree*

2.5.5 AESA

O método *Approximating Eliminating Search Algorithm* (AESA), proposto por Vidal (1986), consiste em construir uma matriz de distâncias entre elementos do conjunto S durante a criação da estrutura. Devido às propriedades de simetria e identidade, o número de cálculos necessários para a criação dessa matriz é $n(n-1)/2$. Nessa estrutura, todos os elementos são representantes e podem ser utilizados numa operação de busca.

Uma operação de $R_q(s_q, \xi)$ escolhe um elemento s_r como representante, calculando a distância entre o elemento de consulta e o representante para utilizar no descarte de elementos. Um elemento s_i pode ser descartado segundo a condição $\mid d(s_r, s_q) - d(s_r, s_i) \mid > \xi$ que utiliza as distâncias já computadas. O algoritmo escolhe outro representante dentre os elementos restantes, tentando maximizar o efeito de descarte através da escolha de um representante que maximize $\mid d(s_r, s_q) - d(s_r, s_i) \mid$. Assim, o processo é recursivamente aplicado até os elementos restantes não poderem ser mais descartados, passando então por uma verificação se $d(s_q, s_i) \leq \xi$.

Embora o tempo experimental da consulta seja $O(1)$, essa proposta tem tempo de construção e complexidade de espaço igual $O(n^2)$, inviabilizando seu uso para grandes conjuntos de dados.

Uma nova versão da AESA, proposta por Micó et al (1994), é a *Linear AESA* (LAESA). Essa nova versão fixa o número de representantes a j elementos. Assim, a complexidade de tempo de construção e espaço é $O(jn)$. A consulta é semelhante à proposta da AESA, diferindo apenas no fato que há apenas j representantes.

2.5.6 Geometric near-neighbor access tree

A *Geometric Near-neighbor Access Tree* (GNAT) foi proposta por Brin (1995) como uma extensão da GH-tree. Essa extensão consiste na escolha de j representantes em cada nó ao invés de apenas dois. A inserção de elementos é semelhante a GH-tree: compara-se a distância entre a elementos e os representantes e insere-se os elementos na subárvore do representante mais próximo. Além disso, para cada representante são guardadas as distâncias mínima e máxima aos outros representantes, permitindo podas adicionais durante uma busca e resultando num algoritmo de R_q um pouco diferente do usado na GH-tree.

Brin (1995) realiza a comparação entre essa árvore e a VP-tree. Nesta comparação, é verificado que o tempo de construção da GNAT é maior que da VP-tree, entretanto os algoritmos de consulta fazem menos cálculos de distâncias. A figura 2.7 ilustra o primeiro nível de uma GNAT com 3 representantes: s_{r_1}, s_{r_2} e s_{r_3}.

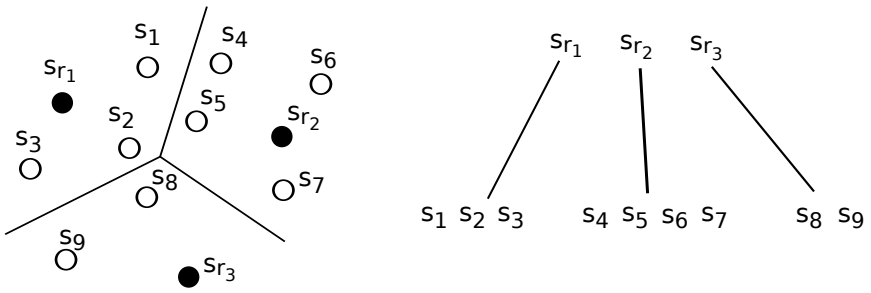


Figura 2.7: Exemplo do primeiro nível de uma *Geometric Near-neighbor Access Tree*

2.5.7 M-tree

Todos os MAMs descritos anteriormente são estáticos, ou seja, exigem que todos os elementos a serem indexados estejam disponíveis durante o processo de criação das árvores, limitando a sua aplicação efetiva em SGBDs. Além disso, a construção da árvore é realizada em um processo recursivo baseado na estratégia *top-down* (do nível raiz em direção as folhas).

Ciaccia et al (1997) propuseram o primeiro MAM dinâmico, a M-tree. Visto como uma adaptação do MAE R-tree para indexar dados em espaços métricos, a M-tree é construída

com uma estratégia *bottom-up*, de forma a garantir o balanceamento da estrutura e tendo os elementos armazenados em nós de tamanho fixo. Além disso, este MAM permite a inserção de elementos a qualquer momento e não exige reestruturações periódicas.

Na M-tree, os nós folhas contêm todos os elementos indexados, enquanto os nós internos replicam os elementos escolhidos como representantes. Cada entrada do nó folha N_i contém o identificador do elemento s_i, informações do elemento que permita o cálculo de distância (em geral o próprio elemento) e a distância ao elemento representante $d_{s_i s_{r_p}}$, o qual está armazenado também no nó pai. Cada entrada do nó interno N_j contém uma ligação para sua subárvore (nó filho), um elemento representante s_{r_i}, a distância do representante ao pai do representante $d_{s_{r_i} s_{r_p}}$ e o raio ξ_{r_i} da região que acomoda todos os elementos indexados pela entrada (chamada de raio de cobertura).

Construção da árvore

O algoritmo de *inserção* da M-tree percorre a árvore recursivamente para localizar o nó folha mais apropriado para inserir o novo elemento. Contudo, pode ser executado o particionamento do nó caso este esteja cheio. O princípio básico de escolha do nó apropriado consiste em descer nas subárvores que têm raio de cobertura suficiente para que o elemento seja inserido sem aumentar o raio. Se há múltiplas subárvores com esta propriedade, a com maior proximidade entre o novo elemento e o elemento representante é escolhida. Entretanto, se nenhuma subárvore satisfaz esta propriedade, é escolhida a subárvore que minimize o aumento do raio de cobertura para abrigar o novo elemento.

Quando a inserção ocorre em um nó cheio, há a necessidade de particionar o nó. O particionamento se dá alocando um novo nó no mesmo nível do nó particionado, distribuindo as entradas do nó original entre os dois nós segundo um critério pré-definido e promovendo dois elementos representantes, um de cada novo nó, para o nó pai. Salienta-se que quando a raiz é dividida, uma nova raiz é criada e a árvore aumenta de nível.

São propostos cinco algoritmos de seleção dos representantes e particionamento do nó:

m_RAD: Em termos de computação de distâncias, esse algoritmo é o mais complexo deles. Ele considera todos os pares de elementos, particionando o conjunto de entradas e promovendo o par de elementos que tiver a menor soma dos raios de cobertura.

mM_RAD: Similar a m_RAD, o par de elementos representante escolhido é aquele que o máximo do raio de cobertura gerados ($max(\xi_{r1}, \xi_{r2})$) seja mínimo.

M_LB_DIST: Esse algoritmo difere do anterior pela utilização das distâncias armazenadas. O primeiro elemento representante escolhido é o representante do nó que esta sendo particionado e o outro elemento escolhido é o mais distante ao primeiro representante selecionado.

RANDOM: Esse algoritmo seleciona de forma aleatória os elementos representantes.

SAMPLING: Essa algoritmo é como o RANDOM, porém interage sobre uma amostra de elementos.

Dentre todos esses algoritmos, avaliações experimentais apontaram que a estratégia mM_RAD, também conhecida como *minMax*, foi no geral a mais eficiente (Ciaccia e Patella, 1998).

Consultas por abrangência e por vizinhos mais próximos

O algoritmo da consulta $R_q(s_q, \xi)$ realiza um percurso em profundidade, iniciando da raiz. Essa busca utiliza as distâncias armazenadas para realizar a poda seguindo alguns critérios, que são apresentados a seguir:

Para os nós internos, considere-se s_r o representante de uma subárvore que tem raio de cobertura ξ_r e elemento pai s_{r_p}. A subárvore pode ser podada se $\mid d(s_q, s_{r_p}) - d(s_r, s_{r_p}) \mid -\xi_{r_i} > \xi$. Contudo, se $\mid d(s_q, s_{r_p}) - d(s_r, s_{r_p}) \mid -\xi_{r_i} \leq \xi$, há a necessidade de calcular $d(s_q, s_r)$ para verificar se a subárvore pode ser podada pelo critério $d(s_q, s_r) - \xi_{r_i} > \xi$. Subárvores não podadas devem ser percorridas.

Para os nós folhas, o processo é similar. Considere s_i um elemento no nó folha e s_{i_p} elemento pai. A condição de descarte do elemento é $\mid d(s_q, s_{i_p}) - d(s_i, s_{i_p}) \mid > \xi$. Caso esse critério não seja satisfeito, a distância entre esse elemento e o elemento de consulta é avaliada e se $d(s_q, s_i) < \xi$, o elemento s_i é retornado como resultado.

Para consultas kNN, o algoritmo proposto usa uma técnica *branch-and-bound* similar ao algoritmo RKV (seção 2.4.2) desenvolvido para a R-tree e baseia-se no algoritmo de consulta por abrangência, mas com o raio de consulta definido como a distância ao k-ésimo atual vizinho mais próximo (Ciaccia et al, 1997).

2.5.8 Slim-tree

A *Slim-tree* (Traina Júnior et al, 2000, 1999) é uma evolução da M-tree. Portanto, ela também é uma árvore dinâmica e balanceada com uma estratégia de construção *bottom-up* (das folhas em direção a raiz). Os elementos são armazenados nas folhas e as entradas dos nós internos definem a hierarquia de organização dos dados. Os nós internos e as folhas da *Slim-tree* armazenam basicamente as mesmas informações já descritas para a M-tree, mas para cada entrada de nó interno adiciona-se a quantidade de elementos na subárvore $N_j.numElem$.

A Slim-tree é o primeiro MAM dinâmico que possui recursos para a avaliação do grau de sobreposição entre seus nós (*Fat-Factor*) e um algoritmo para minimizar o problema de sobreposição, denominado *Slim-down*. Uma descrição detalhada sobre o *Fat-factor* e o *Slim-down* pode ser encontrado em Traina Júnior et al (2000) e em Traina Júnior et al (2002a).

Outro diferencial da *Slim-tree* com relação a M-tree é um novo algoritmo de particionamento de nós (MST), que é descrito a seguir.

Construção da árvore

O algoritmo de inserção especifica que os elementos devem ser inseridos seguindo o seguinte procedimento: iniciando da raiz, ele percorre a estrutura na tentativa de localizar um nó qualificado a armazenar o novo elemento sem a necessidade de aumentar o raio de cobertura. Caso nenhum nó se qualifique, é selecionado o nó cujo centro está mais próximo do novo elemento. Se mais de um nó se qualificar, é executado um algoritmo denominado *ChooseSubtree* para selecionar qual deles armazenará o novo elemento. Este processo é aplicado recursivamente para todos os níveis da árvore até alcançar um nó folha (Traina Júnior et al, 2002a).

São propostas três opções para o algoritmo *ChooseSubtree*:

- **Aleatório** (*random*): escolhe aleatoriamente um dos nós qualificados a armazenar o novo elemento sem aumento do raio de cobertura. Trata-se da opção mais simples e de menor custo;

- **Distância mínima** (*minDist*): escolhe o nó que minimiza a distância entre o novo elemento e seu respectivo representante;

- **Ocupação mínima** (*minOccup*): escolhe o nó que possui a menor taxa de ocupação. Trata-se da opção padrão de escolha para a construção de uma *Slim-tree*.

Quando a inserção ocorre em um nó cheio, é necessário particioná-lo. Esse particionamento se dá da mesma forma que na M-tree, porém é apresentado um novo algoritmo de seleção dos representantes e particionamento do nó, o *MST*. Esse algoritmo cria uma árvore de caminho mínimo (*minimal spanning tree*) contendo os elementos do nó a ser particionado. Considerando G um grafo com C vértices (elementos a serem divididos) e $C(C-1)$ arestas, em que o peso de cada aresta corresponde à distância entre vértices que ela conecta, o algoritmo executa quatro passos básicos:

1. construir a MST de G, isto é, definir o grafo que representa o menor caminho conectando todos os vértices;

2. remover a aresta mais longa;

3. colocar os dois grupos de elementos conectados nos novos nós;

4. para cada nó, escolher como representante o elemento mais próximo a todos os outros, ou seja, o elemento cuja distância máxima em relação aos demais seja a menor.

A figura 2.8 ilustra o processo de particionamento de um nó composto por elementos em um espaço euclidiano bidimensional. Como o algoritmo básico não garante uma porcentagem mínima de elementos em cada nó, uma variação tenta encontrar, sempre que possível, a aresta mais apropriada para gerar uma distribuição equilibrada.

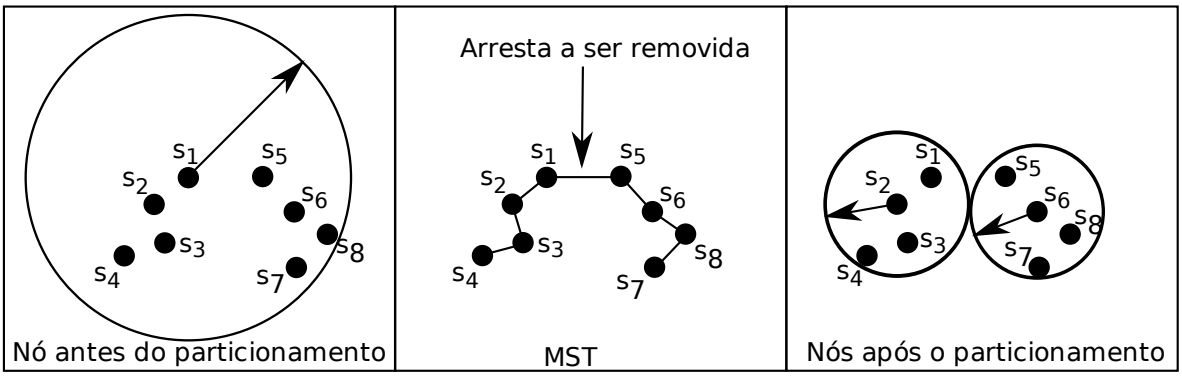


Figura 2.8: Exemplo de particionamento de nó usando MST (Traina Júnior et al, 2000)

Consultas por similaridade

Os algoritmos de consultas por similaridade implementados na *Slim-tree* são o de consulta por abrangência e o de consulta por k vizinhos mais próximos presentes na *M-tree*.

2.5.9 Família OMNI

Em Santos Filho et al (2001), foi proposta a utilização de múltiplos elementos representantes, denominados *focos*, que propiciam a diminuição de cálculos de distância entre os elementos na realização de consultas por similaridade. A ideia deste método é selecionar, em um conjunto de dados $S \subseteq \mathbb{S}$, um subconjunto de elementos (os *focos*) e torna-los referências globais a todo elemento do conjunto de dados S. Assim, os *focos* podem ser utilizados para mensurar a distância em qualquer região do espaço atuando também como geradores de coordenadas espaciais para os elementos.

O conceito OMNI pode ser aplicado junto a qualquer método de acesso existente, seja métrico, espacial ou mesmo sequencial, pois as coordenadas geradas pelos *Omni-focos* podem ser indexadas por qualquer um deles. Portanto, o conceito OMNI acoplado aos MAs já existentes, gera uma nova *família* de MAs, denominada *Omni-family*. A Figura 2.9 ilustra o uso de dois *focos*, onde se pode notar que a utilização deles permite descartar partes do espaço. Uma consulta por abrangência realizada a partir do elemento de consulta s_q, de raio ξ, pode descartar qualquer elemento que não esteja na região de intersecção dos arcos dos dois *focos*, restando poucos falsos positivos a serem avaliados.

Utilizando o conceito OMNI sobre a *Slim-tree*, Traina Júnior et al (2002b) propuseram um novo MAM chamado *DF-tree*. Assim, para uma *DF-tree*, além dos representantes

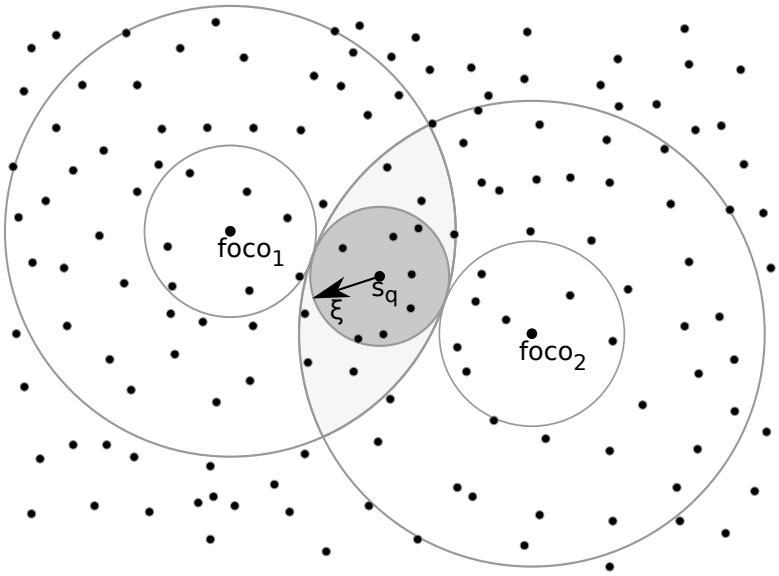


Figura 2.9: Exemplo de utilização do conceito OMNI com dois focos

referentes a cada nó, há também o descarte propiciado pelo uso de representantes globais, diminuindo consideravelmente o número de cálculos de distância requeridos para as consultas, pois permite uma taxa maior de podas de subárvores.

No MAM *DF-tree*, as consultas por similaridade utilizam algoritmos adaptados da *Slim-tree* que avaliam o elemento representante de cada nó e também os representantes globais (*focos*) para realizar a poda de subárvores. Assim, pode-se dizer que a OMINI-family é composta pela aplicação da técnica de mapeamento seguida pelo uso de um método de acesso, gerando por exemplo os métodos OMNI-Sequencial, OMNI-R-tree, OMNI-Slim-tree, etc. Já a DF-tree é um caso de aplicação invertida, especifica para a Slim-tree, gerando o que seria equivalente à Slim-tree-OMNI.

2.6 Imersão de espaços métricos e sua aplicação em MAMs

A imersão de espaços métricos é o último conceito importante a ser apresentado neste capítulo. Especificamente, este trabalho assume que o espaço métrico com n elementos é imerso isometricamente em um espaço euclidiano com a finalidade de utilizar-se de lei dos cossenos e pontos de tangência para realizar podas de subárvores.

Utilizando o teorema 2.1 (Indyk e Matousek, 2004; Deza e Laurent, 1997), pode-se verificar se um espaço métrico pode ser imerso isometricamente em um espaço euclidiano e, assim, utilizar-se de propriedades geométricas euclidianas para realizar podas em espaços métricos.

Teorema 2.1 *Um espaço métrico* $\mathbb{M}$ *com* n *elementos é isometricamente imersível em um espaço eucliadiano se, e somente se a matriz* $n \times n$

$$\left(d(s_0, s_i)^2 + d(s_0, s_j)^2 - d(s_i, s_j)^2\right)_{i,j=1}^{n}$$

é uma matriz semi-definida positiva (Indyk e Matousek, 2004; Deza e Laurent, 1997).

Embora não se possa verificar todos os espaços métricos para avaliar se são imersíveis isometricamente em um espaço eucliadiano, os mais comumente utilizados em problemas de busca são espaços métricos semi-definidos positivos e, portanto, imersíveis isometricamente em um espaço eucliadiano(Ban e Kadobayashi, 2008). Além disso, Ban e Kadobayashi (2008) descreve como estender essa propriedade de imersão isométrica para espaços métricos genéricos e assim poder aplicar propriedades geométricas em MAMs.

Uma aplicação interessante de imersão isométrica em MAMs foi proposta por Ban e Kadobayashi (2008), onde os autores usam lei dos cossenos para introduzir um novo grupo de estratégias de poda para consultas kNN no MAM AESA. Essas regras de podas são comprovadamente mais eficientes que as regras baseadas em desigualdade triangular e requerem que o espaço métrico seja um espaço semi-definido positivo.

2.7 Considerações finais

O presente capítulo apresentou as consultas por similaridade mais importantes e os principais MAMs existentes na literatura, os quais têm como características inerente o suporte às consultas por similaridade.

Com relação aos MAMs, pode-se observar que visam diminuir o número de cálculos de distância e o número de acessos a disco, tentando obter estruturas mais eficientes para processar consultas por similaridade. Além disso, observa-se que as consultas R_q e kNN são implementadas nos MAMs de várias maneiras e utilizam-se dos representantes para realizar podas eficientes.

Certamente uma das consultas por similaridade que tem aumentado em importância nos últimos anos é a consulta aos k-vizinhos mais próximos reversos. Ela foi brevemente explicada neste capítulo e será aprofundada no capítulo seguinte por se tratar da consulta de interesse deste trabalho.

Capítulo

3

k-vizinhos mais próximos reversos

3.1 Considerações iniciais

A consulta aos k-vizinhos mais próximos reversos $RkNN(s_q, k)$ recebe como parâmetros o elemento central da consulta s_q e uma quantidade k, e recupera do conjunto de dados os elementos que têm s_q como um de seus k vizinhos mais próximos. Várias abordagens para implementação desta busca foram desenvolvidas e apresentar as mais significativas é a finalidade desse capítulo. Assim, este capítulo contém os conceitos que envolvem a consulta aos k-vizinhos mais próximos reversos e os trabalhos desenvolvidos sobre este tipo de consulta.

3.2 Conceitos sobre consultas $RkNN$

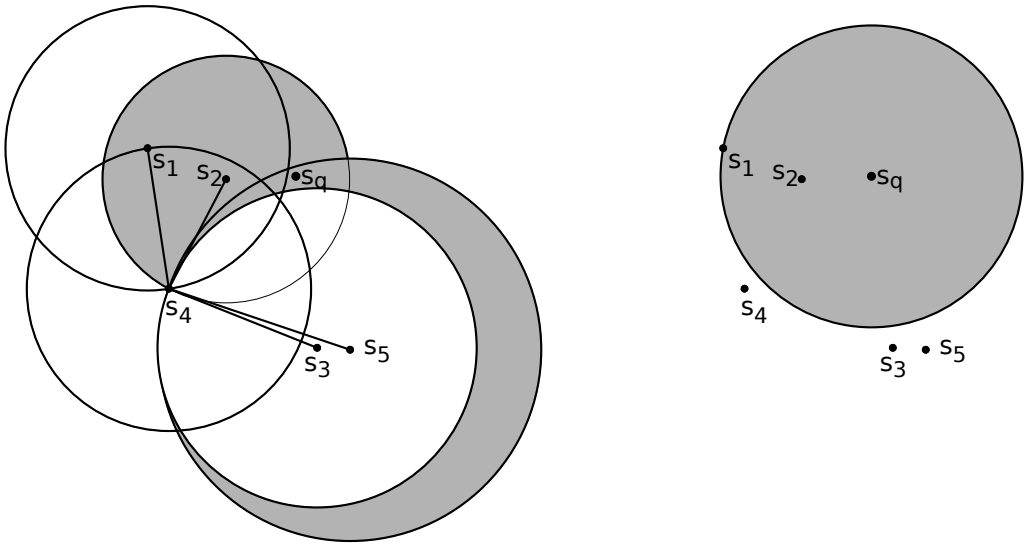
O operador de consulta por similaridade aos k-vizinhos mais próximos reversos foi originalmente descrito por Korn e Muthukrishnan (2000) considerando o caso onde $k = 1$, isto é, considerando o vizinho mais próximo reverso (RNN). Chamado RNN, este operador é capaz de descrever a região de influência de um determinado elemento de consulta sobre um conjunto de dados, isto é, ele recupera a região ao redor do elemento s_q que tem s_q como seu vizinho mais influente ou mais similar.

Korn e Muthukrishnan (2000) também descrevem o operador $RkNN$ como uma noção estendida de região de influência definida para o operador RNN. Esse operador recupera todos os elementos ao redor de s_q que tem s_q como um de seus k-vizinhos mais próximos. Formalizando, pode-se expressar o operador da seguinte maneira:

$$RkNN(s_q, k) = A = \{s_i \in S, s_q \in kNN(s_i, k) \wedge \forall s_j \in S - A : s_q \notin kNN(s_j, k)\}$$

As figuras 3.1(a) e 3.1(b) contêm um mesmo conjunto de dados com 5 elementos $s_1, s_2, ..., s_5$ e um elemento de consulta s_q. Na figura 3.1(a), cada elemento do conjunto de dados está associado a um círculo de cobertura de seus 2 vizinhos mais próximos — por exemplo, o elemento s_2 têm um círculo centrado nele que cobre os pontos s_1 e s_4. Uma consulta $RkNN(s_q, 2)$ sobre o elemento s_q inclui os elementos do conjunto de dados cujos círculos de cobertura contêm s_q — no caso da figura, os elementos retornados são s_2 e s_5. Na figura 3.1(b), a consulta $kNN(s_q, 2)$ retorna os 2 vizinhos mais próximos de s_q que são s_1 e s_2.

O exemplo da figura 3.1 permite verificar que um elemento $s_i \in kNN(s_q, k)$ não necessariamente pertence a $RkNN(s_q, k)$, isto é, o relacionamento entre $RkNN$ e kNN é assimétrico.



(a) Consulta $RkNN$ com $k = 2$ (b) Consulta kNN com $k = 2$

Figura 3.1: $kNN(s_q, 2)$ e $RkNN(s_q, 2)$ exemplificando que o relacionamento entre $RkNN$ e kNN é assimétrico

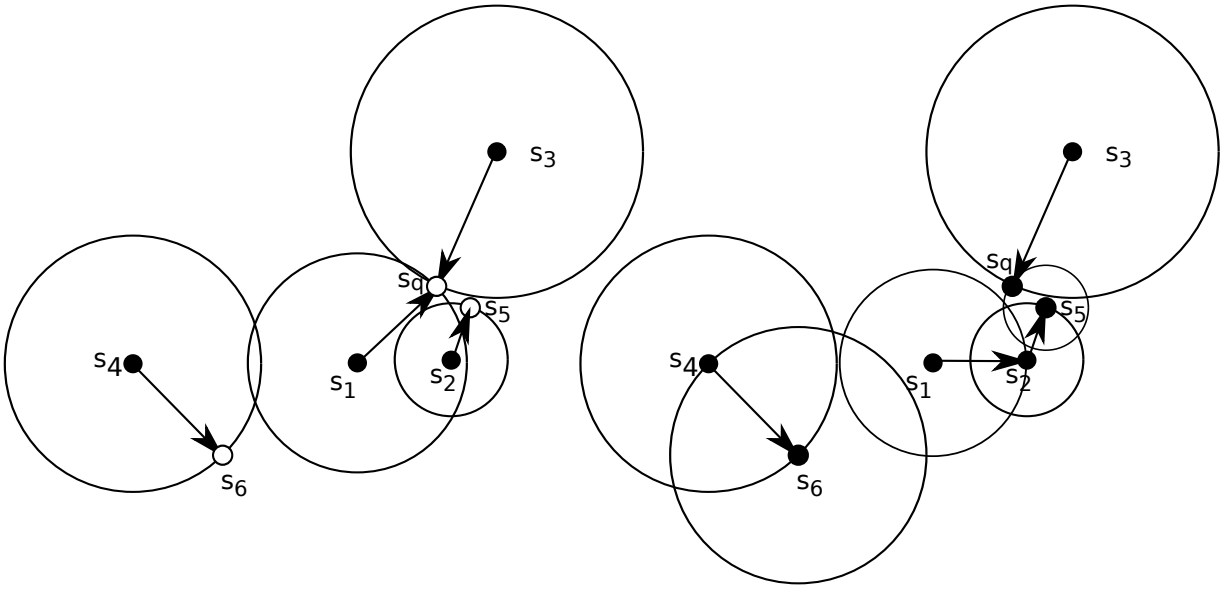
3.2.1 Variantes

Duas variantes para o cenário básico em uma consulta $RkNN$ ganharam destaque:

- A primeira variante é que o operador $RkNN$ pode ser monocromático ou bicromático. Essa distinção ocorre pela quantidade de tipos distintos de pontos que existem em um conjunto de dados para efeito de avaliação da consulta. No caso da consulta monocromática aos vizinhos mais próximos reversos, todos os pontos da base de

dados são considerados como da mesma categoria. Já uma consulta bicromática aos k-vizinhos mais próximos reversos assume que os pontos armazenados são considerados como parte de categorias de pontos, como exemplo, uma categoria de clientes e uma categoria de servidores. Em uma consulta bicromática aos $RkNN$, o retorno para um elemento de consulta s_q de uma categoria é todos os elementos da outra categoria que têm s_q como um de seus k-vizinhos mais próximo na categoria do elemento de consulta.

De forma simplificada, a figura 3.2 exemplifica uma consulta $RkNN$ bicromática e uma consulta $RkNN$ monocromática com $k = 1$. As setas representam o raio de cobertura da circunferência em que se encontra o vizinho mais próximo do elemento central. A figura 3.2(a) exemplifica uma consulta $RkNN$ bicromática com $k = 1$. Nesta consulta, os clientes s_1 e s_3 são os elementos retornados pela consulta $RkNN$ sobre o servidor s_q. A Figura 3.2(b) exemplifica uma consulta $RkNN$ monocromática com $k = 1$, a qual retorna os elementos s_3 e s_5 como os vizinhos mais próximos reversos de s_q.



(a) Bicromático (b) Monocromático

Figura 3.2: Exemplo de $RkNN$ bicromático e monocromático para $k = 1$

- Outra variante do cenário básico de uma consulta $RkNN$ é se o conjunto de dados é estático ou dinâmico. O caso dinâmico é quando deseja-se inserir ou remover pontos em um conjunto de dados e ainda assim suportar a consulta $RkNN$. No caso estático, o conjunto de dados é inserido na criação do MA e depois nunca é modificado. Quando o conjunto de dados é dinâmico, sempre que um novo elemento é inserido, ele pode mudar quais elementos são os vizinhos mais próximos reversos de

vários outros, assim a dinamicidade dos dados pode causar impacto em algoritmos que realizem pré-calculo de estruturas de dados para agilizar a resposta às consultas.

3.3 Algoritmos RNN euclidianos com pré-computação

Por definição, uma consulta $RkNN$ tem solução quadrática: para cada elemento $s_i \in S$, calcula-se seus k vizinhos mais próximos e verifica-se se o elemento de consulta está entre eles. Para solucionar esse problema, as primeiras propostas que surgiram foram algoritmos que calculam a RNN baseando-se na pré-computação.

Considerando inicialmente $k = 1$, armazena-se para cada ponto s_i no conjunto, a sua *distância ao vizinho mais próximo*. Logo, uma consulta RNN sobre s_q pode ser facilmente processada pela verificação se s_q está dentro de um círculo com centro em s_i e raio igual a distância ao vizinho mais próximo de s_i.

Formalizando esse conceito, para cada $s_i \in S$, seja $s'_1 = kNN(s_i, 1)$ o seu vizinho mais próximo (NN) e seja $\xi_i = d(s_i, s'_i)$ a distância de cada elemento ao seu vizinho mais próximo. Então, define-se para cada elemento s_i a sua região de influência, que é o círculo centrado em s_i com raio ξ_i. Uma consulta $RkNN(s_q, 1)$ tem como resposta a coleção de todos os elementos s_i para os quais sua região de influência inclui s_q.

Esse processo pode ser generalizado armazenando-se para cada elemento $s_i \in S$ não apenas seu vizinho mais próximo, mas uma lista de k vizinhos mais próximos com suas respectivas distâncias, para algum $k \geq 1$. Contudo, apesar da pré-computação das distâncias aos k vizinhos mais próximo de todos os elementos simplificar a consulta, ela ocupa considerável espaço e requer extensiva manutenção quando o conjunto de dados é dinâmico, isto é, quando elementos podem ser inseridos ou apagados. Os métodos existentes na literatura para a execução do operador $RkNN$ em espaços euclidianos com pré-computação são descritos a seguir.

3.3.1 RNN-tree

A proposta de Korn e Muthukrishnan (2000) considera espaços multidimensionais e consiste em criar uma estrutura de indexação que armazena, para todo elemento s_i, o próprio elemento s_i e a área coberta por sua região de influência. No caso, a proposta foi de se usar o MBR de cada área coberta em uma R-tree, criando uma árvore que foi chamada RNN-tree. Usando essa árvore, o vizinho reverso de um elemento de consulta s_q pode ser obtido de maneira eficiente através de uma consulta pontual sobre s_q, pois ela retorna todas as áreas contendo s_q. Por ser apoiada na R-tree, esta proposta trata apenas da situação em que $k = 1$ em um espaço euclidiano (Korn e Muthukrishnan, 2000).

Korn e Muthukrishnan (2000) desenvolveram algoritmos e estruturas de dados para buscas RNN para conjuntos de dados estáticos e dinâmicos. No caso estático, eles uti-

lizam a árvore RNN-tree para a realização de consultas RNN eficientes e consultas aos vizinhos mais próximos. Contudo, para o caso dinâmico, eles utilizam uma R-tree adicional (referida como NN-tree) para consultas NN, pois a RNN-tree é otimizada para consultas RNN e não para consultas NN, causando grande sobreposições das regiões em operações de deleção.

Apesar do método proposto ser baseada no caso monocromático, Korn e Muthukrishnan (2000) descrevem a proposta como adaptável para o caso bicromático. Os algoritmos 3.1 e 3.2 contêm, respectivamente, o pseudocódigo de inserção e remoção. Para Inserir, o algoritmo especifica que inicialmente recuperam-se todos os RNN de s_q e suas correspondentes regiões de influência. Para cada RNN de s_q, atualiza-se a região de influência para refletir a inserção de s_q, além de inserir o novo elemento na NN-tree e RNN-tree.

Algoritmo 3.1: Inserir

Entrada: ponto s_q
1 $\{s_1,s_2,...,s_j\} \rightarrow$ consulta s_q em RNN-tree;
2 **para** cada s_i (com correspondente R_i) **faça**
3 | diminua R_i para $(s_i, d(s_i, s_q))$;
4 **fim**
5 Localize NN(s_q) em NN-tree;
6 Inserir s_q em NN-tree;
7 Inserir (s_q, NN(s_q)) em RNN-tree;

Algoritmo 3.2: Remover

Entrada: ponto s_q
1 remover s_q em NN-tree;
2 $\{s_1,s_2,...,s_j\} \rightarrow$ consulta s_q em RNN-tree;
3 remover (s_q, NN(s_q)) em RNN-tree;
4 **para** cada s_i (com correspondente R_i) **faça**
5 | Encontrar NN(s_i) em NN-tree;
6 | Aumentar R_i para (s_i, NN(s_i));
7 **fim**

3.3.2 RdNN-tree

Um desenvolvimento posterior foi proposto por Yang e Lin (2001) e melhora a proposta original de Korn e Muthukrishnan (2000) sincronizando as duas estruturas do caso dinâmico em apenas uma árvore, gerando a RdNN-tree, baseada em uma R^*-tree. A RdNN-tree difere da R^*-tree pelo armazenamento de informações extras sobre distância do NN dos pontos em cada nó, por isso requer a pré-computação de NN de cada ponto.

3.3.3 Outro método baseado em pré-computação

Maheshwari et al (2002) propõe outro método de pré-computação, porém ele é aplicável somente a espaços bidimensionais e foca na fronteira do pior caso — em lugar de comparações experimentais com as outras abordagens (Korn e Muthukrishnan, 2000; Yang e Lin, 2001). Eles observaram que Korn e Muthukrishnan (2000) propuseram a aproximação de cada bola com raio igual a distância ao vizinho mais próximo por uma MBR, usando a R-tree como estrutura de armazenamento. Na proposta de Korn e Muthukrishnan (2000), inserções e remoções podem ser executadas em tempo logarítmico, mas a complexidade das consultas no pior caso é linear. Além disso, por causa das aproximações, há a necessidade de um pós-processamento para que não haja respostas erradas.

Baseado nessas observações, a proposta de Maheshwari et al (2002) contém um método de pré-computação para espaços euclidianos bidimensionais em que foi criada uma estrutura de dados eficiente para o caso estático e dinâmico da consulta RNN, eliminando os problemas de utilizar MBR.

3.3.4 Considerações sobre os algoritmos com pré-computação

As propostas feitas em Korn e Muthukrishnan (2000), Maheshwari et al (2002) e em Yang e Lin (2001) pré-indexam todos os dados da base de dados segundo os vizinhos mais próximos naquele momento. Além do consumo de espaço, existe o problema que, quando um elemento do conjunto é removido ou um novo elemento é inserido, os dados sobre os vizinhos mais próximos têm que ser recalculados e a estrutura de indexação pode precisar ser bastante modificada, pois se um novo elemento é inserido, todos os elementos existentes que passam a tê-lo como elemento mais próximo têm que ser atualizados. Essa operação corresponde a atualizar os MBRs de todos os elementos que originalmente cobrem o novo elemento. Se um elemento é removido, toda a estrutura deve ser refeita, a menos que heurísticas especiais possam ser utilizadas em casos particulares. Dessa maneira, essas propostas não são adequadas para situações em que exista grande quantidade de operações de inserção e de remoções. Além disso, a generalização da operação de $RkNN$ de $k = 1$ para qualquer valor de $k \geq 1$ não é trivial, e gera uma sobrecarga de atualizações ainda mais intensa. Portanto, técnicas baseadas na pré-indexação das regiões de influência têm aplicações restritas a casos em que a base de dados não é dinâmica.

3.4 Algoritmos RNN euclidiano com propriedades para espaços de duas ou três dimensões

O uso de propriedades para limitar o espaço de busca, tanto de operações de consulta quanto de operações de atualização, é uma técnica que tem sido explorada para conjuntos

de dados com propriedades especiais, particularmente para o uso da métrica euclidiana em espaços de duas ou no máximo três dimensões. Um exemplo de uma propriedade bastante explorada é a seguinte propriedade.

Propriedade 3.1 *Em um espaço bidimensional regido pela métrica euclidiana, para qualquer elemento central de uma consulta s_q, a resposta a uma consulta $RkNN(s_q, 1)$ pode ter no máximo 6 elementos.(Smid, 1997; Korn e Muthukrishnan, 2000)*

Essa propriedade pode ser entendida lembrando que o caso limite máximo de seis elementos só ocorre quando as respostas estão igualmente espaçadas sobre as bissetrizes dos seis setores de 60° que definem o espaço ao redor de s_q. A figura 3.3(a) exemplifica esse caso especial com seis elementos igualmente espaçados entre si e em relação ao elemento de consulta s_q. Note-se que é possível haver no máximo um elemento em cada setor, que é o mais próximo do centro. Qualquer outro elemento no setor estará necessariamente mais próximo desse objeto do que do centro.

Espaços de outras dimensões têm limites diversos. Por exemplo, em espaços euclidianos tridimensionais, a operação $RkNN(s_q, 1)$ têm uma cardinalidade máxima igual a 12 e essa cardinalidade aumenta exponencialmente com o aumento da dimensão. Esse número também depende da função de distância envolvida. Por exemplo, em espaços E-dimensionais usando a métrica L_∞, a cardinalidade máxima da operação $RkNN(s_q, 1)$ é igual a $3^E - 1$. Uma explicação melhor sobre esses limites por ser encontrado em Singh et al (2003).

Por fim, salienta-se que esse crescimento exponencial no número de RNN com o aumento da dimensionalidade torna as abordagens utilizando essas propriedades geométricas inviáveis para dados em espaços de alta dimensionalidade.

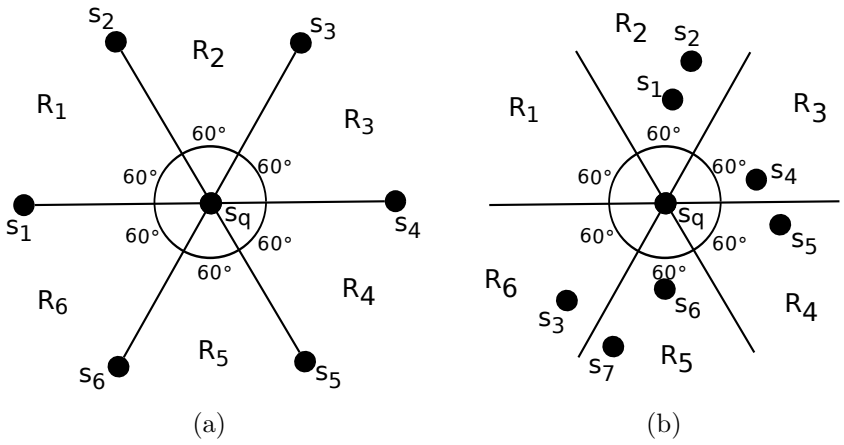


Figura 3.3: Ilustração do método de Stanoi et al (2000) baseado na propriedade 3.1

3.4.1 A abordagem de Stanoi et al. (2000)

Stanoi et al (2000) eliminaram a necessidade de pré-computação de todos NN pela exploração de uma interessante propriedade de recuperação de RNN. Essa propriedade é a 3.2, que é consequência da propriedade 3.1.

Propriedade 3.2 *Seja uma consulta $RkNN(s_q, 1)$ em um espaço euclidiano bidimensional que o divide em seis setores não sobrepostos S_i, $1 \leq i \leq 6$ com vértice em s_q, todos com ângulo de abertura de $60°$ e cada um deles incluindo uma das linhas que separam o espaço. Seja $s_i \in S_i$ o vizinho mais próximo do elemento s_q, qualquer que seja a região S_i. Então ou $s_i \in RkNN(s_q, 1)$ ou não existe nenhum vizinho reverso de s_q no setor S_i.*

Essa propriedade possibilita parar a busca por vizinhos reversos em um dos setores, caso o vizinho mais próximo do elemento de consulta no setor não seja RNN do elemento de consulta. É importante salientar que no caso de buscas por um $k > 1$, então, no espaço euclidiano bidimensional, o número máximo de vizinhos reversos será $6 \cdot k$.

Em resumo, o algoritmo desenvolvido por Stanoi et al (2000) inicia pela procura pelo vizinho mais próximo em cada uma das regiões S_i utilizando-se de uma consulta por NN condicional que limita o espaço de busca pela região de interesse (figura 3.3(b)). A seguir, ele testa se cada elemento s_i encontrado satisfaz a condição de ser um dos k vizinhos mais próximos reversos do elemento de consulta. Essa verificação se dá pela busca do NN dos elementos s_i e a comparação da distância entre s_i e seu NN com a distância entre s_i e o elemento de consulta s_q.

3.4.2 A abordagem de Stanoi et al. (2001) para o caso bicromático

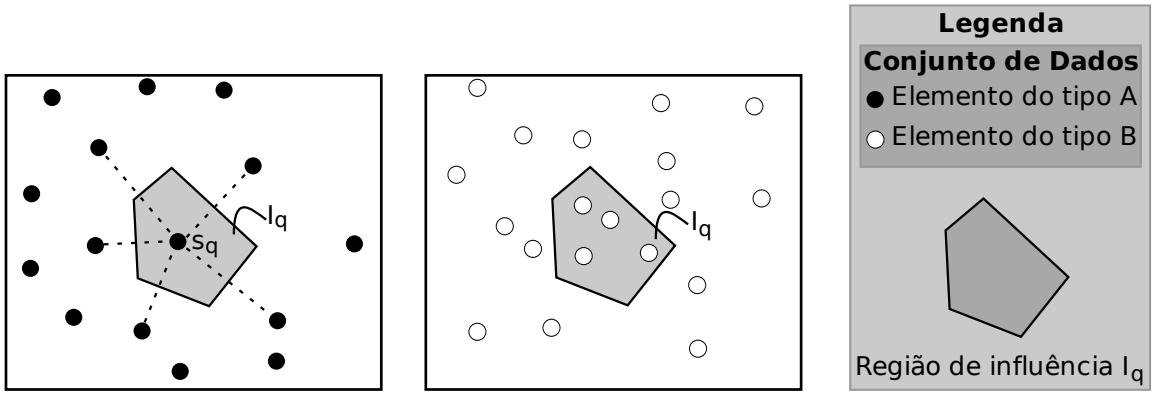
O primeiro algoritmo focado em consultas RNN bicromáticas foi desenvolvido por Stanoi et al (2001). A única abordagem anterior que pode ser aplicada para o caso bicromático é a abordagem com pré-computação, porém esta abordagem apresenta restrições quanto à base de dados ser dinâmica e quanto à generalidade da consulta (veja seção 3.3). A solução apresentada em Stanoi et al (2000) apenas é válida para consultas monocromáticas, pois a propriedade geométrica de que se utiliza não é extensível para o caso bicromático (Stanoi et al, 2001).

O método proposto em Stanoi et al (2001) divide o processamento da consulta em duas fases:

- Primeiramente, dado um conjunto de dados do tipo A armazenado em uma R-tree e um elemento de consulta s_q, calculam-se os limites da região de influência I_q.

- Na segunda fase utiliza-se a região de influência encontrada e para recuperar todos os pontos do tipo B, armazenados em outra R-tree, que estão dentro desta região.

A figura 3.4 exemplifica as duas fases do método, sendo que a figura 3.4(a) representa a primeira fase do método e a figura 3.4(b) representa a segunda fase.



(a) Fase 1: Computa a região de influência I_q

(b) Fase 2: Recupera pontos baseado em I_q

Figura 3.4: Exemplo de uma consulta RNN em duas fases do método de Stanoi et al (2001)

A segunda fase do algoritmo é simplesmente uma consulta por abrangência dentro dos limites da região de influência I_q. Entretanto, a primeira fase do método precisa ser melhor definida, pois a solução básica para computação da região de influência consiste em comparar a posição de s_q com todos os elementos do tipo A.

A forma proposta para reduzir esse custo é utilizar aproximações para encontrar a região de influência. Essa fase é subdividida em duas etapas. Na primeira etapa é computada uma aproximação da região de influência $Approx(I_q)$. Esta aproximação ocorre com a localização dos NN dos dados do tipo A em cada um dos quadrantes que junto com o elemento de consulta, possibilita determinar esta região aproximada $Approx(I_q)$. Como segunda etapa, é realizado um refinamento desta aproximação que define os limitantes da região que melhoram a primeira aproximação, garantindo assim uma resposta correta.

3.5 Algoritmos $RkNN$ com aproximação

As abordagens usadas nos algoritmos com aproximação propiciam uma execução eficiente em dados de alta dimensionalidade ou com valores grandes de k, porém não garantem o retorno de todos os resultados. A seguir são apresentadas as abordagens de Singh et al (2003) e de Xia et al (2005), ambas baseadas em aproximação para buscas $RkNN$ em dados de alta dimensionalidade.

3.5.1 Abordagem de Singh et al. (2003)

O método proposto por Singh et al (2003) é baseado na premissa que as consultas $RkNN(s_q, k)$ e $kNN(s_q, K)$ são correlacionadas para um K suficientemente maior que k. O algoritmo consiste em recuperar candidatos através de uma consulta aos K vizinhos mais próximos de s_q e posteriormente, verificar os candidatos através de consultas por abrangência.

Contudo, a correlação entre $RkNN$ e kNN não é forte. Logo, K deve ser suficientemente grande para reduzir *falsos negativos*[1], o que o faz muito custoso para valores muito grandes de k.

Algoritmo RNN

A figura 3.5, encontrada em Singh et al (2003, p. 93), resume o processamento da consulta RNN. O método é dividido em duas etapas, sendo que a segunda, conhecida como etapa de filtragem, é feita em dois passos.

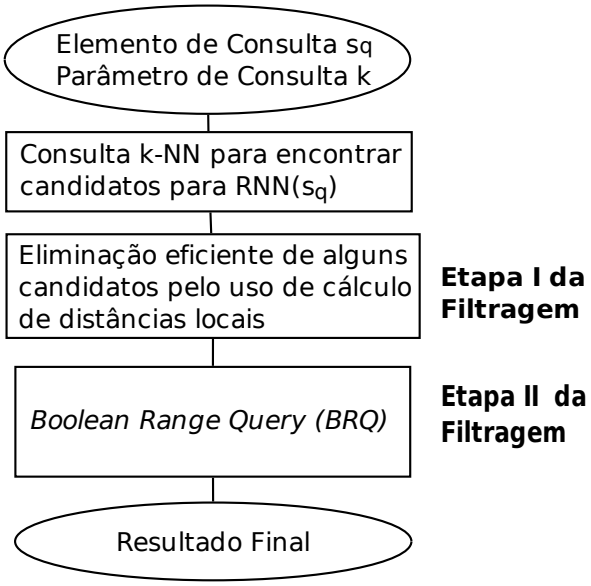


Figura 3.5: Algoritmo proposto para RNN por Singh et al (2003, p. 93)

A primeira etapa é a execução de uma consulta kNN que retorna os candidatos a serem analisados na próxima etapa. O valor k desta consulta é dependente do conjunto de dados e pode ser estimado por experimentos, como mostrado no artigo.

Após essa etapa, inicia-se a etapa de filtragem. O primeiro passo da etapa de filtragem elimina elementos do conjunto de candidatos pelo cálculo da distância local, verificando se dentre os candidatos, um elemento tem k outros candidatos mais próximos de si antes do elemento de consulta s_q. No caso de um algoritmo RNN, o valor de k é um.

[1]No caso de consultas $RkNN$, *falsos negativos* são elementos que são $RkNN$, mas não aparecem no conjunto resposta

O segundo e último passo da etapa de filtragem utiliza o conceito *Boolean Range Queries* para eliminar candidatos que não são RNN do elemento central de consulta. Uma *Boolean Range Queries* $BRQ(s_q, r)$ retorna verdadeiro se o conjunto $\{s_i \in S \mid d(s_q, s_i) < r\}$ não é vazio e retorna falso caso contrário, sendo mais eficiente que uma consulta R_q. Assim, se a *Boolean Range Queries* retorna falso, então o elemento é RNN, caso contrário ele não é.

Variantes da consulta RNN

Singh et al (2003) também descrevem em seu artigo como adaptar o método para realizar consultas $RkNN$. Eles definem o conceito *Count Range Queries* que substitui o algoritmo de *Boolean Range Queries* no último passo da segunda etapa do método proposto. O algoritmo de *Count Range Query*, considerado uma generalização do *boolean range query* para k arbitrário, realiza uma consulta $R_q(s_i, \xi_i)$, em que $\xi_i = d(s_i, s_q)$, verificando se a consulta R_q sobre o candidato s_i contém k outros elementos.

3.5.2 Abordagem ERkNN

Xia et al (2005) apresentam uma alternativa de consulta $RkNN$ baseada em estimativas que faz uso de um *framework* de filtragem e refinamento. O algoritmo 3.3 contém o pseudocódigo da consulta $RkNN$ baseado em estimação (ERkNN), contendo claramente a etapa de filtragem e uma posterior etapa de refinamento.

Algoritmo 3.3: $ERkNN(T, s_q, k)$

Entrada: T é uma árvore de indexação, sé o elemento de consulta, k é um inteiro
Resultado: Conjunto resposta de $RkNN$s

```
1 A = ∅;
2 Filtragem(T, s_q, k, A);
3 Refinamento(T, s_q, k, A);
```

A etapa de filtragem encontra candidatos a $RkNN$ baseada nas estimativas de *distância* kNN (distância de um elemento ao seu k-ésimo vizinhos mais próximo). Esse filtro baseado em estimativas possibilita um baixo custo computacional, melhorando a velocidade das consultas $RkNN$. Por conseguinte, a etapa de refinamento verifica os candidatos através de consultas por abrangência.

São propostos dois métodos para a estimação da distância kNN local, o método PDE e o método kDE. O método PDE é baseado no estimador de densidade de *Parzen* com núcleo uniforme, que é o mais comumente utilizado estimador de densidade não paramétrica. Já o método kDE é baseado numa dedução do estimador de densidade kNN (Xia et al, 2005).

Os procedimentos para a filtragem e refinamento da ERkNN são implementados tendo como base de armazenamento a estrutura de dados *RdNN-tree* (Yang e Lin, 2001). A modificação realizada nessa estrutura para suportar a proposição deste artigo consiste em definir uma constante $\mathcal{K}$ suficientemente grande que garanta uma boa aproximação e armazenar as distâncias $\mathcal{K}NN$ em vez que armazenar as distâncias ao NN.

O procedimento de filtragem percorre a árvore aplicando a seguinte estratégia de poda: Se $MinDist(N_j, s_q) \geq MaxEQ(N_j)$, então o nó N_j pode ser podado. $MaxEQ(N_j)$ é a estimativa de distância kNN e é definido pelo método PDE como

$$MaxEQ(N_j) = MaxDnn_{\mathcal{K}} \cdot \sqrt[d]{\frac{k}{\mathcal{K}}} \tag{3.1}$$

e para o método kDE como

$$MaxEQ(N_j) = \left\{ \begin{array}{cc} MaxDnn_{\mathcal{K}} \cdot \prod_{i=\mathcal{K}}^{k-1}(1+\frac{1}{i \cdot d}) & para \text{ k}> \mathcal{K} \\ MaxDnn_{\mathcal{K}} & para \text{ k}= \mathcal{K} \\ \frac{MaxDnn_{\mathcal{K}}}{\prod_{i=k}^{\mathcal{K}-1}(1+\frac{1}{i \cdot d})} & para \text{ k}< \mathcal{K} \end{array} \right\} \tag{3.2}$$

Desde de que $MaxDnn_{\mathcal{K}} = Max_{i=1}^{m} dnn_{\mathcal{K}}(s_i)$, $MaxED(N_j) = Max_{i=1}^{m} ednn_k(s_i)$ onde $ednn_k(s_i)$ é a estimativa de distância kNN de s_i, sendo s_i um elemento em N_j.

O processo de Refinamento realizado após a primeira etapa tem uma fase em memória, verificando se os candidatos podem ser eliminados através de uma consulta R_q sobre o conjunto de candidatos, e uma fase completa, verificando se os candidatos restantes podem ser eliminados através de uma consulta R_q na RdNN-tree.

3.6 Abordagem TPL

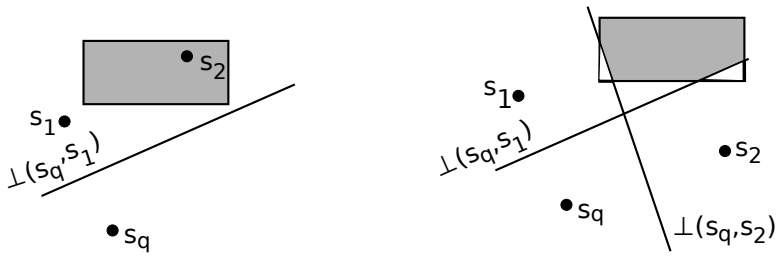
Tao et al (2004) desenvolveram uma proposta baseada na propriedade 3.3.

Propriedade 3.3 *Dado um elemento qualquer s_i da base de dados em um espaço e um elemento central de consulta s_q, a bissetriz da reta que passa por s_i e s_q divide o espaço em dois semi-espaços: um que contém s_i e outro que contém s_q. Nenhum elemento que esteja no mesmo semi-espaço que contém s_i pode estar em $RkNN(s_q, 1)$, uma vez que esse elemento necessariamente estará mais próximo a s_i do que de s_q.*

Assim Tao et al (2004) propõe armazenar conjuntos de dados multidimensionais com distância euclidiana em uma R-tree e utilizar-se a propriedade 3.3 para efetuar a poda de nós baseando-se nas distâncias dos vértices de seus MBR a s_i e a s_q. Para isso é usada a distância euclidiana de um elemento (cada um dos vértices do MBR) ao hiperplano que bisecciona o espaço, agilizando a resposta a consultas $RkNN$ em espaços multidimensionais de qualquer dimensão usando a métrica euclidiana.

A técnica é então generalizada para tratar qualquer número k de vizinhos reversos, combinando semi-espaços gerados por k elementos da base de dados. No entanto, essa combinação reduz rapidamente a capacidade de poda. Assim, embora a técnica seja aplicável a conjuntos de dados com dimensionalidades maiores do que 2 ou 3, ela ainda é limitada a baixas dimensionalidades (uma vez que a técnica baseia-se nas R-trees, as quais notoriamente degradam rapidamente e se tornam inviáveis para dimensionalidades superiores a 10 (Berchtold et al, 1996)), e é efetiva apenas para valores baixos de k. Na realidade, Tao et al (2004) mostram o comportamento da técnica apenas para um limite máximo de $k = 16$ e dimensionalidade máxima de 5 dimensões. Além disso, a técnica é restrita à utilização da distância euclidiana.

A figura 3.6 apresenta exemplos de poda dos semi-espaços, sendo a figura 3.6(a) exemplo da abordagem com $k = 1$ e a figura 3.6(b) da abordagem com $k = 2$.



(a) Poda com um elemento (b) Poda com dois elementos

Figura 3.6: Ilustração da poda por semi-espaços (Tao et al, 2004)

3.7 $RkNN$ em espaços métricos

Todas as abordagens apresentadas até aqui são implementadas em métodos de acesso espaciais, em especial na R-tree e em suas variantes. Dentre essas abordagens, as únicas que podem ser adaptadas para espaços métricos são as de pré-computação de Korn e Muthukrishnan (2000) e a técnica de Yang e Lin (2001), porém elas não são aplicáveis a k arbitrário, nem a bases de dados dinâmicas.

Nesta seção, são apresentadas as abordagens que tratam da implementação especificamente em espaços métricos.

3.7.1 Abordagem de Tao et al. (2006)

Uma técnica proposta para espaços métricos é apresentada em Tao et al (2006). Ela consiste na implementação de uma consulta $RkNN$ exata e eficiente que funciona para todos os tipos de métricas (euclidianas ou não) e funções de distância, porém limita o valor de k a f_{min} (capacidade mínima de um nó da árvore) e seu desempenho degrada rapidamente com o aumento de k.

Os autores assumem 3 premissas para propor o algoritmo: O processo da consulta utiliza somente as distâncias entre os elementos. Os dados são indexados em uma M-tree e o parâmetro k da consulta é menor que f_{min} da M-tree. Além disso, o algoritmo é baseado na observação do lema 3.1 e dele derivam-se todas as regras de poda.

Lema 3.1 *Seja s_i um elemento indexado, s_{r_i} o elemento representante de uma subárvore, $maxND_k(s_i)$ a distância máxima em que se encontrará o k-ésimo vizinho mais próximo de s_i e $maxND_k(s_{r_i})$ a distância máxima a que, dentre todos os elementos da subárvore, encontra-se o k-ésimo vizinho mais próximo deles. A subárvore N_i de um nó interno não pode conter nenhum resultado de uma consulta $RkNN$ se a distância mínima $MinDist(s_{r_i}, s_q)$ entre s_{r_i} e s_q é ao menos $maxND_k(s_{r_i})$. Similarmente, um elemento s_i não pode satisfazer s_q se $d(s_i, s_q) \geq maxND_k(s_i)$.*

A figura 3.7(a) ilustra as regras de poda para os nós internos da M-tree. Considerando s_q como o elemento de consulta, s_{r_i} e ξ_{r_i} respectivamente o elemento representante e o raio de cobertura da subárvore, se $k = 1$ então a subárvore pode ser podada (não necessita ser visitada) se $d(s_{r_i}, s_q) > 2\xi_{r_i}$. Para $k > 1$, há duas regras de poda. A primeira diz que a subárvore pode ser podada se $d(s_{r_i}, s_q) > 3\xi_{r_i}$. Contudo a segunda regra mostra que se existirem $k - 1$ elementos s_i já processados na consulta de modo que $d(s_{r_i}, s_q) \geq 2\xi_{r_i} + max_{i=1}^{k-1}(2\xi_{r_i} + d(s_{r_i}, s_i))$, então a subárvore pode ser podada. Para diminuir o cálculo de distâncias exigidas por essas regras, embora perdendo poder de poda, é proposta uma terceira regra que consiste em substituir $d(s_{r_i}, s_q)$ por $\mid d(s_{r_p}, s_q) - d_{s_{r_i} s_{r_p}} \mid$, sendo $d_{s_{r_i} s_{r_p}}$ a distância de s_{r_i} ao elemento pai s_{r_p} da subárvore que o contém.

A figura 3.7(b) ilustra as podas para nós folhas. A regra é que um elemento s_i pode ser podado se $d(s_i, s_q) \geq maxND_k(s_i)$ e, similarmente as justificativas dadas para a terceira regra de poda para nós internos, há uma regra que permite substituir $d(s_i, s_q)$ por $\mid d(s_{r_p}, s_q) - d_{s_i s_{r_p}} \mid$.

Baseando-se nestas regras, foi proposto um algoritmo de filtragem e refinamento. Na etapa de filtragem, é encontrado um conjunto de candidatos S_{can} executando um percurso em profundida na M-tree segundo as regras de poda. Esse candidatos são então verificados na etapa de refinamento por um algoritmo kNN adaptado, parando assim que o candidato é confirmado como não sendo um dos $RkNN$.

3.7.2 Abordagem MRkNNCoP-tree

Achtert et al (2006b) propõe uma nova estrutura de indexação que é uma extensão da M-tree (Ciaccia et al, 1997) para resolver as consultas $RkNN$. Conceitualmente similar a RdNN-tree (Yang e Lin, 2001), esta nova estrutura, conhecida como MRkNNCoP-tree, armazena aproximações progressivas e conservativas para todas as distâncias kNN ao invés da distância kNN de um k pré-fixado. A limitação da proposta dos autores é que

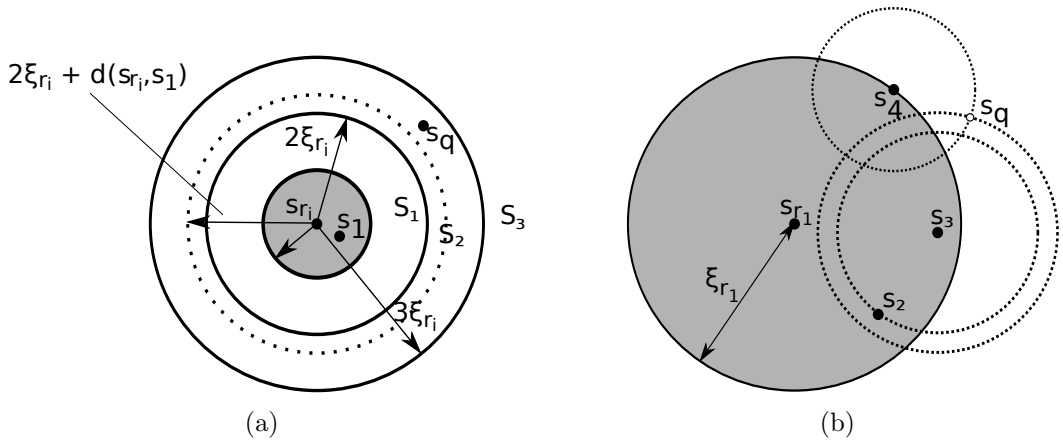


Figura 3.7: Ilustração das regras de poda (Tao et al, 2006)

k deve ser menor ou igual a um k_{max} que é dependente de como a estrutura de dados é criada. Assim, para consultas com $k > k_{max}$ não há garantia de resultados corretos (Achtert et al, 2007, 2006b).

A aproximação conservativa é sempre maior ou igual a distância kNN real. Logo, ela permite identificar elementos que podem se descartados se a distância entre ele o elemento de consulta estiver mais distante do que essa aproximação. Além disso, a aproximação progressiva é sempre menor ou igual a distância real, possibilitando encontrar elementos $RkNN$ se a distância entre ele e o elemento de consulta é inferior a aproximação progressiva. Assim, os elementos que tem distância ao elemento de consulta menor ou igual que a aproximação conservativa e maior que aproximação progressiva são candidatos a $RkNN$ e devem passar por um refinamento. Este refinamento consiste em realizar consultas kNN sobre os candidatos e verificar se s_q pertence ao conjunto resposta.

A figura 3.8 ilustra a ideia do uso destas aproximações. Se s_{q_1} é o elemento de consulta, s_1 é $RkNN$ dele e não necessita de nenhum refinamento. Agora, se s_{q_3} é o elemento de consulta, s_1 pode ser seguramente descartado, pois não será $RkNN$. Enfim, se s_{q_2} é o elemento de consulta , s_1 é um candidato e precisa passar por um refinamento para verificar se é $RkNN$ do elemento de consulta.

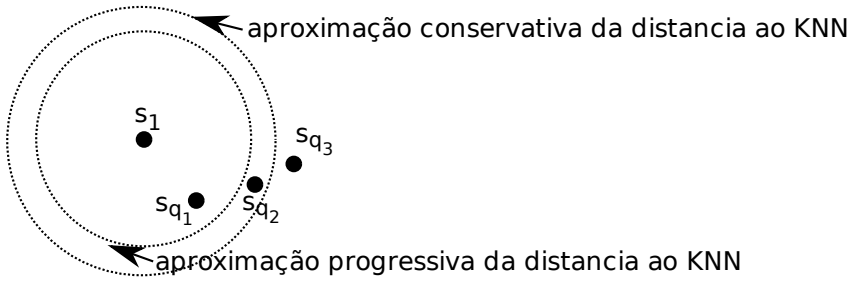


Figura 3.8: Ilustração do uso de distâncias kNN conservativa e progressiva (Achtert et al, 2006b)

3.7.3 Abordagem AMRkNN-tree

Uma solução para o problema de realizar consultas $RkNN$ em espaços métricos para um k arbitrário é proposto em Achtert et al (2006a) e posteriormente estendido em Achtert et al (2007). Esta solução, entretanto, fornece apenas resultados aproximados.

Similar à proposta de Achtert et al (2006b), o algoritmo da consulta é implementado numa estrutura de indexação desenvolvida especificamente para essa proposta, a AMRkNN-tree. Essa estrutura é uma extensão da M-tree (Yang e Lin, 2001) que armazena aproximações de todas as distâncias kNN para um parâmetro k_{max} que não é limitante da consulta. Uma função foi proposta para calcular a distância kNN para qualquer k utilizando as aproximações armazenadas. O algoritmo faz um percurso único sobre a árvore, usando aquela função para verificar as sub-árvores e elementos que podem ser descartados e retornando elementos que são considerados $RkNN$ do elemento de consulta, sem a necessidade de um etapa de refinamento.

Assim, a proposta possibilita consultas para um k arbitrário e com desempenho superior em relação as outras propostas, porém com resultados aproximados.

3.8 A abordagem k-counting e k-browsing

Recentemente, Lee et al (2008) propuseram duas abordagens para resolver uma nova variante da consulta $RkNN$ na R-tree, a consulta aos vizinhos mais próximos reversos *ranqueado* ($RRNN$). Essa consulta parte da ideia que o objetivo principal da consulta $RkNN$ é determinar o *conjunto de influência*, isto é, o subconjunto de elementos em S considerados influenciados pelo elemento de consulta s_q. Assim, é definido o conceito de *grau de influência* (κ_{s_p}) conforme a definição 3.1.

Definição 3.1 ***Grau de influência***: *dado um conjunto de dados S e um elemento de consulta s_q, o grau de influência de s_q em $s_p \in S$, denotado por κ_{s_p}, é o número de elementos não mais distântes de s_q do que de s_p. Formalmente, $\kappa_{s_p} = |\{s'_p | s'_p \in X \cup \{s_p\} \wedge d(s'_p, s_p) \leq d(s'_p, s_q)\}|$ onde $X = S - \{s_q\}$ (caso monocromático) (Lee et al, 2008).*

Utilizando essa definição de grau de influência, a solução de uma consulta $RRNN(s_q, t)$ obtém t *tuplas* (s_i, k_{s_i}) ordenadas de forma decrescente por k_{s_i}, onde $s_i \in S$ e k_{s_i} é o grau de influência. Ambas as abordagens propostas para solucionar a consulta $RRNN$, conhecidas como *k-counting* e *k-browsing*, progressivamente obtêm um grau de influência estimado para um subconjunto de elementos realizando apenas um percurso na árvore. A diferença chave entre as abordagens está na função de ordenação adotada e no número de elementos visitados durante o processamento da consulta.

Objetivando otimizar o número de elementos processados, o algoritmo k-browsing destacou-se por seu melhor desempenho experimental em termos de acesso a disco e

tempo. Essa algoritmo permite a otimização através da visita aos elementos em ordem de seus graus de influência e, por isso, uma noção de min_k é introduzido e usado no algoritmo com o propósito de eficientemente processar a consulta $RRNN$. O min_k de uma elemento é a estimativa de um limitante mínimo de k baseado em distâncias métricas e utilizando-se apenas de desigualdade triangular.

3.9 Abordagem AKKRZ

A primeira solução para a consulta $RkNN$ em espaços métricos sem restrição é conhecida como AKKRZ (Achtert et al, 2009). Essa abordagem AKKRZ combina grupos de abordagens existentes (*self-prunning* e *mutual-pruning*) com o propósito de reduzir o número de acessos a disco durante o percurso no MAM, produzindo os resultados com apenas uma etapa. Embora o algoritmo desenvolvido para realizar a consulta $RkNN$ seja diferente dos desenvolvidos por Lee et al (2008), as estratégias de podas desenvolvidas para o cálculos limitantes mínimos e máximos deste artigo são semelhantes aos demonstrado por Lee et al (2008) no algoritmo k-browsing.

3.10 Considerações finais

A tabela 3.1 resume os métodos para implementação da consulta $RkNN$ com as considerações pertinentes. Verifica-se que, até hoje, quase a totalidade das técnicas de busca por vizinhos mais próximos reversos têm pelo menos uma das seguintes limitações: (a) estão restritas a valores de $k = 1$ ou k limitado; (b) operam apenas em espaços multidimensionais de dimensionalidade muito baixa (2 ou 3 no máximo); (c) operam apenas em espaços com métrica Euclidiana; (d) podem ser usadas apenas em conjuntos de dados estáticos; (e) apresentam respostas aproximadas.

Nesta monografia, desenvolveu-se uma técnica para executar buscas exatas pelos k-vizinhos mais próximos reversos em conjuntos de dados dinâmicos em espaços métricos, que será apresentada no capítulo seguinte. Com isso, as limitações (b) e (c) são superadas pela própria definição do modelo de espaço que será considerado, uma vez que, baseando-se em uma métrica qualquer, tanto a restrição da métrica ser euclidiana quanto a própria necessidade de haver dimensões desapareceram. As demais limitações foram superadas utilizando técnicas de indexação em espaços métricos e técnicas de podas apropriadas.

Métodos	Considerações
Algoritmos com pré-computação	Bases de dados não dinâmica com $RkNN$ para $k = 1$
Algoritmos com propriedades euclidianas	Espaço euclidiano bidimensional
Algoritmos com aproximação	Solução aproximada
Algoritmo TPL	Espaço Euclidiano - solução exata com valor de k do operador $RkNN$ limitado
Tao et al (2006) e MRkNNCoP-tree	Solução exata com valor de k do operador $RkNN$ limitado superiormente
AMRkNN-tree	Solução aproximada
k-counting, k-browsing e AKKRZ	Soluções exatas

Tabela 3.1: Resumo dos métodos para implementação da consulta $RkNN$ apresentados neste capítulo

Capítulo

4

Algoritmo $RkNN$-MG e podas utilizando lei dos cossenos

4.1 Considerações iniciais

A abordagem proposta neste trabalho combina a eficiência da desigualdade triangular para a poda de elementos no espaço de busca com uma inovação criada neste trabalho, que é a poda eficiente de subárvores pela leis dos cossenos em espaços métricos. Esta proposta assume que o conjunto de dados está em um espaço métrico isometricamente imersível num espaço euclidiano. A descrição feita neste capítulo, bem como os experimentos apresentados no capítulo seguinte assume que os dados estão indexados numa *Slim-tree*, mas os conceitos desenvolvidos podem ser estendidos para a maioria das estruturas de indexação de dados em espaços métricos. Para efeito de facilitar a compreensão, as explicações e ilustrações presente neste capítulo são em espaço euclidianos bidimensionais, que são mais fáceis de se ilustrar graficamente e mais intuitivas, mas não se limitam a este espaço, pois os conceitos desenvolvidos são genéricos para qualquer espaço métrico isometricamente imersível num espaço euclidiano.

A seção 4.2 deste capítulo contém uma explanação geral do algoritmo $RkNN$-MG, enquanto a seção 4.3 detalha todos as podas propostas e o conceito min_k utilizado. Por fim, a seção 4.4 apresenta o algoritmo desenvolvido, que é utilizado para a avaliação da eficiência das novas podas propostas.

4.2 Visão geral do algoritmo $RkNN$-MG

O algoritmo $RkNN$-MG (Reverse k-nearest neighbor query using metric geometry - consulta aos k-vizinhos mais próximos reversos usando geometria métrica) tem como premissa ordenar os acessos aos elementos ou as entradas de nós índices baseado na vizinhança conhecida dos elementos de resposta. A figura 4.1 ilustra a ideia do algoritmo, que atua analisando um a um (explorando) elementos próximos ao centro de consulta. Baseado em dois elementos ainda não explorados s_i e s_j, o algoritmo deve explorar antes o elemento s_j que tenha o grau de influência κ_{s_j} menor que o grau de influência κ_{s_i} do elemento s_i. Contudo, os graus de influências exatos não são conhecidos sem explorar outros elementos e entradas de nós. Por isso, uma questão que este algoritmo apresenta é como determinar a ordem de acesso entre s_i e s_j sem o conhecimento de seus graus de influência.

Como solução, é utilizada a noção de min_κ, associado a todo elemento ou entrada de nós índice, para representar o número mínimo de elementos que estão mais próximos aos seus elementos associados ou entrada de nós índice do que ao elemento de consulta. O conceito de min_κ foi inicialmente proposto para o algoritmo *k-browsing*, porém, no $RkNN$-MG, esse conceito é melhorado utilizando-se de geometria em espaços métricos. Basicamente, min_κ é o limitante mínimo do grau de influência de um elemento ou todos os elementos internos a um nó da árvore. Um elemento ou entrada de nó índice com um min_κ relativamente grande obviamente tem menos probabilidade de ser ou conter os elementos mais influentes, e assim a prioridade de acesso deve ser para quem tem menor min_κ. Usando o conceito de min_κ, o algoritmo pode também podar eficientemente o espaço de busca.

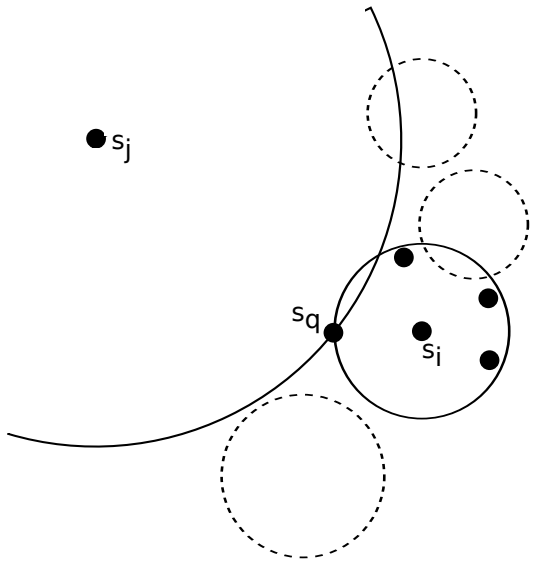


Figura 4.1: Visão geral do algoritmo $RkNN$-MG

4.3 Conceito min_κ utilizando geometria métrica

Neste trabalho, a estimativa para min_κ proposta explora as propriedades de imersão isométrica de espaços métricos em espaços euclidianos e usa novas estratégias que se beneficiam da lei dos cossenos e de pontos de tangência em subárvores da *Slim-tree*.

Basicamente, o método proposto utiliza a propriedade 4.1 definida a seguir, definida para o algoritmo TPL. A propriedade 4.1 é válida para espaços métricos imersos isometricamente num espaço euclidiano, os quais também possibilitam empregar lei dos cossenos para realizar o cálculo da distância de uma bola da árvore a um hiperplano. Considere-se o hiperplano H definido por dois pontos s_i e s_q formado pelos pontos s_h tal que $d(s_i, s_h) = d(s_q, s_h)$, que separa espaço original em dois subespaços. A questão de como podar regiões definidas pelos nós que constituem o MAM depende de como o cálculo da distância de um elemento ao hiperplano H é definida. Para espaços métricos isometricamente imersíveis num espaço euclidiano, o cálculo dessa distância pode utilizar a lei dos cossenos e ser expressa apenas em função da distância entre três pontos s_q, s_i e s_j da seguinte maneira: Considere-se um elemento hipotético s_x tal que a distância de s_x ao hiperplano seja igual à distância de s_j ao hiperplano e que $d(s_q, s_i) = d(s_q, s_x) + d(s_x, s_i)$ (veja figura 4.2). A lei dos cossenos permite que a distância de s_i a s_x seja determinada pela equação $d(s_i, s_x)^2 = d(s_j, s_i)^2 - d(s_q, s_j)^2 + d(s_q, s_x)^2$. Desenvolvendo-se essas equações, chega-se que a distância de s_j ao hiperplano orientado pelo eixo coordenado $\overrightarrow{s_i s_q}$ é dada por:

$$d_H(s_q, s_i; s_j) = \frac{d(s_q, s_j)^2 - d(s_i, s_j)^2}{2 \times d(s_q, s_i)} \tag{4.1}$$

Com isso, pode-se utilizar a propriedade 4.1 para realizar a poda de sub-árvores de uma estrutura de acesso métrico indexando os elementos a serem pesquisados.

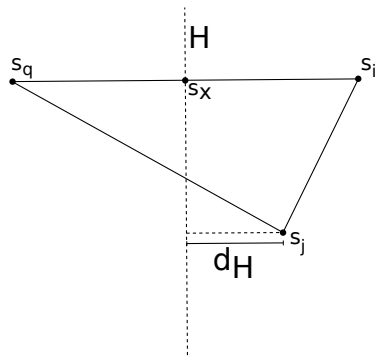


Figura 4.2: Ilustração do cálculo da distância ao hiperplano

Propriedade 4.1 *Dado um elemento qualquer s_i da base de dados em um espaço métrico e um elemento central de consulta s_q, a bissetriz da reta que passa por s_i e s_q divide o espaço em dois semi-espaços: um que contém s_i e outro que contém s_q. Nenhum elemento*

que esteja no mesmo semi-espaço que contém s_i pode estar em $RkNN(s_q, 1)$, uma vez que esse elemento necessariamente estará mais próximo de s_i do que de s_q.

A equação 4.1 baseia-se apenas em distâncias entre elementos conhecidos e permite podas equivalentes àquelas utilizando a propriedade da desigualdade triangular. Contudo, pode-se estender este cálculo para determinar as distância ao hiperplano das entradas dos nós índice da árvore. Essa extensão é mais eficiente do que a poda por desigualdade triangular, como pode ser observado nos experimentos apresentados do capítulo seguinte.

Antes de realizar a demonstração das novas podas derivadas da equação 4.1, cabe comentar algumas propriedades e equações auxiliares que auxiliam no entendimento dos cálculos que devem ser realizados. A equação 4.2 representa a distância do elemento s_j a reta que passa por $\overline{s_q s_i}$ (altura do triângulo $\triangle s_q s_i s_j$, como exemplificado na figura 4.3).

$$\begin{aligned} height(s_q, s_i; s_j) = \tfrac{1}{2 \times d(s_q, s_i)} \times \sqrt{2(d(s_q, s_i)^2 d(s_i, s_j)^2 + } \\ \overline{d(s_q, s_i)^2 d(s_q, s_j)^2 + d(s_i, s_j)^2 d(s_q, s_j)^2) -} \\ \overline{(d(s_q, s_i)^4 + d(s_i, s_j)^4 + d(s_q, s_j))} \end{aligned} \tag{4.2}$$

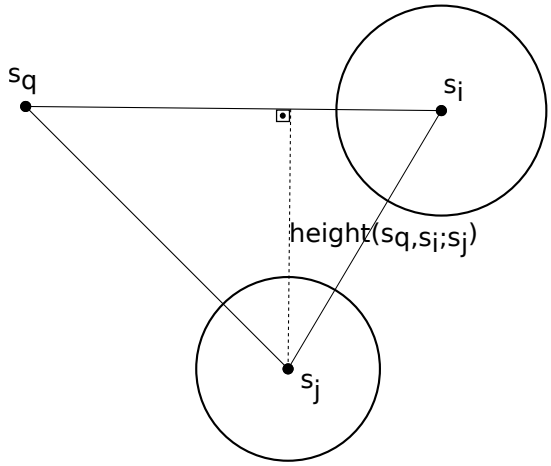


Figura 4.3: Ilustração de $height(s_q, s_i; s_j)$

Além desta equação, as propriedades 4.2 e 4.3 permitem decidir quando elementos ou entradas de nós índices não auxiliam a estimativa do grau de influência nem parcialmente nem completamente na estimativa do min_κ de um elemento ou entrada de nó índice sem realizar nenhum cálculo extra. A figura 4.4 exemplifica a propriedade 4.2 e contém o elemento de consulta s_q coberto pela bola da entrada de nó índice N_j, enquanto que a propriedade 4.3 é ilustrada na figura 4.5, a qual mostra que $minDist(s_q, N_i) >= 2maxDist(s_q, N_j)$. Por fim, usa-se a seguinte notação: as subárvores do MAM são notadas como N_x tendo o elemento representante s_x e raio de cobertura ξ_x. Para manter a generalidade e para evitar repetições nas explicações, considera-se que um elemento s_x é representado por N_x com elemento representativo s_x e raio de cobertura 0.

Propriedade 4.2 *Dado um elemento de consulta s_q, uma entrada de nó índice N_j e um conjunto V de elementos e entradas de nós índices, se $minDist(s_q, N_j) = 0$ então nenhum dado em V incrementará o $min_\kappa(N_j, s_q, V)$ para todo N_j e, portanto, um refinamento de N_j é necessário.*

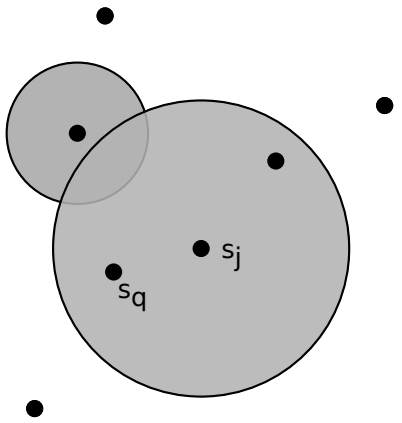


Figura 4.4: Ilustração da propriedade 4.2

Propriedade 4.3 *Dado um elemento de consulta s_q, uma entrada de nó índice N_j e um elemento ou entrada de nó índice N_i pertencente ao conjunto de elementos e entradas de nós índices mantidos em V, se $minDist(s_q, N_i) >= 2maxDist(s_q, N_j)$, então nenhum elemento que pertence a N_i contribui para $min_\kappa(N_j, s_q, V)$ e, portanto, pode ser podada da verificação em N_j.*

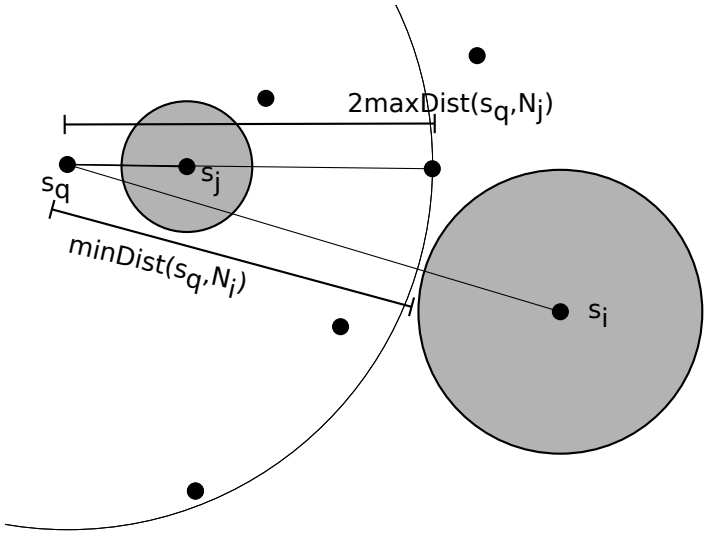


Figura 4.5: Ilustração de propriedade 4.3

A partir deste ponto, apresenta-se progressivamente as explicações do cálculo da distância ao hiperplano para entradas de nós índices com o objetivo de facilitar a compreensão

dos resultados obtidos. Inicialmente, considera-se que a verificação da distância ao hiperplano não é apenas de um elemento s_j, sendo, portanto, uma entrada de nó índice N_j. Essa generalização, presente na equação 4.3, é facilmente definida mantendo o mesmo cálculo básico anterior e diferenciando apenas a verificação de poda, que ao invés de podar se essa distância for maior que 0, poda-se sempre que a distância for maior que ξ_j.

$$d_H(s_q, s_i; N_j) = \frac{d(s_q, s_j)^2 - d(s_i, s_j)^2}{2 \times d(s_q, s_i)} \tag{4.3}$$

A próxima generalização é feita pela verificação que o auxiliar da poda s_i não é simplesmente um elemento, mas uma entrada de um nó índice da árvore N_i (e, portanto, s_i é o representativo de uma subárvore que define uma bola de raio ξ_j). Para melhorar a eficiência do algoritmo, foram desenvolvidos novos cálculos que são baseados na propriedade equivalente da geometria plana "*Por um ponto s_i externo a uma circunferência é possível traçar duas retas que tangenciam a circunferência em pontos distintos, cujas distâncias ao ponto dado são iguais*". Essa propriedade pode ser generalizada para espaços euclidianos de qualquer dimensionalidade considerando uma híper-esfera no lugar da circunferência, e também para espaços métricos isometricamente imersíveis em espaços euclidianos considerando-se que as duas retas devem estar no mesmo plano que contém s_i, o centro da híper-esfera N_i, s_j, o centro da híper-esfera N_j e o elemento central de uma consulta. Esses cálculos dividem-se em dois casos específicos, dependentes de $minDist(N_j, s_i)$ ser igual a zero ou diferente de zero. No caso quando $minDist(N_j, s_i) = 0$, encontram-se dois pontos de tangência imaginários ($s_{i_{t1}}$ e $s_{i_{t2}}$) sobre N_i em relação ao objeto s_q e utilizam-se estes dois pontos como auxiliares para a detecção de pontos imaginários e representativos para N_i, de forma que seja possível traçar os hiperplanos e verificar as podas. No segundo caso, quando $minDist(N_j, s_i) > 0$, encontram-se dois pontos de tangência imaginários ($s_{j_{t1}}$ e $s_{j_{t2}}$) sobre N_j em relação ao objeto s_i e utilizam-se estes pontos como auxiliares para a detecção de pontos imaginários e representativos para N_i, de forma que seja possível traçar os hiperplanos e verificar as podas. Observa-se que em ambos os casos os dois pontos são imaginários e portanto não é possível aplicar uma função de distância sobre eles. Porém é possível determinar as várias distâncias entre os elementos baseados na geometria métrica e no conhecimento das distâncias $d(s_q, s_i)$, $d(s_q, s_j)$ e $d(s_i, s_j)$ e dos raios ξ_i e ξ_j. A figura 4.6 contém uma ilustração dos pontos de tangência ($s_{j_{t1}}$ e $s_{j_{t2}}$) sobre N_j em relação a s_i.

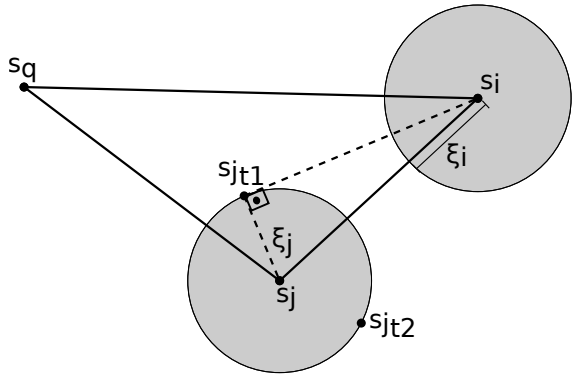


Figura 4.6: Exemplo de como encontrar os pontos de tangência

Inicia-se agora a apresentação dos cálculos para a definição da distância ao hiperplano quando N_i é uma entrada de nó índice. Pode-se dividir esses cálculos em 3 grupos:

1. Quando $d(s_q, s_j) > d(s_i, s_j)$ e $minDist(N_j, s_i) = 0$, o cálculo de $d_H(s_q, N_i; N_j)$ é definido segundo o lema 4.1. A figura 4.7 contém uma representação geométrica bidimensional do cálculo da distância ao hiperplano segundo o lema 4.1. Observa-se que o ponto s_p é externo a N_i, porém ele é o representativo de N_i, o que permite realizar uma poda exata.

   Lema 4.1 *Seja s_q um elemento de consulta, N_j um elemento que é uma entrada de nó índice ou um elemento e N_i uma entrada de nó índice pertencente ao conjunto de elementos e entradas de nós índices mantidos em V. Se $d(s_q, s_j) > d(s_i, s_j)$ e $minDist(N_j, s_i) = 0$, então é possível definir um ponto imaginário s_p que será representativo para N_i e a distância deste ponto ao hiperplano são definidos da seguinte maneira:*

$$w = \sqrt{d(s_q, s_i)^2 - \xi_i^2} \tag{4.4}$$

$$d(s_q, s_p) = w + \xi_i \tag{4.5}$$

$$d(s_i, s_p) = \sqrt{2 \times \xi_i^2} \tag{4.6}$$

$$height_1 = height(s_q, s_i, s_j) \tag{4.7}$$

$$height_2 = height(s_q, s_p, s_i) \tag{4.8}$$

$$c_1 = \sqrt{d(s_i, s_j)^2 - height_1^2} \tag{4.9}$$

$$c_2 = \sqrt{d(s_i, s_p)^2 - height_2^2} \tag{4.10}$$

$$d(s_j, s_p) = \sqrt{(c_1 + c_2)^2 + (height_1 + height_2)^2} \tag{4.11}$$

$$d_H(s_q, N_i; N_j) = \frac{d(s_q, s_j)^2 - d(s_j, s_p)^2}{2 \times d(s_q, s_p)} \tag{4.12}$$

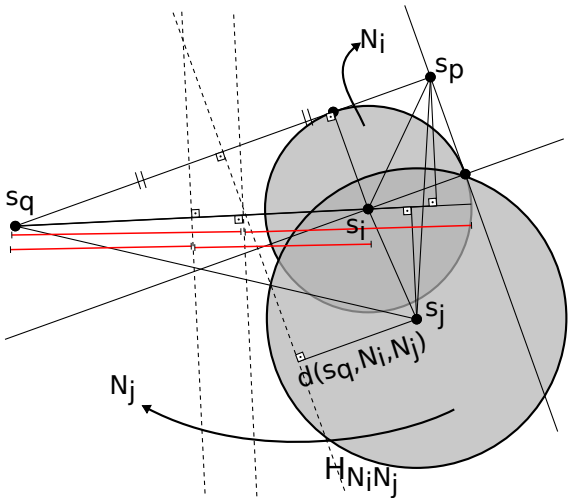


Figura 4.7: Ilustração do lema 4.1

2. Quando $d(s_q, s_j) > d(s_i, s_j)$ e $minDist(N_j, s_i) > 0$, o cálculo de $d_H(s_q, N_i; N_j)$ é definido segundo o lema 4.2. A figura 4.8 ilustra esse segundo caso para o cálculo da distância ao hiperplano relativo ao elemento imáginário s_p.

Lema 4.2 *Seja s_q um elemento de consulta, N_j uma entrada de nó índice ou elemento de um nó folha e N_i uma entrada de nó índice pertencente ao conjunto de elementos e entradas de nós índices mantidos em V. Se $d(s_q, s_j) > d(s_i, s_j)$ e $minDist(N_j, s_i) > 0$, é possível definir um ponto imaginário s_p que será represen-*

tativo para N_i e os cálculos que definem este ponto e a distância ao hiperplano são definidos conforme dado a seguir:

$$
\begin{aligned}
w &= \sqrt{d(s_i,s_j)^2 - \xi_j^2} && (4.13)\\
\frac{c.a._1}{hip_1} &= \frac{c.a._2}{hip_2} \\
d(s_i,s_p) &= \frac{\xi_i \times d(s_i,s_j)}{w} && (4.14)\\
d(s_j,s_p) &= \sqrt{(w + d(s_i,s_p))^2 + \xi_j^2} && (4.15)\\
height_1 &= height(s_q,s_i,s_j) && (4.16)\\
c_1 &= \sqrt{d(s_i,s_j)^2 - height_1^2} && (4.17)\\
\sin(\alpha-\beta) &= \sin(\alpha)\cos(\beta) - \cos(\alpha)\sin(\beta) \\
\sin(\alpha-\beta) &= \frac{height_1}{d(s_i,s_j)}\frac{w}{d(s_i,s_j)} - \frac{c_1}{d(s_i,s_j)}\frac{\xi_j}{d(s_i,s_j)} && (4.18)\\
height_2 &= \sin(\alpha-\beta) \times d(s_i,s_p) && (4.19)\\
\cos(\alpha-\beta) &= \cos(\alpha)\cos(\beta) + \sin(\alpha)\sin(\beta) \\
\cos(\alpha-\beta) &= \frac{c_1}{d(s_i,s_j)}\frac{w}{d(s_i,s_j)} + \frac{height_1}{d(s_i,s_j)}\frac{\xi_j}{d(s_i,s_j)} && (4.20)\\
c_2 &= \cos(\alpha-\beta) \times d(s_i,s_p) && (4.21)\\
d(s_q,s_p) &= \sqrt{(d(s_q,s_i) + c_2)^2 + (height_2)^2} && (4.22)\\
d_H(s_q,N_i;N_j) &= \frac{d(s_q,s_j)^2 - d(s_j,s_p)^2}{2 \times d(s_q,s_p)} && (4.23)
\end{aligned}
$$

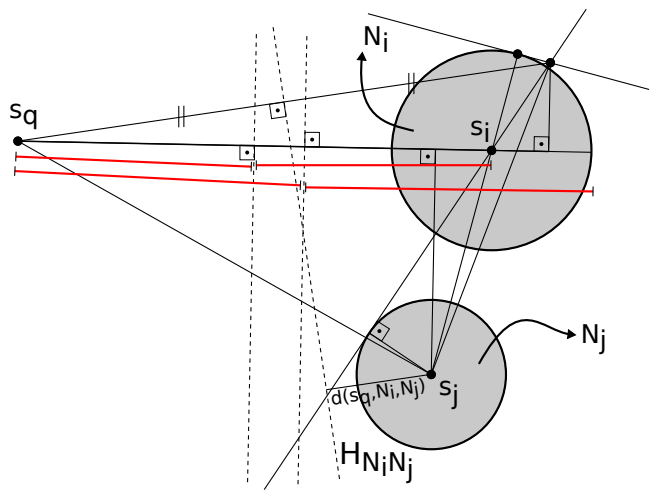


Figura 4.8: Ilustração do lema 4.2

3. Quando $d(s_q,s_j) \leq d(s_i,s_j)$ e $minDist(N_j,s_i) > 0$, o cálculo de $d_H(s_q,N_i;N_j)$ é definido segundo o lema 4.3. A figura 4.9 ilustra esse caso.

Lema 4.3 *Seja s_q um elemento de consulta, N_j uma entrada de nó índice ou elemento e N_i uma entrada de nó índice pertencente ao conjunto de elementos e entradas de nós índices mantidos em V. Se $d(s_q, s_j) \leq d(s_i, s_j)$ e $minDist(N_j, s_i) > 0$, é possível definir um ponto imaginário s_p que será representativo para N_i cuja distância ao hiperplano pode ser definida conforme dado a seguir:*

$$
\begin{aligned}
w &= \sqrt{d(s_i, s_j)^2 - \xi_j^2} && (4.24)\\
\frac{c.a._1}{hip_1} &= \frac{c.a._2}{hip_2} \\
d(s_i, s_p) &= \frac{\xi_i \times d(s_i, s_j)}{w} && (4.25)\\
d(s_j, s_p) &= \sqrt{(w - d(s_i, s_p))^2 + \xi_j^2} && (4.26)\\
height_1 &= height(s_q, s_i, s_j) && (4.27)\\
c_1 &= \sqrt{d(s_i, s_j)^2 - height_1^2} && (4.28)\\
\sin(\alpha + \beta) &= \sin(\alpha)\cos(\beta) + \cos(\alpha)\sin(\beta) \\
\sin(\alpha + \beta) &= \frac{height_1}{d(s_i, s_j)}\frac{w}{d(s_i, s_j)} + \frac{c_1}{d(s_i, s_j)}\frac{\xi_j}{d(s_i, s_j)} && (4.29)\\
height_2 &= \sin(\alpha + \beta) \times d(s_i, s_p) && (4.30)\\
\cos(\alpha + \beta) &= \cos(\alpha)\cos(\beta) - \sin(\alpha)\sin(\beta) \\
\cos(\alpha + \beta) &= \frac{c_1}{d(s_i, s_j)}\frac{w}{d(s_i, s_j)} - \frac{height_1}{d(s_i, s_j)}\frac{\xi_j}{d(s_i, s_j)} && (4.31)\\
c_2 &= \cos(\alpha + \beta) \times d(s_i, s_p) && (4.32)\\
d(s_q, s_p) &= \sqrt{(d(s_q, s_i) - c_2)^2 + (height_2)^2} && (4.33)\\
d_H(s_q, N_i; N_j) &= \frac{d(s_q, s_j)^2 - d(s_j, s_p)^2}{2 \times d(s_q, s_p)} && (4.34)
\end{aligned}
$$

A estimativa de min_κ é realizada ao longo do percurso para o nó índice e utiliza o cálculo de distância ao hiperplano definido acima para podar elementos e subárvores. O estado do percurso na árvore que está sendo examinada é representado por um conjunto de nós índices e elementos, que aqui é denotado por V. Os elementos e nós índices em V constituem o conjunto de dados que não estão aninhados. No estado inicial, V contém apenas o nó raiz da árvore. Conforme a algoritmo percorre e expande os nós índices, V agrega os conjuntos de elementos e nós índices que provêm conhecimento mais preciso quanto à distribuição dos dados no espaço.

O valor min_κ de um dado um elemento s_j indica o número mínimo de elementos que podem estar mais próximos dele próprio do que do elemento de consulta s_q, baseando-se no conhecimento que pode ser inferido do estado atual do conjunto V. A figura 4.10 exemplifica como o valor min_κ de s_j é estimado. Centrada em s_j, a bola com raio igual a distância ao elemento de consulta contém alguns elementos e entradas de nó índices

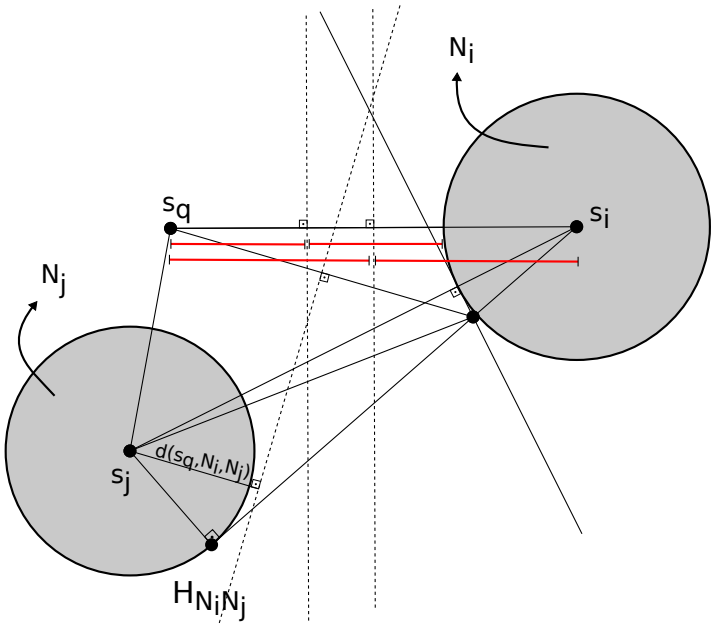


Figura 4.9: Ilustração do lema 4.3

(subárvores). Primeiramente, garante-se que o elemento s_1, interno à bola, esteja mais próximo de s_j do que de s_q, e, assim sendo, ele é contado junto ao min_κ de s_j. De maneira similar, a subárvore N_4 está completamente interna à bola centrada em s_j com raio $d(s_j, s_q)$, o que significa que todos os elementos internos a N_4 são definitivamente mais próximo de s_j do que de s_q. Por isso, N_4 contribui com quantidade de elementos na subárvore, correspondente a $N_4.numElem$. Contudo, N_1 e N_2 são apenas parcialmente cobertos pela bola centrada em s_j. Com $minmaxdist$, pode-se determinar qual subárvore pode contribuir em ao menos 1 para min_κ de s_j, pois pode-se confirmar se ao menos o elemento representante do nó está dentro da bola. Por fim, a subárvore N_3 está completamente fora da bola $cir(s_j, s_q)$ e, consequentemente, não contribui para o min_k de s_j. O resultado de todas essas avaliações faz com que min_κ neste momento seja estimado como $1 + 1 + N_4.numElem + 1$. Note que é necessário adicionar 1 em min_k para representar s_q.

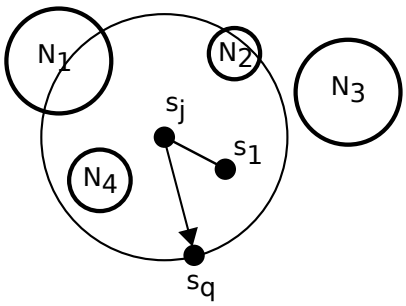


Figura 4.10: Exemplo de min_κ

Por isso, baseado no conjunto V (o estado atual de um espaço explorado em termos de elementos e entradas de nós índice), a seguinte expressão computa o min_κ de um elemento s_j e o lema 4.4 define a condição em que min_κ pode ser finalizado e converge para k:

$$min_\kappa(s_q, s_j, V) = 1 + \sum_{N_i \in V} count(s_j, N_i) \tag{4.35}$$

onde

$$count(s_j, N_i) = \begin{cases} 1 & \text{se } N_i \text{ é elemento } \wedge d_H(s_q, N_i; s_j) > 0 \\ & \vee \\ & \text{se } N_i \text{ é uma entrada de nó índice } \wedge \\ & d(s_q, s_j) > d(s_i, s_j) \wedge \\ & d_H(s_q, N_i; s_j) \leq 0 \\ N_i.numElem & \text{se } N_i \text{ é uma entrada de nó índice } \wedge \\ & d(s_q, s_j) > d(s_i, s_j) \wedge \\ & d_H(s_q, N_i; s_j) > 0 \\ 0 & \text{caso contrário.} \end{cases} \tag{4.36}$$

Lema 4.4 *Dado um elemento s_j, um elemento de consulta s_q e um conjunto de elementos e entradas de nós índices mantidos em V, o valor $min_\kappa(s_j, s_q, V)$ é igual a κ_{s_j} se não há entradas de nós índices $N_i \in V$ que são semi-cobertos pela bola centrada em s_j e com raio igual a ds_js_q, isto é,*

$$\forall_{N_i \in V | mindist(s_j, N_i) \leq dist(s_j, s_q)} maxdist(s_j, N_i) \leq d(s_j, s_q)$$

Dada uma entrada de nó índice N_j, o valor min_κ de N_j indica o mínimo número possível de elementos que são mais próximos de todos os elementos interno a N_j do que de s_q, inclusive se esses elementos são internos a N_j. Comparado com o caso de um elemento, o calculo de min_κ para um nó índice é mais complicado. Assim a posição exata do elementos interno a um nó índice são desconhecidos, mas como sua distância é limitada pelo raio de cobertura da subárvore, pode-se estimar min_k baseado em geometria métrica nas subárvores da *Slim-tree* da seguinte maneira:

$$min_\kappa(s_q, N_j, V) = 1 + \sum_{s_i \in V} count(N_j, N_i) \tag{4.37}$$

onde

$$count(N_j, N_i) = \begin{cases} 1 & \text{se } N_i \text{ é elemento } \wedge d_H(s_q, s_i; N_j) > \xi_j \\ & \vee \\ & \text{se } N_i \text{ é uma entrada de nó índice } \wedge d(s_q, s_i) > d(s_i, s_j) \wedge \\ & d_H(s_q, N_i; N_j) \leq \xi_j \wedge d_H(s_q, s_i; N_j) > \xi_j \\ N_i.numElem & \text{se } N_i \text{ é uma entrada de nó índice } \wedge d(s_q, s_i) > d(s_i, s_j) \wedge \\ & d_H(s_q, N_i; N_j) > \xi_j \\ 0 & \text{caso contrário.} \end{cases} \tag{4.38}$$

Por fim, há quatro condições possíveis para que N_i possa contribuir para o $min_\kappa(s_q, N_j, V)$, descritas a seguir.

1. O elemento ou a entrada de nó índice N_i contribui totalmente para a contagem de N_j e pode ser podado quando:

$$N_j \text{ é um elemento } \wedge N_i \text{ é um elemento } \wedge d(s_q, s_i) > d(s_i, s_j)$$
$$\vee$$
$$N_j \text{ é um elemento} \wedge N_i \text{ é uma entrada de nó índice } \wedge d(s_q, s_i) > d(s_i, s_j) \wedge$$
$$d_H(s_q, N_i; s_j) > 0 \wedge$$
$$\vee$$
$$N_j \text{ é uma entrada de nó índice} \wedge N_i \text{ é elemento } \wedge d_H(s_q, s_i; N_j) > \xi_j$$
$$\vee$$
$$N_j \text{ é uma entrada de nó índice} \wedge N_i \text{ é uma entrada de nó índice} \wedge d(s_q, s_i) > d(s_i, s_j)$$
$$\wedge d_H(s_q, N_i; N_j) > \xi_j$$

2. O elemento ou a entrada de nó índice N_i contribui em 1 para a contagem de N_j, mas não pode ser podado e deve ser melhor examinado quando:

$$N_j \text{ é um elemento} \wedge N_i \text{ é uma entrada de nó índice } \wedge d(s_q, s_i) > d(s_i, s_j) \wedge$$
$$d_H(s_q, N_i; N_j) \leq \xi_j \wedge d_H(s_q, s_i; N_j) > \xi_j$$
$$\vee$$
$$N_j \text{ é uma entrada de nó índice} \wedge N_i \text{ é uma entrada de nó índice } \wedge d(s_q, s_i) > d(s_i, s_j) \wedge$$
$$d_H(s_q, N_i; N_j) \leq \xi_j \wedge d_H(s_q, s_i; N_j) > \xi_j$$

3. O elemento ou a entrada de nó índice N_i não contribui totalmente para a contagem de N_j e pode ser podado quando:

$$N_j \text{ é um elemento} \wedge N_i \text{ é elemento } \wedge d(s_q, s_i) \leq d(s_i, s_j) \wedge \\ minDist(s_i, N_j) > 0 \\ \vee \\ N_j \text{ é um elemento} \wedge N_i \text{ é uma entrada de nó índice} \wedge d(s_q, s_i) \leq d(s_i, s_j) \wedge \\ minDist(s_i, N_j) > 0 \wedge d_H(s_q, N_i; s_j) < 0 \\ \vee \\ N_j \text{ é uma entrada de nó índice} \wedge N_i \text{ é um elemento} \wedge d(s_q, s_i) \leq d(s_i, s_j) \wedge \\ minDist(s_i, N_j) > 0 \wedge d_H(s_q, s_i; N_j) < -\xi_j \\ \vee \\ N_j \text{ é uma entrada de nó índice} \wedge N_i \text{ é uma entrada de nó índice} \wedge d(s_q, s_i) \leq d(s_i, s_j) \wedge \\ minDist(s_i, N_j) > 0 \wedge d_H(s_q, N_i; N_j) < -\xi_j$$

4. O elemento ou a entrada de nó índice N_i não contribui para a contagem de N_j, mas não pode ser podado e deve ser melhor examinado quando não obedece a nenhum dos outros casos.

4.4 Algoritmo $RkNN$-MG

Nesta seção apresenta-se o algoritmo $RkNN$-MG, que é baseado na noção de min_k, uma estimativa de limite mínimo para o número de vizinhos k para um elemento. Este algoritmo é uma versão adaptada do algoritmo *k-browsing* para consultas $RkNN$ (veja seção 3.8). No entanto, a noção de min_k apresentada para o algoritmo *k-browing* é diferente daquela utilizada neste trabalho, como apresentado na seção anterior.

Embora o algoritmo *k-browsing* seja adotado para realizar nossos experimentos, todas as estratégias de poda propostas neste trabalho podem ser adaptadas para outros algoritmos existentes na literatura, como o recente algoritmo AKKRZ (veja seção 3.9).

O pseudocódigo do algoritmo $RkNN$ é mostrado no Algoritmo 4.1 e utiliza o algoritmo 4.2 para retirar elementos da fila de prioridade. O algoritmo usa a fila de prioridade P para deixar elementos e entradas de nós índices não examinados de acordo com a ordem decrescente de seus min_ks. Em caso de empate em que dois ou mais elementos/entradas de nós índices tenham o mesmo min_k, aquele com o menor *mindist* é tratado primeiro. Além disso, um conjunto V é mantido para capturar o conhecimento atual da distribuição dos elementos $s_i \in P$ via um conjunto de elementos e entradas de nó índice, baseado no min_k estimado de cada entrada $s_i \in P$.

O algoritmo $RkNN$-MG, descrito no algoritmo 4.1, sempre retira de V a primeira entrada até que o min_k desta entrada seja maior que k. Todos os elementos e nós índices

Algoritmo 4.1: FindRknn

Input: Um elemento de consulta (s_q), a raiz de $S(root)$, o número solicitado de k vizinhos mais próximos reversos
Data: Uma fila priorizada (P), um conjunto de nós índice(V), um indicativo de parada (continue)
Result: Os elementos da resposta $rknn$

```
1  V ← root;
2  enqueue (root, min_k(q, root, V)) to P;
3  continue = true;
4  while P is not empty and continue = true do
5      (N, min_k) ← DequeueWithUpdate(P, q, V);
6      if N is an index node then
7          suppose N has n children N_i(i ∈ [1, n]);
8          V ← V − {N} ∪ (∪_{i=1}^{n} {N_i});
9          foreach N_i(1 ≤ i ≤ n) do
10             enqueue (N_i, min_k(q, N_i, V)) to P;
11         end
12     else
13         if ∃N_a partially covered by cir(N, s_q) then
14             suppose N_a has n children N_i(i ∈ [1, n]);
15             V ← V − {N_a} ∪ (∪_{i=1}^{n} {N_i});
16             remove N_a from P;
17             foreach N_i(1 ≤ i ≤ n) do
18                 enqueue (N_i, min_k(q, N_i, V)) to P;
19             end
20             enqueue (N, min_k(q, N, V)) to P;
21         else
22             if min_k ≤ k then
23                 output (N, min_k);
24             else
25                 continue = false;
26             end
27         end
28     end
29 end
```

em V (exceto o próprio elemento s_j) contribuem para o min_k do elemento s_j. Por isso, o min_k de um elemento s_j baseado no conjunto V pode ser representado por $min_k(s_j, s_q, V - \{s_j\})$. Adicionalmente, o min_k de uma entrada de nó índice N_j baseado no conjunto V é representado por $min_k(N_j, s_q, V - \{N_j\}) + sc(s_q, N_j)$, onde $sc(s_q, N_j)$ computa o número de elementos $s_{j'}$ internos a N_j que são mais próximos do elemento s_j do que de s_q, com $s_{j'}, s_j \in N_j$, e $s_{j'} \neq s_j$. Especificamente,

Algoritmo 4.2: DequeueWithUpdate

Input: A fila de prioridade (P), o centro da consulta (q), uma coleção de nós índice e elementos de dados (V)

Output: A tupla (N, k) com o menor valor de k estimado;

```
1  while true do
2      (N, k) ← dequeue(P);
3      if N is a data point then
4          k ← min_k(q, N, V − N);
5      else
6          k ← min_k(q, N, V − N) + sc(q, N);
7      end
8      (N′, k′) ← head(P);
9      if k ≤ k′ then
10         return (N, k);
11     end
12     enqueue (N, k) to P;
13 end
```

$$sc(N_j, s_q, k) = \begin{cases} N_j.cnt - 1 & \text{if } k = 1 \wedge \\ & dist_{s_q s_j} > 2 * \xi_j \\ & \text{or} \\ & \text{if } k > 1 \wedge \\ & dist_{s_q s_j} > 3 * \xi_j \\ 0 & \text{otherwise.} \end{cases} \tag{4.39}$$

A execução do algoritmo *RkNN-MG* funciona inicialmente colocando a raiz na fila com prioridade P com seu min_k associado. Assim, o conjunto V sempre inicia com a raiz como seu conteúdo. Com o ínicio da execução, enquanto verifica-se que a primeira entrada da fila contém um $min_k \leq k$, o algoritmo retira esta entrada da fila com seu respectivo min_k chamando o algoritmo *DequeueWithUpdate* presente no algoritmo 4.2. Este algoritmo 4.2 fica atualizando o min_k da primeira entrada da fila enquanto não é garantido que essa primeira entrada da fila seja priorizada para o teste. Essa priorização é baseada na garantia de que essa entrada tem o menor min_k dentre todas as entradas da fila. Retornada a entrada com menor min_k em P, o algoritmo 4.1 faz uma verificação se essa entrada da fila é uma entrada de nó índice ou um elemento. Caso seja um entrada de nó índice, ele é expandido e são calculados os min_ks dos filhos, sendo estes filhos inseridos na fila de prioridade P. Caso a entrada da fila seja um elemento, verifica-se se não há nenhuma entrada de nó índice interno parcialmente coberto pela bola centrada no elemento de verificação e com raio igual a distância dele ao elemento de consulta. Essa verificação confirma que não há nenhuma entrada de nó índice que possa incrementar o valor de min_k e, se $min_k \leq k$, este elemento é um *RkNN*. Contudo, havendo uma entrada

de nó índice que pode influênciar o elemento, esse entrada de nó índice é expandido e seus filhos junto com o elemento N são inseridos na fila P.

4.5 Considerações finais

Este capítulo detalhou as novas propostas de podas que utilizam a lei dos cossenos e pontos de tangência e detalhou o algoritmo $RkNN$-MG proposto, que utiliza essas podas para realizar consultas $RkNN$ exatas de maneira eficiente. Inicialmente, o capítulo apresentou uma visão geral do algoritmo proposto, destacando o conceito min_κ. Em seguida, foi detalhado o cálculo de distância ao hiperplano utilizando a geometria métrica para conseguir resultados mais precisos, o conceito de min_κ definindo os critérios de poda que a abordagem proposta permite. No capítulo seguinte são apresentados os testes de desempenho realizados, os quais mostra resultados bem melhores quando comparados com as propostas existentes que utilizam apenas a desigualdade triangular para realizar podas.

Capítulo

5

Experimentos

Para avaliar o algoritmo *RkNN-MG* e os novos critérios de poda baseados na geométrica dos espaços métricos propostos, o algoritmo foi codificado em linguagem C++, bem como os algoritmos de k-browsing (veja-se a seção 3.8) e Tao et al (veja seção 3.7.1) propostos na literatura, para que pudesse ser feita uma comparação dos três em um mesmo ambiente de execução, permitindo que a comparação pudesse ser a mais justa possível. Os dois algoritmos existentes na literatura são os que apresentam o melhor desempenho em suas respectivas classes de aplicação.

5.1 Conceituação inicial

O algoritmo *RkNN-MG* foi avaliado por meio de testes de desempenho utilizando ampla variedade de conjuntos reais e sintéticos. A tabela 5.1 apresenta os conjuntos de dados mais significativos utilizados nos experimentos. Para cada conjunto é indicado: o nome, o número total de elementos indexados, a dimensão, a métrica de comparação e a descrição de sua origem.

O algoritmo *RkNN-MG* proposto é comparado com uma adaptação da proposta k-browsing (veja-se a seção 3.8) para realizar consultas *RkNN*. Além disso, no primeiro experimento é feita uma comparação com o algoritmo de Tao et al (veja seção 3.7.1). A comparação do algoritmo proposto com esses dois possibilitam validar a abordagem proposta neste trabalho, visto que aqueles são dois algoritmos muito recentes e que apresentam bons desempenhos. A abordagem AKKRZ (veja seção 3.9), que é a primeira solução para a consulta *RkNN* em espaços métricos sem restrição, não é utilizada nas

Tabela 5.1: Conjunto de dados utilizados nos experimentos

Conjunto de Dados	Nro. Objetos	Dimensão	Métrica	Descrição
EigenFaces	11900	16	L_1	Projeto Informedia da Carnegie Mellon University (Wactlar et al, 1996).
Cidades Americanas	25375	2	L_2	Latitude e Longitude de Cidades Americanas. url: `http://www.census.gov/`
Currency	2311	6	L_1	Cotações de moedas em relação ao Dólar Canadense. As 2561 cotações são dos dias úteis de 01/01/1987 a 28/01/1997, das moedas: *Hong Kong Dollar*, *Japanese Yen*, *American Dollar*, *German Mark*, *French Franc* e *British Pound*.
Faces (SISAP)	762	761	L_1	Conjunto *Faces* do *International Workshop on Similarity Search and Applications (SISAP)* url: `http://www.sisap.org/library/dbs/faces/`
Histogramas de Cor	68040	32	L_2	Histogramas de Cor de Imagens do repositório do KDD na Universidade da California (`http://kdd.ics.uci.edu`).
EnglishWords	24893	adimensional	L_{edit}	Palavras de um dicionário da língua inglesa.

comparações por consistir das mesmas regras de podas utilizadas no algoritmo k-browsing, diferindo apenas na estratégia de execução do algoritmo. De fato, é possível implementar as novas regras de podas na mesma estratégia adotada pelo algoritmo AKKRZ e assim realizar comparações diretas entre o abordagem AKKRZ e aquela proposta neste trabalho. No entanto, a complexidade para implementar a abordagem AKKRZ é grande, e os resultados seriam equivalentes aos obtidos com a comparação com a proposta k-browsing.

Os experimentos foram executados em um computador equipado com processador *Intel Core i7 920*, 6GB RAM de memória RAM e disco rígido de 1TB. O sistema operacional utilizado é Ubuntu 9.04 - versão *Desktop*. Os principais aspectos de configuração relacionados aos testes foram:

- Os elementos de cada conjunto de dados foram "embaralhados", simulando a situação real de inserções de objetos que ocorrem em Sistemas Gerenciadores de Bases de Dados (SGBDs) - inserções ocorrem de forma aleatória;

- Após o embaralhamento, extraiu-se uma amostra de elementos, correspondendo ao subconjunto de dados usados como centro da região de busca. Os elementos foram escolhidos aleatoriamente entre todos os pertencentes ao conjunto e o tamanho das amostras selecionadas foram de 500 objetos para os conjuntos de dados, exceto para

os conjuntos "Faces (SISAP)", "Color Histogram" e "EnglishWords", para os quais foram selecionados 254, 50 e 20 elementos para a amostra, respectivamente.

- Os dados foram indexados pelo método de acesso métrico *Slim-tree*, configurado para usar como algoritmo de divisão de nós o $minMax$ e como algoritmo de escolha de sub-árvore *ChosseSubtree* o algoritmo $minDist$. A taxa de ocupação mínima foi definida como 50%.
- Os valores de k escolhidos para encontrar os k-vizinhos mais próximos reversos foram 1, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30 e 32.
- O tamanho das páginas em disco foi definido em $4KB$, com exceção dos experimentos com os conjuntos de dados "Faces (SISAP)", "Color Histogram" e "EnglishWords", os quais utilizam páginas de 256KB, 16KB e 16KB, respectivamente.

5.2 Experimentos com o conjunto de dados Eigen Faces

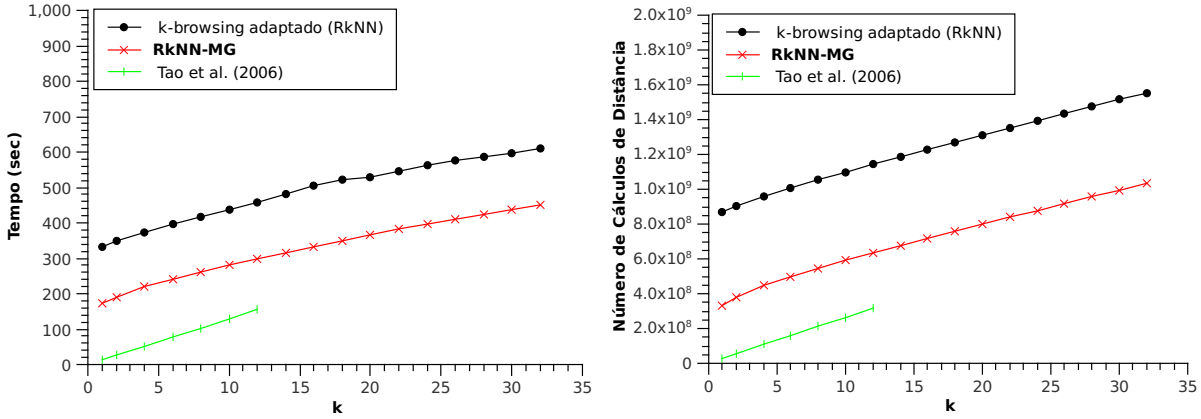
O experimento apresentado nesta seção foi executado utilizando o conjunto de dados *Eigen Faces*. As figuras 5.1(a) e 5.1(b) contêm respectivamente à comparação do tempo de execução e da quantidade de cálculos de distâncias executados pelos três algoritmos. Comparando a nova proposta com o algoritmo de Tao et al. (2006), verifica-se que, nosso algoritmo perde em desempenho, mas não tem a limitação no valor de k. De fato, o algoritmo de Tao et al. pode realizar consultas nesse conjunto de dados para um k máximo igual a 12, o que inviabiliza seu uso para as aplicações que requeiram k maior.

Comparado com a abordagem k-browsing adaptada, nossa proposta apresenta uma redução média em tempo de 34%. Quando $k = 1$, situação que é destacada em muitos artigos na literatura, essa redução é de 48%. A relação entre os cálculos de distância entre os dois algoritmo demonstra uma redução média de 44% no número de cálculos de distância necessários.

Como o objetivo do trabalho foi criar uma abordagem para a operação $RkNN$ que não tivesse limitações nos experimentos subsequentes o algoritmo de Tao et al. não é mais usado nas comparações.

5.3 Experimentos com o conjunto de dados Cidades Americanas

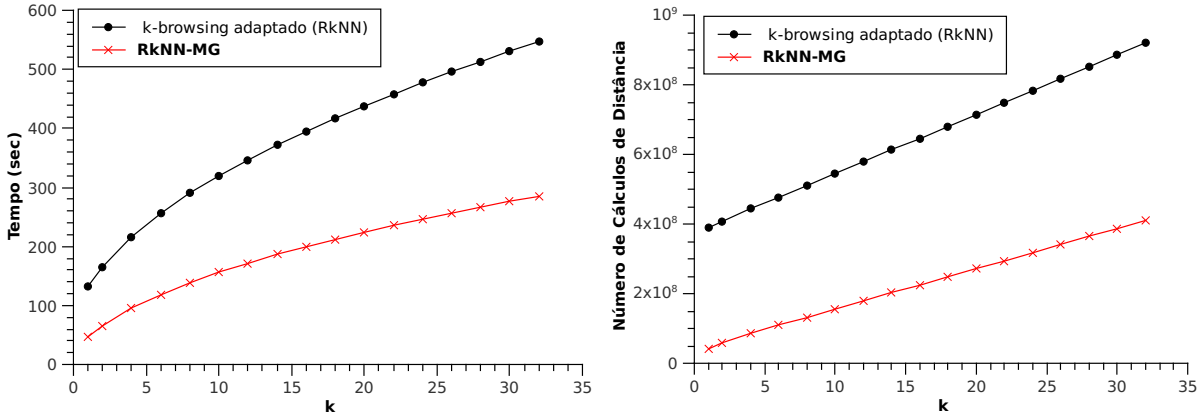
Para o conjunto de cidades americanas, o qual têm baixa dimensionalidade (apenas duas dimensões), o desempenho do algoritmo proposto foi ainda melhor do que sobre o conjunto *Eigen Faces*. A redução média do tempo de execução e do número de cálculos de distâncias foi de 51% e 68% respectivamente. Um ponto de destaque neste experimento é a redução



(a) (b)

Figura 5.1: Gráficos de desempenho do algoritmo *RkNN-MG* sobre o conjunto de dados *Eigen Faces*.

em 89% no número de cálculo de distâncias quando $k = 1$. Tudo isso é registrado na figura 5.2 que mostra os gráficos de tempo e quantidade cálculos de distância.

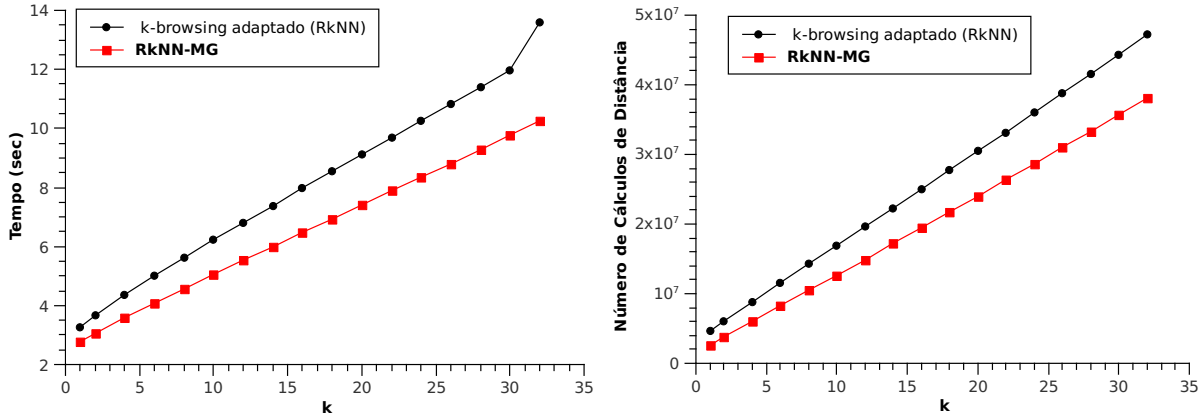


(a) (b)

Figura 5.2: Gráficos de desempenho do algoritmo *RkNN-MG* sobre o conjunto de dados *Cidades Americanas*

5.4 Experimentos com o conjunto de dados Currency

Outro conjunto interessante testado e que é de média dimensionalidade, é o conjunto *Currency*. Os gráficos da figura 5.3 apresentam também redução média no tempo e no cálculo de distâncias, porém menores do que dos dois conjuntos anteriores, ficando em de 19% e 25% respectivamente. Para $k = 1$, a redução na número de cálculos de distâncias é de 42% e a redução do tempo de execução é de 15%.

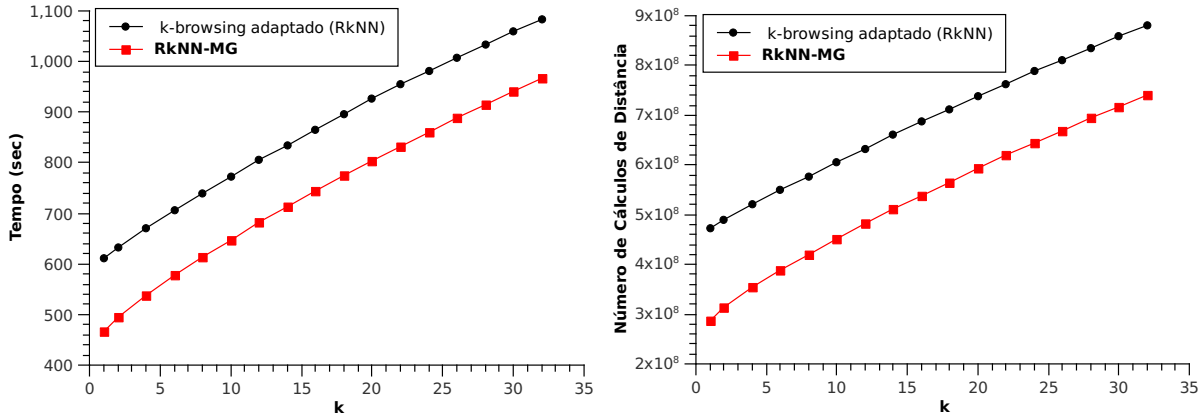


(a) (b)

Figura 5.3: Gráficos de desempenho do algoritmo *RkNN-MG* sobre o conjunto de dados *Currency*

5.5 Experimentos com o conjunto de dados Histograma

O próximo experimento apresentado foi executado utilizando o conjunto de dados *Histograma* e pode ser verificado na figura 5.4. Em média, a redução do tempo de execução e da quantidade de cálculos de distâncias são respectivamente de 15% e 24%. Para $k = 1$, a redução do tempo de execução é de 23% e a redução do número de cálculos de distâncias é de 39%.

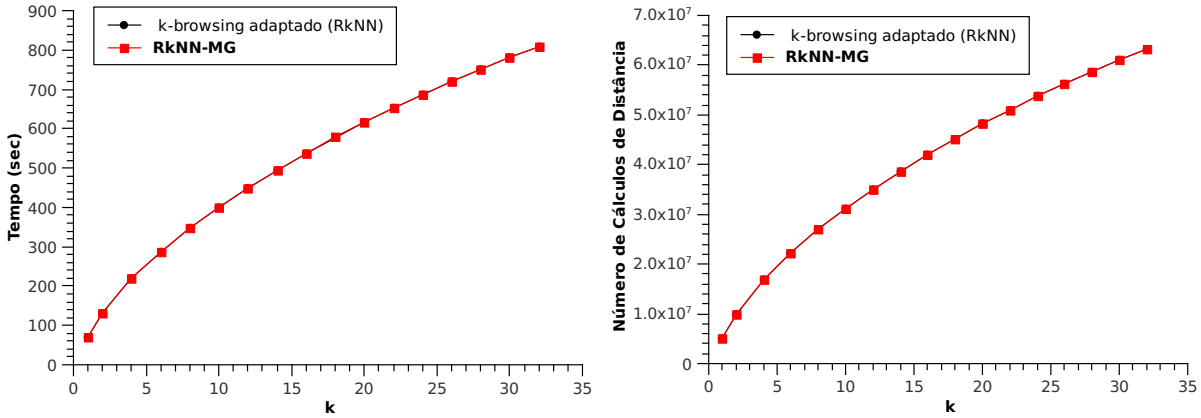


(a) (b)

Figura 5.4: Gráficos de desempenho do algoritmo *RkNN-MG* sobre o conjunto de dados *Color Histrogram*.

5.6 Experimentos com o conjunto de dados Faces (SISAP)

O conjunto *Faces (SISAP)* é um conjunto interessante por sua característica de alta dimensionalidade. Os experimentos realizados são mostrados nas figuras 5.5(a) e 5.5(b). Como se pode ver, a alta dimensionalidade degrada as possibilidades de poda tanto para o algoritmo que utiliza apenas desigualdade triangular quanto para o algoritmo proposto, levando ao empate das duas propostas. Deve-se ressaltar no entanto, que ambos apresentam resultados exatos, mesmo em situações de alta dimensionalidade.



(a) (b)

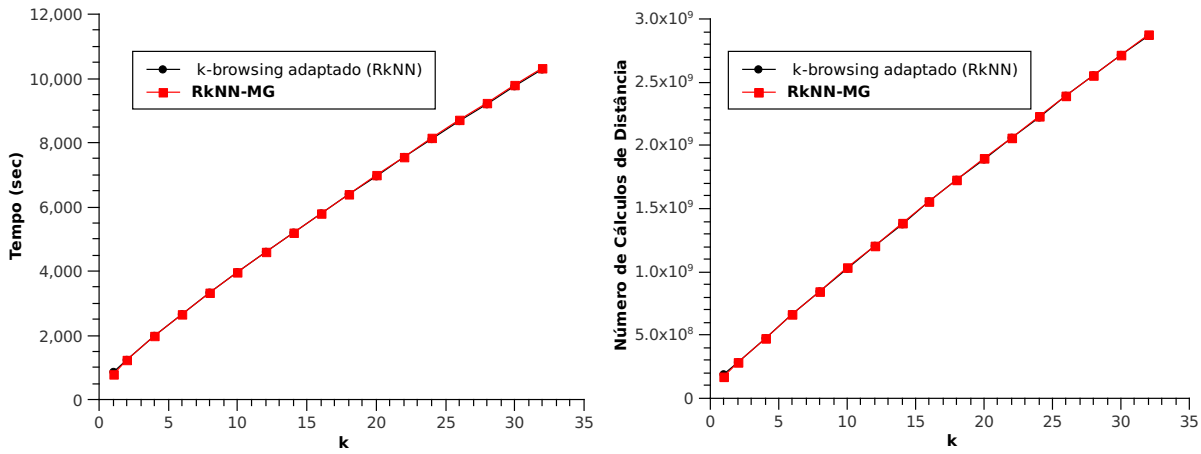
Figura 5.5: Gráficos de desempenho do algoritmo *RkNN-MG* sobre o conjunto de dados *Faces (SISAP)*

5.7 Experimentos com o conjunto de dados EnglishWords

Os experimentos presentes na figura 5.6 contêm resultados do interessante conjunto *EnglishWords* que é um conjunto métrico. A dimensão fractal deste conjunto é bastante elevada e por isso, como se pode ver nas figuras, os resultados para ambos os algoritmos são equivalentes. Confirmando a tendência já apresentada na seção anterior.

5.8 Teste de escalabilidade

Para verificar a escalabilidade do algoritmo proposto, foram criados conjuntos de dados sintéticos utilizando a ferramenta DBGen (Bueno et al, 2004) e realizados experimentos comparativos com o algoritmo k-browsing adaptado. Basicamente, são dois grupos de



(a) (b)

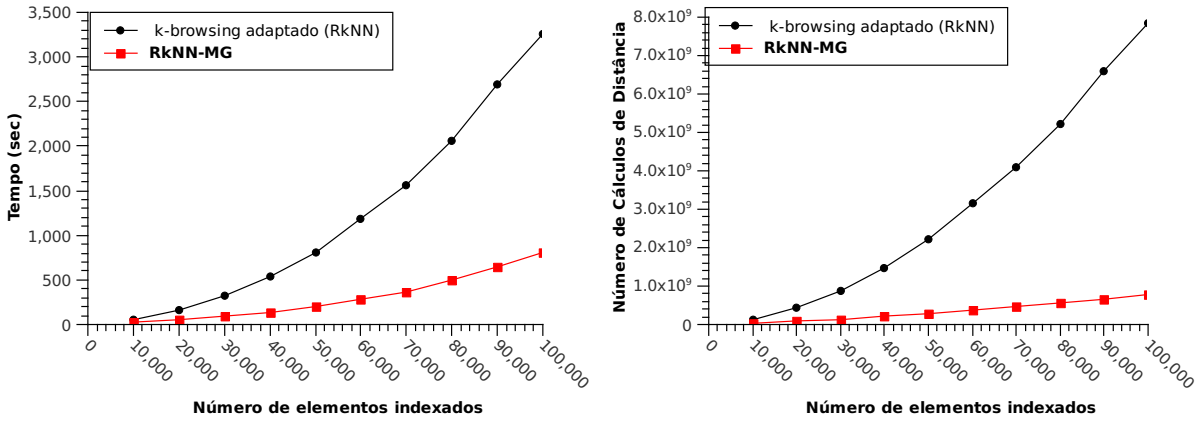
Figura 5.6: Gráficos de desempenho do algoritmo *RkNN-MG* sobre o conjunto de dados *EnglishWords*

conjuntos de dados: o primeiro grupo (SC) são conjuntos de dados gerados utilizando distribuição Gaussiana com geração de 5 agrupamentos em um conjunto de dados de 4 dimensões; o segundo grupo (SU) de conjuntos de dados foram gerados utilizando distribuição Uniforme em um conjunto de dados de 4 dimensões. Para ambos os grupos, foram criados conjuntos de dados de 10 mil a 100 mil elementos incrementados de 10 mil em 10 mil.

As figuras 5.7 e 5.8 contém os resultados dos experimentos com os conjuntos de dados SC e SU, respectivamente. Observa-se nos gráficos a redução tanto em tempo de execução quanto no número de cálculos de distância. As tabelas 5.2 e 5.3 contém a porcentagem de redução do tempo de execução e do número de cálculos de distâncias, onde se verifica que o ganho em tempo de execução do algoritmo proposto foi de no mínimo 63%, e a redução do número de cálculos de distância foi de no mínimo 75%.

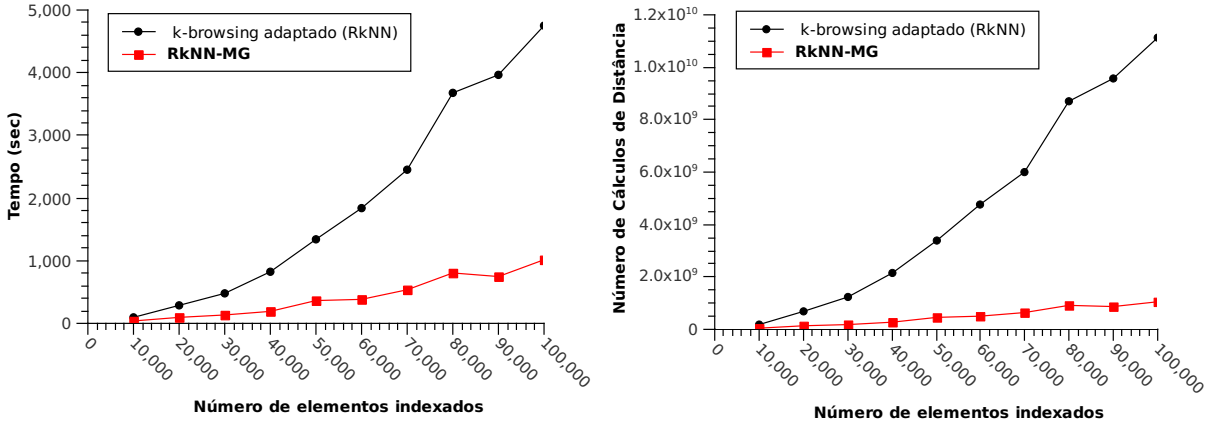
	Porcentagem de redução no	
	Tempo de execução	Número de cálculos de distância
10000	63%	75%
20000	70%	82%
30000	73%	84%
40000	74%	86%
50000	75%	87%
60000	77%	88%
70000	76%	89%
80000	76%	89%
90000	76%	90%
100000	75%	90%

Tabela 5.2: Porcentagem de redução do conjunto SC



(a) (b)

Figura 5.7: Gráficos de desempenho do algoritmo *RkNN-MG* sobre o conjunto de dados *SC*.



(a) (b)

Figura 5.8: Gráficos de desempenho do algoritmo *RkNN-MG* sobre o conjunto de dados *SU*.

5.9 Considerações Finais

Este capítulo definiu o ambiente de testes utilizado na avaliação do abordagem *RkNN-MG* e descreveu os resultados dos testes de desempenho realizados. Os resultados obtidos foram comparados com a abordagem *k*-browsing adaptada para executar consultas *RkNN*, a qual representa a mais recente das duas únicas soluções existentes anteriormente para a consulta *RkNN* sem restrições de execução no conjunto de dados. Nos testes, foram analisados o tempo de execução e o número de cálculos de distância da execução da consulta *RkNN*.

	Porcentagem de redução no	
	Tempo de execução	Número de cálculos de distância
10000	63%	77%
20000	66%	81%
30000	73%	85%
40000	77%	87%
50000	73%	86%
60000	79%	89%
70000	78%	90%
80000	78%	89%
90000	81%	91%
100000	79%	91%

Tabela 5.3: Porcentagem de redução do conjunto SU

Os testes indicaram que a abordagem proposta *RkNN-MG* foi sempre melhor, sendo tanto mais rápida na execução quanto reduzindo o número de cálculos de distância. Observa-se ainda com os experimentos que, mesmo em situações adversas como no caso de altas dimensionalidades, que o desempenho de ambas as abordagens se equivalem, mas nunca é inferior. Dessa forma, pode-se concluir que a abordagem *RkNN-MG* é uma técnica robusta, sem restrições com desempenho e superior às abordagens existentes que se utilizam apenas da desigualdade triangular.

Capítulo

6

Conclusões e trabalhos futuros

Esta dissertação apresenta a proposta de uma nova maneira de se realizar podas para um tipo de consulta por similaridade que tem recebido considerável atenção nos últimos anos (Korn e Muthukrishnan, 2000; Maheshwari et al, 2002; Yang e Lin, 2001; Tao et al, 2004, 2006; Achtert et al, 2006a,b; Lee et al, 2008; Achtert et al, 2009), a operação de consultas aos k-vizinhos mais próximos reversos. Essa consulta tem recebido atenção nos últimos tempos, devido a sua larga aplicação em *marketing*, sistemas de apoio a decisão, mineração de dados, etc. A consulta aos k-vizinhos mais próximos reversos obtém todos os elementos da base de dados que tem o elemento indicado na consulta como um de seus k-vizinhos mais próximos.

Essa dissertação propôs uma nova abordagem para procurar pela resposta exata de operações de consultas aos k-vizinhos mais próximos reversos em espaços métricos, que utiliza geometria métrica para realizar podas eficientes. Especificamente, a nova abordagem realiza consultas $RkNN$ para o caso monocromático, podendo ser usada indistintamente em conjuntos de dados estáticos e dinâmicos. Ela diferencia-se das abordagens propostas anteriormente na literatura por fornecer respostas exatas utilizando conceitos da geometria métrica para realizar podas com eficiência e por não ter restrições quanto a definição do parâmetro k da consulta.

Os experimentos realizados mostraram que o algoritmo $RkNN$-MG desenvolvido utilizando essa nova abordagem tem desempenho superior ao algoritmo que utiliza apenas a desigualdade triangular. A comparação do algoritmo $RkNN$-MG com o outro mais recente apresentado na literatura em experimentos com dados reais e sintéticos mostrou resultados que permitem assegurar que o algoritmo $RkNN$-MG não perde para as abor-

dagens que utilizam apenas podas baseadas em desigualdade triangular, chegando a obter redução no tempo em média superior a 40%.

Os experimentos mostraram que o algoritmo $RkNN$-MG proposto não é o mais rápido: o algoritmo proposto por Tao et al. (2006) pode dar respostas mais rapidamente, mas apenas para valores de k pequenos, e que é limitado pela maneira como a estrutura de indexação que dá suporte às buscas é construída. Dessa maneira, apesar de mais rápido, o algoritmo de Tao et al. não consegue responder a consultas $RkNN$com valores de k maior do que o limite (que pode ser tão pequeno quanto $k \leq 12$ para o conjunto de dados *Eigen Faces*), o que torna o algoritmo proposto o mais rápido para consultas $RkNN$quaisquer, ao mesmo tempo em que ele fornece sempre respostas exatas, seja para conjuntos de dados estáticos ou dinâmicos.

6.1 Principais contribuições

As principais contribuições decorrentes da realização deste trabalho são:

- Proposta de novas podas em espaços métricos que utilizam lei dos cossenos para melhorar a eficiência;
- Desenvolvimento de um novo algoritmo exato para responder às consultas aos $RkNN$s utilizando as novas podas propostas;
- Redução do número de cálculos de distância e do tempo de execução das consultas $RkNN$.

6.2 Trabalhos futuros

Várias extensões podem ser realizadas ao trabalho realizado nesta dissertação. Dentre essas extensões, pode-se incluir os tópicos abaixo como trabalhos futuros:

- Desenvolvimento de uma estratégia para execução da consultas $RkNN$ bicromáticas para conjuntos de dados em espaços métricos que utilizem as novas regras de podas propostas;
- Desenvolvimento de operadores de junção por similaridade, especialmente as junções k-NNJ e k-CNJ (as junções *k-nearest neighbor join* e *k-closest neighbor join*), que aproveitem as novas regras de podas desenvolvidas;
- Avaliar o impacto que as novas regras de poda podem causar quando aplicadas a espaços métricos que não sejam imersíveis em espaços euclidianos, e identificar os problemas e respectivas soluções para estender os podas baseadas em geometria métrica para esses espaços;

- Identificação das propriedades algébricas que vigoram em expressões que envolvem a operação de consulta aos k-vizinhos mais próximos reversos, tanto em expressões que utilizem apenas essa operação, quanto em expressões que utilizem outras formas de consulta por similaridade, como a consulta por abrangência e a consulta aos k-vizinhos mais próximos, bem como o desenvolvimento de algoritmos que possam executar as combinações mais frequentes dessas consultas;

- Criação de uma extensão à linguagem SQL para que a operação de consultas aos k-vizinhos mais próximos reversos possa ser expressa nessa linguagem.

Referências Bibliográficas

Achtert, E.; Böhm, C.; Kröger, P.; Kunath, P.; Pryakhin, A.; Renz, M. Approximate reverse k-nearest neighbor queries in general metric spaces. In: *CIKM '06: Proceedings of the 15th ACM international conference on Information and knowledge management*, New York, NY, USA: ACM, 2006a, p. 788–789.
Disponível em `http://doi.acm.org/10.1145/1183614.1183731`

Achtert, E.; Böhm, C.; Kröger, P.; Kunath, P.; Pryakhin, A.; Renz, M. Efficient reverse k-nearest neighbor search in arbitrary metric spaces. In: *SIGMOD '06: Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, New York, NY, USA: ACM Press, iSBN:1-59593-434-0. SESSION: Skyline and similarity search, 2006b, p. 515–526.
Disponível em `http://portal.acm.org/citation.cfm?id=1142473.1142531`

Achtert, E.; Böhm, C.; Kröger, P.; Kunath, P.; Pryakhin, A.; Renz, M. Efficient reverse k-nearest neighbor estimation. In: *Datenbanksysteme in Business, Technologie und Web (BTW 2007)*, GI, iSBN: 978-3-88579-197-3, 2007, p. 344–363 (*LNI*, v.103).
Disponível em `http://www.dbs.ifi.lmu.de/Publikationen/Papers/btw-rknn.pdf`

Achtert, E.; Kriegel, H.-P.; Kröger, P.; Renz, M.; Züfle, A. Reverse k-nearest neighbor search in dynamic and general metric databases. In: *12th Int. Conf. on Extending Database Technology (EDBT'09)*, Saint-Petersburg, Russia, to appear, 2009.

Baeza-Yates, R. A. Searching: an algorithmic tour. *Encyclopedia of Computer Science and Technology*, v. 37, p. 331–359, 1997.

Baeza-Yates, R. A.; Cunto, W.; Manber, U.; Wu, S. Proximity matching using fixed-queries trees. In: *CPM '94: Proceedings of the 5th Annual Symposium on*

Combinatorial Pattern Matching, Crochemore, Maxime and Gusfield, Dan, Asilomar, CA: Springer Verlag, 1994, p. 198–212 (*Lecture Notes in Computer Science*, v.807). Disponível em `http://portal.acm.org/citation.cfm?id=647814.738307`

Ban, T.; Kadobayashi, Y. On tighter inequalities for efficient similarity search in metric spaces. In: *IAENG International Journal of Computer Science*, v. 35, IAENG - International Association of Engineers, p. 392–402, 2008.

Beckmann, N.; Kriegel, H.-P.; Schneider, R.; Seeger, B. The r*-tree: an efficient and robust access method for points and rectangles. In: *SIGMOD '90: Proceedings of the 1990 ACM SIGMOD international conference on Management of data*, New York, NY, USA: ACM, iSSN:0163-5808, 1990, p. 322–331. Disponível em `http://portal.acm.org/citation.cfm?id=98741`

Berchtold, S.; Keim, D. A.; Kriegel, H.-P. The x-tree: An index structure for high-dimensional data. In: Vijayaraman, T. M.; Buchmann, A. P.; Mohan, C.; Sarda, N. L., eds. *Proceedings of the 22nd International Conference on Very Large Databases*, San Francisco, U.S.A.: Morgan Kaufmann, iSBN: 1-55860-382-4, 1996, p. 28–39. Disponível em `http://www.vldb.org/conf/1996/P028.PDF`

Beyer, K. S.; Goldstein, J.; Ramakrishnan, R.; Shaft, U. When is "nearest neighbor" meaningful? In: *ICDT '99: Proceeding of the 7th International Conference on Database Theory*, London, UK: Springer-Verlag, iSBN:3-540-65452-6, 1999, p. 217–235. Disponível em `http://portal.acm.org/citation.cfm?id=656271`

Böhm, C.; Berchtold, S.; Keim, D. A. Searching in high-dimensional spaces - index structures for improving the performance of multimedia databases. *ACM Computing Surveys*, v. 33, n. 3, p. 322–373, 2001. Disponível em `http://portal.acm.org/citation.cfm?doid=502807.502809`

Brin, S. Near neighbor search in large metric spaces. In: Dayal, U.; Gray, P. M. D.; Nishio, S., eds. *International Conference on Very Large Databases (VLDB)*, Zurich, Switzerland: Morgan Kaufmann, 1995, p. 574–584. Disponível em `http://www.vldb.org/conf/1995/P574.PDF`

Bueno, R.; Ferreira, M. R. P.; Traina Jr., C. *Dbgen - manual da ferramenta*. Technical Report 246, Institute of Mathematics and Computer Sciences (ICMC) / University of São Paulo (USP), ISSN - 0103-2569, 2004.

BURKHARD, W. A.; KELLER, R. M. Some approaches to best-match file searching. *Communications of the ACM (CACM)*, v. 16, n. 4, p. 230–236, iSSN:0001-0782, 1973. Disponível em `http://portal.acm.org/citation.cfm?id=362025`

CHÁVEZ, E.; NAVARRO, G.; BAEZA-YATES, R.; MARROQUÍN, J. L. Searching in metric spaces. *ACM Computing Surveys (CSUR)*, v. 33, n. 3, p. 273–321, iSSN:0360-0300, 2001.
Disponível em `http://portal.acm.org/citation.cfm?id=502808`

CIACCIA, P.; PATELLA, M. Bulk loading the m-tree. In: *ADC Australasian Database Conference*, 1998, p. 15–26.

CIACCIA, P.; PATELLA, M.; ZEZULA, P. M-tree: An efficient access method for similarity search in metric spaces. In: JARKE, M.; CAREY, M. J.; DITTRICH, K. R.; LOCHOVSKY, F. H.; LOUCOPOULOS, P.; JEUSFELD, M. A., eds. *VLDB '97: Proceedings of the 23rd International Conference on Very Large Data Bases*, Morgan Kaufmann Publishers Inc., iSBN:1-55860-470-7, 1997, p. 426–435.
Disponível em `http://portal.acm.org/citation.cfm?id=645923.671005`

COPSON, E. T. *Metric space*. Cambridge At The University Press, 1968.

DEZA, M.; LAURENT, M. Geometry of cuts and metrics. In: *Algorithms and Combinatorics*, v. 15, Springer, 1997.
Disponível em `www.liga.ens.fr/~deza/BOOK1/cutbook.pdf`

GAEDE, V.; GÜNTHER, O. Multidimensional access methods. *ACM Computing Surveys (CSUR)*, v. 30, n. 2, p. 170–231, iSSN:0360-0300, 1998.
Disponível em `http://portal.acm.org/citation.cfm?doid=280277.280279`

GUTTMAN, A. R-trees: a dynamic index structure for spatial searching. In: *SIGMOD '84: Proceedings of the 1984 ACM SIGMOD international conference on Management of data*, New York, NY, USA: ACM, sESSION: Physical database design. ISBN:0-89791-128-8, 1984, p. 47–57.
Disponível em `http://portal.acm.org/citation.cfm?id=602266`

HJALTASON, G. R.; SAMET, H. Ranking in spatial databases. In: EGENHOFER, M. J.; HERRING, J. R., eds. *SSD '95: Proceedings of the 4th International Symposium on Advances in Spatial Databases*, London, UK: Springer-Verlag, iSBN:3-540-60159-7, 1995, p. 83–95.

HJALTASON, G. R.; SAMET, H. Distance browsing in spatial databases. *ACM Transactions on Database Systems (TODS)*, v. 24, n. 2, p. 265–318, iSSN:0362-5915, 1999.

HJALTASON, G. R.; SAMET, H. Index-driven similarity search in metric spaces (survey article). *ACM Transactions on Database Systems (TODS)*, v. 28, n. 4, p. 517–580, iSSN:0362-5915, 2003.
Disponível em `http://portal.acm.org/citation.cfm?id=958948`

INDYK, P.; MATOUSEK, J. Low-distortion embeddings of finite metric spaces. In: *Handbook of Discrete and Computational Geometry*, CRC Press, 2004, p. 177–196.

KORN, F.; MUTHUKRISHNAN, S. Influence sets based on reverse nearest neighbor queries. In: EICH, M. H., ed. *SIGMOD '00: Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, Dallas, Texas, United States: ACM Press, iSSN:0163-5808. Chairman-Maggie Dunham and Chairman-Jeffrey F. Naughton and Chairman-Weidong Chen and Chairman-Nick Koudas, 2000, p. 201–212.
Disponível em `http://portal.acm.org/citation.cfm?id=335415`

LEE, K. C.; ZHENG, B.; LEE, W.-C. Ranked reverse nearest neighbor search. *IEEE Transactions on Knowledge and Data Engineering*, v. 20, n. 7, p. 894–910, 2008.

LEVENSHTEIN, V. Binary codes capable of correcting deletions, insertions, and reversals. *Cybernetics and Control Theory*, v. 10, n. 8, p. 707–710, 1966.

MAHESHWARI, A.; VAHRENHOLD, J.; ZEH, N. On reverse nearest neighbor queries. In: *14th Canadian Conference on Computational Geometry*, Alberta, Canada, 2002, p. 128–132.

MICÓ, L.; ONCINA, J.; VIDAL, RUIZ, E. A new version of the nearest-neighbor approximating and eliminating search (aesa) with linear processing-time and memory requirements. *Pattern Recognition Letters*, v. 15, p. 9–17, 1994.

NANOPOULOS, A.; THEODORIDIS, Y.; MANOLOPOULOS, Y. C2p: Clustering based on closest pairs. In: APERS, P. M. G.; ATZENI, P.; CERI, S.; PARABOSCHI, S.; RAMAMOHANARAO, K.; SNODGRASS, R. T., eds. *International Conference on Very Large Databases (VLDB)*, Roma, Italy: Morgan Kaufmann, 2001, p. 331–340.
Disponível em `http://www.vldb.org/conf/2001/P331.pdf`

ROUSSOPOULOS, N.; KELLEY, S.; VINCENT, F. Nearest neighbor queries. In: CAREY, M.; SCHNEIDER, D., eds. *SIGMOD '95: Proceedings of the 1995 ACM SIGMOD international conference on Management of data*, New York, NY, USA: ACM, iSBN:0-89791-731-6, 1995, p. 71–79.
Disponível em `http://portal.acm.org/citation.cfm?id=223784.223794`

SANTOS FILHO, R. F.; TRAINA, A. J. M.; TRAINA JÚNIOR, C.; FALOUTSOS, C. Similarity search without tears: The omni family of all-purpose access methods. In:

IEEE International Conference on Data Engineering (ICDE), Washington, DC, USA: IEEE Computer Society, iSSN:1063-6382, 2001, p. 623–630.
Disponível em `http://portal.acm.org/citation.cfm?id=876881.879683`

Singh, A.; Ferhatosmanoglu, H.; Şaman Tosun, A. High dimensional reverse nearest neighbor queries. In: *CIKM '03: Proceedings of the twelfth international conference on Information and knowledge management*, New York, NY, USA: ACM Press, iSBN:1-58113-723-0. SESSION: Database session 2: querying high-dimensional data II, 2003, p. 91–98.
Disponível em `http://portal.acm.org/citation.cfm?id=956863.956882`

Smid, M. Closest-point problems in computational geometry. 1997.
Disponível em `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.48.4292`

Stanoi, I.; Agrawal, D.; Abbadi, A. E. Reverse nearest neighbor queries for dynamic databases. In: *ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, 2000, p. 44–53.
Disponível em `http://www.cs.ucr.edu/~dg/DMKD.html`

Stanoi, I.; Riedewald, M.; Agrawal, D.; Abbadi, A. E. Discovery of influence sets in frequently updated databases. In: Apers, P. M. G.; Atzeni, P.; Ceri, S.; Paraboschi, S.; Ramamohanarao, K.; Snodgrass, R. T., eds. *VLDB '01: Proceedings of the 27th International Conference on Very Large Data Bases*, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., iSBN:1-55860-804-4, 2001, p. 99–108.
Disponível em `http://portal.acm.org/citation.cfm?id=645927.672207`

Tao, Y.; Papadias, D.; Lian, X. Reverse knn search in arbitrary dimensionality. In: Nascimento, M. A.; Özsu, M. T.; Kossmann, D.; Miller, R. J.; Blakeley, J. A.; Schiefer, K. B., eds. *Proceedings of the Thirtieth International Conference on Very Large Data Bases*, Morgan Kaufmann, iSBN: 0-12-088469-0. Research Session 20: Nearest Neighbor Search, 2004, p. 744–755.
Disponível em `http://www.vldb.org/dblp/db/conf/vldb/vldb2004.html`

Tao, Y.; Yiu, M. L.; Mamoulis, N. Reverse nearest neighbor search in metric spaces. *IEEE Transactions on Knowledge and Data Engineering*, v. 18, n. 9, p. 1239–1252, iSSN: 1041-4347, 2006.
Disponível em `http://doi.ieeecomputersociety.org/10.1109/TKDE.2006.148`

Traina Júnior, C.; Traina, A. J. M.; Faloutsos, C.; Seeger, B. Fast indexing and visualization of metric datasets using slim-trees. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, v. 14, n. 2, p. 244–260, iSSN:1041-4347, 2002a.
Disponível em `http://portal.acm.org/citation.cfm?id=628207`

Traina Júnior, C.; Traina, A. J. M.; Santos Filho, R. F.; Faloutsos, C. How to improve the pruning ability of dynamic metric access methods. In: *CIKM '02: Proceedings of the eleventh international conference on Information and knowledge management*, New York, NY, USA: ACM Press, iSBN:1-58113-492-4. SESSION: Sequence similarity search and access methods, 2002b, p. 219–226.
Disponível em `http://portal.acm.org/citation.cfm?doid=584792.584831`

Traina Júnior, C.; Traina, A. J. M.; Seeger, B.; Faloutsos, C. *Slim-trees: High performance metric trees miniminzing overlap between nodes.* Research Paper CMU-CS-99-170, Carnegie Mellon University - School of Computer Science, Pittsburgh, PA, USA, 19 pages, 1999.
Disponível em `http://reports-archive.adm.cs.cmu.edu/cs1999.html`

Traina Júnior, C.; Traina, A. J. M.; Seeger, B.; Faloutsos, C. Slim-trees: High performance metric trees minimizing overlap between nodes. In: Zaniolo, C.; Lockemann, P. C.; Scholl, M. H.; Grust, T., eds. *EDBT '00: Proceedings of the 7th International Conference on Extending Database Technology*, London, UK: Springer-Verlag, iSBN:3-540-67227-3, 2000, p. 51–65 (*Lecture Notes in Computer Science*, v.1777).
Disponível em `http://portal.acm.org/citation.cfm?id=645339.650146`

Uhlmann, J. K. Satisfying general proximity/similarity queries with metric trees. *Information Processing Letters*, v. 40, n. 4, p. 175–179, 1991.

Vidal, Ruiz, E. An algorithm for finding nearest neighbours in (approximately) constant average time. *Pattern Recognition Letters*, v. 4, n. 3, p. 145–157, 1986.

Wactlar, H. D.; Kanade, T.; Smith, M. A.; Stevens, S. M. Intelligent access to digital video: Informedia project. *IEEE Computer*, v. 29, n. 5, p. 46–52, 1996.

Xia, C.; Hsu, W.; Lee, M. L. 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*, New York, NY, USA: ACM Press, iSBN:1-59593-140-6. SESSION: Paper session IR-6 (information retrieval): IR models 1, 2005, p. 533–540.
Disponível em `http://portal.acm.org/citation.cfm?id=1099554.1099697`

Yang, C.; Lin, K.-I. An index structure for efficient reverse nearest neighbor queries. In: *IEEE International Conference on Data Engineering (ICDE)*, Washington, DC, USA: IEEE Computer Society, iSSN:1063-6382, 2001, p. 485–492.
Disponível em `http://portal.acm.org/citation.cfm?id=876881.879668`

Yianilos, P. N. Data structures and algorithms for nearest neighbor search in general metric spaces. In: *SODA '93: Proceedings of the fourth annual ACM-SIAM Symposium on Discrete algorithms*, Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 1993, p. 311–321.
Disponível em `http://portal.acm.org/citation.cfm?id=313789`

Zezula, P.; Amato, G.; Dohnal, V.; Batko, M. *Similarity search: The metric space approach*, v. 32 de *Advances in Database Systems*. Springer, 1–220 p., iSBN: 0-387-29146-6, 2006.