Exploração do Uso de *Short-term Memory* na Construção de Métodos de Acesso Métricos Dinâmicos Sobre a Perspectiva de Diferentes Políticas de Divisão de Nós

Régis Michel dos Santos Sousa

Universidade Federal de Uberlândia
Faculdade de Computação
Programa de Pós-Graduação em Ciência da Computação

Uberlândia
2017

Régis Michel dos Santos Sousa

Exploração do Uso de *Short-term Memory* na Construção de Métodos de Acesso Métricos Dinâmicos Sobre a Perspectiva de Diferentes Políticas de Divisão de Nós

Dissertação de mestrado apresentada ao Programa de Pós-graduação da Faculdade de Computação da Universidade Federal de Uberlândia como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Área de concentração: Ciência da Computação

Orientador: Prof. Dr. Humberto Luiz Razente

Uberlândia
2017

Dados Internacionais de Catalogação na Publicação (CIP)
Sistema de Bibliotecas da UFU, MG, Brasil.

S725e
2017

Sousa, Régis Michel dos Santos, 1990-
Exploração do uso de short-term memory na construção de métodos de acesso métricos dinâmicos sobre a perspectiva de diferentes políticas de divisão de nós / Régis Michel dos Santos Sousa. - 2017.
98 f. : il.

Orientador: Humberto Luiz Razente.
Dissertação (mestrado) - Universidade Federal de Uberlândia, Programa de Pós-Graduação em Ciência da Computação.
Inclui bibliografia.

1. Computação - Teses. 2. Recuperação da informação - Teses. 3. Indexação - Teses. I. Razente, Humberto Luiz. II. Universidade Federal de Uberlândia. Programa de Pós-Graduação em Ciência da Computação. III. Título.

CDU: 681.3

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

FACULDADE DE COMPUTAÇÃO

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Os abaixo assinados, por meio deste, certificam que leram e recomendam para a Faculdade de Computação a aceitação da dissertação intitulada "**Exploração do Uso de *Short-term Memory* na Construção de Métodos de Acesso Métricos Dinâmicos Sobre a Perspectiva de Diferentes Políticas de Divisão de Nós**" por **Régis Michel dos Santos Sousa** como parte dos requisitos exigidos para a obtenção do título de **Mestre em Ciência da Computação.**

Uberlândia, 17 de Março de 2017

Orientador: _____

Prof. Dr. Humberto Luiz Razente

Universidade Federal de Uberlândia

Banca Examinadora:

_____

Prof. Dr. Bruno Augusto Nassif Travençolo

Universidade Federal de Uberlândia

_____

Prof. Dr. Daniel Santos Kaster

Universidade Estadual de Londrina

Para Anézia, Fernando e Ronaldo com quem divido todas as minhas conquistas.

Agradecimentos

Agradeço em especial à minha mãe que sempre acreditou no meu potencial para enfrentar esse desafio de concluir mais uma etapa da minha formação profissional. Agradeço aos meus amigos pelo apoio incondicional, as palavras de incentivo, os objetivos compartilhados. Em especial a Fernando Silva, Ronaldo Silva e Gilberto Silva que se tornaram mais que amigos, são hoje a minha família aqui em Uberlândia. Aos meus queridos amigos e companheiros de laboratório Thaína, Sara Melo, Rafael Bernardes, Saulo Coutinho, Edinilson, Guilherme Silva, Luiz Fernando Afra Brito, Vinícios Silva, Bruno Messias e Juscelina Torres que estiveram sempre presentes e contribuíram em algum momento na elaboração deste trabalho. Em especial quero agradecer ao meu orientador Prof. Dr. Humberto Luiz Razente, que sempre esteve presente para me aconselhar e direcionar durante toda essa trajetória.

Agradeço também, os professores José Eduardo Lopes meu grande amigo e conselheiro profissional, ao Prof. Dr. Bruno Augusto Nassif Travençolo, Prof. Dr. Maria Camila e a Prof. Dr. Cristiane Betanho que me ajudaram de algum modo muitas vezes durante essa caminhada. Um agradecimento mais do que especial ao secretário Erisvaldo Araújo Fialho por toda a paciência e boa vontade em sanar minhas dúvidas.

Aos professores da FACOM pelos ensinamentos. À CAPES e CNPQ pelo apoio financeiro.

"O que eu conheço de Deus são as pegadas de sua ausência."(Padre Fábio de Melo).

Resumo

Este trabalho propõe o desenvolvimento de uma nova abordagem utilizando uma estrutura denominada *short-term memory*, para a construção de Métodos de Acesso Métricos dinâmicos derivados da Slim-Tree. Deste modo, o objetivo é minimizar a sobreposição das estruturas e, consequentemente, otimizar as operações de consulta por similaridade. A estratégia para alcançar este objetivo, fundamenta-se em adiar o processo da indexação para contabilizar no índice novos objetos com distâncias próximas. Além disso, foram empregadas distintas políticas de divisão de nós, o que permitiu mensurar os impactos destas no que refere-se a qualidade das estruturas resultantes, principalmente com relação ao grau de sobreposição. Os novos métodos propostos foram avaliados por conjunto de dados reais e sintéticos e comparados com MAM Slim-tree original. Os resultados dos experimentos foram significativos com relação à qualidade das árvores resultantes o que, consequentemente, impactou em melhor eficiência nas operações de consulta por similaridade, ao reduzir significativamente o número de cálculos de distância, a quantidade de acessos a páginas de disco e, consequentemente, o tempo de execução de consultas aos k-vizinhos mais próximos.

Palavras-chave: MAM dinâmicos. Busca por similaridade. Short-term memory. Dados Complexos. Políticas de divisão de nós.

Abstract

This work proposes the development of a new approach using a structure called short-term memory for the construction of dynamic MAMs derived from Slim-Tree. The goal is to minimize the overlap of nodes in the structure and, consequently, to optimize similarity queries. The strategy is based on postponing the indexing process to allow inserting, in the same index entry, new objects with closer distances. In addition, different node split policies were employed, which allowed to measure the impacts of these in what refers to the quality of the resulting structures, mainly with respect to the degree of overlap. The proposed new methods were evaluated by real and synthetic datasets and compared with the original MAM Slim-tree. The results of the experiments were significant with respect to the quality of the resulting trees, which consequently impacted on better efficiency in similarity query operations, significantly reducing the number of distance calculations, the number of disk page accesses and the execution time of k-nearest neighbors.

Keywords: Dinamics MAMs. Similarity Search. Short-term Memory. Complex data, Policies of Division of Nodes.

Lista de ilustrações

Lista de tabelas

Lista de siglas

DM Dissimilaridade Máxima

FFabs *Fat Factor Absoluto*

FFrel *Fat Factor Relativo*

HCS *Hybrid ChooseSubTree*

k-NN *k-Nearest Neighbor Neighbor*

MAM *Metric Access Methods*

MAE Método de Acesso Espacial

MinMax Mínimo de maiores raios

MST *Minimal Spanning Tree*

RQ *range query*

SBGD Sistemas Gerenciadores de Bases de Dados

SDC Soma das distâncias dos Caminhos

SM *Short-term Memory*

SM-Random *Slim Memory Random*

SM-Density *Slim Memory Density*

SM-Cluster *Slim Memory Cluster*

SynD *Synthetic data with only concept-drift*

SynEDC *Synthetic data with concept-drift and novel-class*

Sumário

Capítulo 1

Introdução

Indexação e recuperação por similaridade são operações fundamentais para aplicações que lidam com dados complexos. Para compreensão a complexidade deste tema, é imprescindível considerar que os dados atualmente são produzidos a cada ligação telefônica, mensagem enviada, pesquisa em máquina de busca (Google, Yahoo, etc.), compras em cartões de crédito e em praticamente toda ação do cotidiano. Grande parte destes dados é gerada em tempo real, e podem assumir uma série de formatos variados, tais como: vetoriais, de trajetórias, multimídia (imagens, áudios e vídeos), grafos, séries temporais, entre outros, e são denominados dados complexos. Para esses dados é empregada a noção de similaridade quando o objetivo é a recuperação, sendo útil recuperar os documentos similares (ou menos dissimilares) aos de referência.

As estruturas que permitem indexar e recuperar rapidamente um determinado objeto são conhecidas como Métodos de Acesso Métricos (MAMs) e Métodos de Acesso Espaciais (MAEs). Nesses métodos, o uso de pivôs (elementos de referência) globais e locais têm potencial para melhorar o desempenho da computação das consultas por similaridade, entre elas, as consultas aos k-vizinhos mais próximos (OLIVEIRA; TRAINA-JR; KASTER, 2015). Em MAMs dinâmicos construídos incrementalmente e que fazem uso de particionamento do espaço por bola, como o caso da M-Tree e suas variantes, a medida que novos elementos são inseridos na estrutura de indexação, os nós das subárvores tendem a aumentar e, consequentemente, seus raios de cobertura também aumentam. Assim, eleva-se a probabilidade de ocorrer sobreposição entre as regiões de cobertura dos nós, o que, em geral, resulta na degradação do desempenho durante as operações de consulta por similaridade (TRAINA-JR et al., 2000).

Diversos trabalhos encontrados na literatura, por exemplo (SOUZA; RAZENTE; BARIONI, 2014; SKOPAL, 2006), têm como foco melhorar os mecanismos de particionamento (também conhecido como políticas de divisão de nós) quando os nós atingem sua capacidade máxima. Essas estratégias pretendem melhorar a distribuição hierárquica dos dados nas subárvores e auxiliar na redução sobreposição das estruturas. Desse modo, este método contribui diretamente para melhorar o desempenho das consultas, pois propor-

ciona maior poder de poda por meio da propriedade da desigualdade triangular. Uma outra abordagem recente que se beneficia em otimizar o poder de poda nas estruturas é proposta em (OLIVEIRA; TRAINA-JR; KASTER, 2015). Nesse trabalho utiliza-se como estratégia adição de pivôs locais. Ambos as pesquisas mencionadas anteriormente, utilizaram conjuntos de dados (reais e sintéticos) dos quais são indexados para posteriormente realizar lotes de consultas para avaliação de desempenho dos métodos de acesso propostos, com relação ao número de cálculos de distância, acessos a disco e outros parâmetros. Mas, essas soluções têm como principal desvantagem configurar-se como uma solução que ocorre localmente entre os nós. Dessa forma, apesar de trazer bons resultados ao reduzir a sobreposição, essas soluções não são abrangentes. Uma solução global, que leva em consideração todos os elementos do conjunto, seria reconstruir a estrutura, ideia similar a proposta por meio do algoritmo *Slim Down* (TRAINA-JR et al., 2000), no entanto o custo é proibitivo.

Esta dissertação apresenta o desenvolvimento de uma nova abordagem para a construção de MAMs dinâmicos derivados da Slim-Tree. A estratégia fundamenta-se em adiar o processo da indexação para inserir no índice novos elementos com distâncias próximas. A ideia foi inspirada nos conceitos da área de *data streams*, especificamente da abordagem de processamento proposta em (FARIA et al., 2016a) que faz uso de uma estrutura denominada *short-term memory* (SM) para receber temporariamente os elementos que não são cobertos pelo modelo de dados atual. No contexto de indexação, serão movidos para essa estrutura os elementos que tragam aumento dos raios de cobertura das subárvores.

Diante disto, é formulado o objetivo geral que incide na construção de MAMs dinâmicos assistidos por *short-term memory* na perspectiva de diferentes políticas de divisão de nós, com a finalidade de minimizar a sobreposição das estruturas e, consequentemente, otimizar as operações de consulta por similaridade. Além disso, a estratégia pode ser aplicada às estruturas métricas ou multidimensionais, como a M-tree (CIACCIA; PATELLA; ZEZULA, 1997), R-tree (GUTTMAN, 1984) e variantes. Os métodos foram avaliados utilizando conjunto de dados reais e sintéticos e comparados com MAM Slim-tree original (TRAINA-JR et al., 2000). Os resultados dos experimentos foram significativos no que se refere à qualidade das árvores resultantes o que, consequentemente, impactou em melhor eficiência nas operações de consulta por similaridade.

1.1 Motivação

Os Métodos de Acesso Métricos (MAM) são estruturas que permitem indexar e recuperar rapidamente dados complexos. Estas estruturas reduzem o número de cálculos de distância e o número de acessos a páginas de disco durante as operações de consulta por similaridade, em relação à comparação sequencial de dados. No entanto, o custo computacional dos algoritmos que determinam a similaridade entre pares de elementos torna as

buscas por similaridade operações de custo elevado.

As propostas de MAMs que trabalham em memória principal têm vantagens no sentido de apresentarem melhores desempenhos, principalmente por não ter a necessidade de acessar partições de disco para recuperar os dados. No entanto, a memória é limitada, o que a torna inadequada para bases de dados volumosas.

Em relação aos métodos que fazem uso de memória secundária para armazenamento, é importante esclarecer que a unidade básica de operações com um disco é um bloco. Quando uma informação é lida em um disco, tem-se que o bloco inteiro que contém essa informação é lido na memória principal, e, quando a informação é armazenada no disco, um bloco inteiro é escrito para ele (DROZDEK; PAIVA, 2002). Comparado com o tempo de transferência de informação dentro da memória principal, esse processo é extremamente lento. Assim, justifica-se a necessidade de minimizar o número de acesso a discos para melhorar o desempenho das consultas.

As propriedades dos espaços métricos, principalmente a desigualdade triangular, auxiliam na otimização dos MAMs ao permitir a realização de poda nas estruturas de indexação, o que reduz o número de cálculos de distância e o número de acessos a páginas de disco durante as consultas por similaridade. Porém ao utilizar a técnica de particionamento do espaço por bola, a medida que os dados vão crescendo em proporção na estrutura, muitas regiões vão sofrendo degradação pelo fato de ocorrer sobreposição entre as subárvores, o que influencia negativamente para a realização de podas das subárvores.

Diante dos argumentos expostos é fundamental propor novas técnicas para reduzir a sobreposição resultante das estruturas, visando melhora do particionamento do espaço, ou seja, melhorar a distribuição hierárquica, visto a necessidade de não comprometer a dinamicidade da estrutura ao realizar a inserção de novos elementos e contribuir para consultas por similaridade eficientes.

1.2 Objetivos e Desafios da Pesquisa

O objetivo geral desta pesquisa é propor e desenvolver métodos de acesso métrico dinâmicos derivados da Slim-Tree assistidos por *short-term memory*, na perspectiva de diferentes políticas de divisão de nós, a fim de otimizar as operações de consulta por similaridade.

Para alcançar este objetivo foi necessário satisfazer os seguintes objetivos específicos:

i) Utilizar a estrutura *short-term memory* para auxiliar a construção de métodos de acesso métricos na perspectiva de distintas políticas de divisão de nós;

ii) Elaborar estratégias para processar os dados presentes na *short-term memory*, com intuito de gerar nós folhas completos, respeitando a capacidade e a taxa de ocupação dos nós;

iii) Propor um método de inserção nós folhas derivados da *short-term memory* na estrutura de indexação;

iv) Testar os métodos propostos para bases de dados distintas;

v) Avaliar os custos com relação ao processo de construção das estruturas de indexação;

vi) Avaliar o grau de sobreposição das árvores resultantes por meio do conjunto de medidas fat-factor;

vii) Avaliar a qualidade dos resultados para uso na indexação métrica, durante as operações de consultas por similaridade, em comparação com o método Slim-tree.

1.3 Hipótese

Hipótese 1. *É possível minimizar a taxa de sobreposição de estruturas de indexação métricas postergando-se a inserção de elementos que aumentam o raio de cobertura de grupos indexados e, posteriormente, adicionando-se estes elementos como grupos de registros com distâncias similares.*

1.4 Organização da Dissertação

Esta dissertação está organizada em quatro capítulos conforme disposto a seguir:

- Capítulo 1: Apresenta disseminação na utilização de buscas por similaridade em sistemas reais, e a necessidade da criação de estratégias de armazenar, organizar e recuperar essas informações indexadas em MAMs de forma eficiente. Apresenta também a diversidade de técnicas propostas na literatura para otimização dos métodos, e como tais técnicas ainda necessitam de serem melhoradas, visto que é necessário balancear entre o custo benefício com relação à construção e as consultas por similaridade. Ainda nas subseções, 1.1 aborda-se com detalhes a motivação do problema. Na 1.2 os objetivos do trabalho, principal e específicos, e por fim, 1.3 são apresentadas as hipóteses que direcionam o trabalho.

- Capítulo 2: Apresenta conceitos que servem de base para a estruturação do projeto no que se refere validação dos métodos e técnicas proposta. São apresentados os conceitos de consultas por similaridade, métodos de acesso, com ênfase nos métodos de acesso métricos dinâmicos. Em seguida, os principais trabalhos correlatos a abordagem proposta nesta dissertação.

- Capítulo 3: Descreve a abordagem proposta para melhorar o desempenho dos Métodos de Acesso Métricos dinâmicos, ao fazer uso da *short-term memory*. Além

disso, são apresentados os algoritmos implementados e utilizados para elaboração do trabalho.

- Capítulo 4: Apresenta os resultados obtidos nos experimentos realizados.

  É importante destacar que os experimentos foram realizados com bases de dados distintas, das quais foram considerados as peculiaridades de cada uma das bases de dados, tais como cardinalidade, dimensionalidade e distribuição dos dados. Os testes tiveram como intuito a realizar de lotes consultas (k-NN) para avaliação do desempenho dos métodos, com relação ao número de cálculos de distância, número de acessos a páginas de disco e tempo total das consultas. Outros parâmetros explorados foram o conjunto de medidas fat-factor, que avalia a qualidade das árvores resultantes.

- Capítulo 5: São exibidas as considerações finais com relação a esta pesquisa, objetivando apresentar a sua importância para a comunidade acadêmica, devido a suas possibilidade e descobertas. Além disso, serão apontadas as limitações deste trabalho.

Capítulo 2

Fundamentação Teórica

Nesta seção é apresentada uma breve revisão da literatura correlata a esta proposta de dissertação. O objetivo é posicionar esta proposta em relação aos demais trabalhos já desenvolvidos e publicados. Esta seção está organizada como segue: (2.1) apresenta-se os conceitos sobre medidas de similaridade; (2.2) expõem-se a definição consultas por similaridade destacando os e os principais tipos de consultas; (2.3) é apresentado a definição de Espaço Métrico e Funções Métricas destacando-se conceitos sobre a família de distância Minkowski; (2.4) é apresentado a definição de Métodos de Acesso Métricos, destacando a principalmente o MAM Slim-Tree e também as medidas Fat-Factor para mensuração do grau de sobreposição. E por fim, na Seção (2.5) são apresentados os principais trabalhos correlatos que norteiam esta pesquisa.

2.1 Medidas de Similaridade

As operações que envolvem a recuperação por conteúdo ou similaridade são baseadas no cálculo da dissimilaridade, de modo que um objeto de consulta seja comparado com os demais elementos presentes em um determinado conjunto. Desse modo, a similaridade entre os elementos é definida por meio de uma função de distância $d(O_i, O_j)$, segundo a qual, quanto menor a distância menor será a dissimilaridade entre os elementos O_i e O_j.

Assim sendo, para realizar buscas por similaridade surgiram na literatura métodos de indexação, tais como: Método de Acesso Espaciais (MAE) e Métodos de Acesso Métricos (MAM) (CHÁVEZ et al., 2001), sendo estes capazes de indexar elementos por meio de uma métrica e, durante o processo de recuperação atingem bons resultados ao reduzir o espaço de busca e, consequentemente, minimizam o número de cálculos de distância e número de acessos a disco. Tais detalhes serão abordados na Seção 2.4.

Enquanto os MAEs assumem que os elementos estão em um espaço multidimensional, ou seja, vetores com n-dimensões, os MAMs consideram o domínio do espaço métrico. Sendo o espaço métrico mais genérico, pois é possível indexar qualquer tipo de objeto, incluindo os adimensionais, ou seja, elementos que não podem ser identificados por um

conjunto de coordenadas em eixos ortogonais. Um exemplo prático consiste de um domínio de palavras do qual cada palavra tem um número variável de caracteres e uma função de distância como a distância de edição.

2.2 Consultas por Similaridade

A recuperação dos dados é feita por meio de consultas por similaridade, sendo esta um processo para se obter um conjunto de elementos ordenados pela sua distância, ou dissimilaridade para um dado objeto de consulta (*query*). A função de dissimilaridade permite realizar uma ordenação ou classificação do conjunto de dados em relação ao objeto de consulta, tendo como critério a distância de cada objeto para o objeto de consulta. Esta retorna zero se ambos os elementos O_i e O_j forem idênticos, e um valor positivo que aumenta quanto maior for a distância ou dissimilaridade entre os elementos.

Embora este princípio funcione para qualquer função de dissimilaridade, para a otimização por meio de métodos de acesso métrico, as funções precisam satisfazer as propriedades métricas, apresentadas na seção 2.3.

Os principais tipos de consulta por similaridade são *Range Query* (consulta por abrangência) e *k-Nearest Neighbor Query* (consulta aos k-vizinhos mais próximos), apresentadas respectivamente pelas Figuras 1 (a) e (b). Seja Q definido como o objeto de consulta e D como o domínio da aplicação as consultas são definidas como:

- ***Range Query* (rq):** recupera todos os elementos cuja distância ao centro do objeto Oq é menor ou igual ao raio fornecido (CIACCIA; PATELLA; ZEZULA, 1997). Dada pela expressão $rq(Oq, r) = x \epsilon X | d(Oq, x) \leq rq$.

- ***K-Nearest Neighbor Query* (k-nn):** recupera k elementos que possuem a menor distância em relação a Q, ou seja, a consulta $k - nn(Q, k)$ da qual busca recuperar os k elementos mais próximos de Q. Sendo que, $Q \in D$ e k é um número inteiro, tal que, $k \geq 1$ (CIACCIA; PATELLA; ZEZULA, 1997).

Conforme representado na Figura 1, a consulta por similaridade é realizada utilizando uma função de dissimilaridade, por exemplo, distância Euclidiana. O objeto O_q é o objeto de busca (*query*), enquanto os elementos representados em tons de cinza, constituem os elementos do conjunto resposta a dada consulta. Em (a) é representado uma consulta por abrangência - $rq(O_q, r_q)$ enquanto em (b) consulta aos k-vizinhos mais próximos - $knn(O_q, 4)$. Na próxima seção são apresentados conceitos com relação as métricas de distância empregadas nesse trabalho.

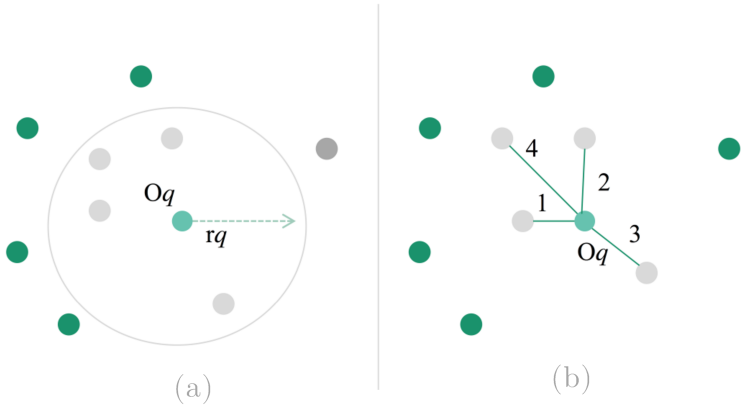

Figura 1 – Consultas por similaridade em domínio bidimensional no qual o objeto Oq é o objeto de consulta. Em (a) ilustra a consulta por abrangência com o raio de busca rq e (b) ilustra a consulta aos k-vizinhos mais próximos com k=4. Adaptado de (CHINO, 2004).

2.3 Definição de Espaço Métrico e Funções Métricas

Um espaço métrico $\mathbb{M}$ é definido pelo par $< \mathbf{D}, d() >$, onde d() é uma função de distância, denominada métrica, e $\mathbf{D}$ é o domínio dos dados. Nesse contexto, dados os elementos $a_1, a_2, a_3 \in D$ os seguintes axiomas devem satisfazer as propriedades:

- **Identidade:** $d(a_1, a_1) = 0$.
- **Simetria:** $d(a_1, a_2) = d(a_2, a_1)$.
- **Não negatividade:** $0 < d(a_1, a_2) < \infty$, $se\ a_1 \neq a_2$.
- **Desigualdade triangular:** $d(a_1, a_2) \leq d(a_1, a_3) + d(a_3, a_2)$.

Para indexar um objeto do qual se possa extrair suas características, é importante que seus vetores de características estejam bem representados. Assim, é possível utilizar uma métrica que melhor se adeque ao domínio dos dados. Entre as métricas mais utilizadas estão aquelas da família de *Minkowski*.

2.3.1 Família de distâncias *Minkowski*

Na literatura existem diversos tipos de funções distâncias, ou simplesmente métricas, designadas a aplicações especificadas, no entanto a mais utilizadas são da família *Minkowski* ou métricas L_p (BUGATTI; TRAINA; TRAINA-JR, 2008), das quais são frequentemente aplicadas em domínios multidimensionais onde os elementos são definidos por meio de tuplas $x_1, x_2, x_3, ..., x_n$, de valores reais, onde n é o número de dimensões. A família de distâncias de Minkowski (L_p), pode ser definida pela Equação 1:

$$L_p(x, y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{1/p} , p \geq 1, x, y \in \mathbb{R}^n, \quad (1)$$

onde n é o número de dimensões e a $L_p(x, y)$ é a distância do elemento x ao elemento y. Os casos especiais desta métrica comumente utilizados são: $L1$ ou distância de *Manhanttan* quando $p = 1$, $L2$ ou distância de Euclidiana quando $p = 2$ e $L\infty$ ou *Chebychev* quando p tende ao infinito, sendo estas definidas formalmente pelas Equações 2, 3 e 4 respectivamente.

$$d_{L1}(x, y) = \sum_{i=1}^{n} |x_i - y_i| \tag{2}$$

$$d_{L2}(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{3}$$

$$d_{L\infty}(x, y) = \lim_{p \to \infty} \left(\sum_{i=1}^{n} (x_i - y_i)^p \right)^{1/p} = max(|x_i - y_i|) \tag{4}$$

Cada uma das métricas apresentadas possui comportamento distinto que varia de acordo com o valor numérico atribuído ao parâmetro p, o que no contexto de recuperação por similaridade pode retornar elementos distintos como resposta dependo do tipo de consulta realizado. Esse fato é ilustrado na Figura 2.

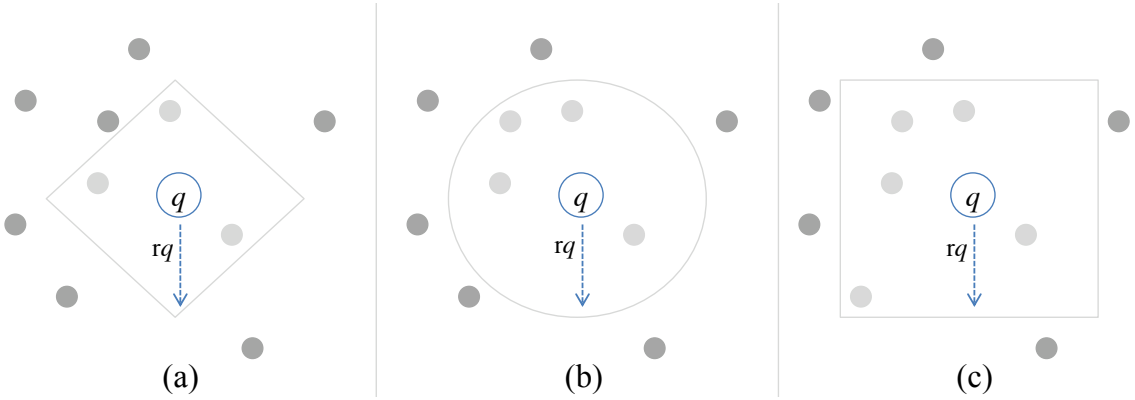

Figura 2 – Representação das formas geométricas geradas para as funções de distância L_1, L_2 e $L\infty$ para os pontos equidistantes à distância ε a partir do elemento central s_q, por meio de consulta por abrangência com o raio de busca rq. Adaptado de (TEIXEIRA, 2013).

Nesse sentido, a escolha da função de distância dependerá do contexto da aplicação. Sendo que cada uma das funções apresenta custos computacionais diferentes para serem computadas. Na próxima seção serão apresentados os conceitos sobre Métodos de Acesso Métricos, estrutura pela qual esta dissertação tem como objetivo otimizar com a finalidade de acelerar consultas por similaridade.

2.4 Métodos de Acesso Métricos

Os Métodos de Acesso Métricos (MAMs) estão entre as principais técnicas utilizadas em consultas por similaridade sobre dados complexos, que permitem indexar os elementos

apenas em função do valor de distância entre eles. Esses métodos são baseados nos Métodos de Acesso (MAs) uma das principais técnicas usadas nos SGDBs para recuperar informação de maneira mais eficiente (VIEIRA et al., 2010).

Os MAMs indexam um determinado conjunto de dados de maneira que as consultas por conteúdo sejam processadas com mais eficiência, isto porque, ao fazer uso de índices, não é necessário pesquisar todo o conjunto de dados. Consequentemente, minimiza-se o número de comparações entre os elementos ao fazer uso das propriedades do espaço métrico, principalmente da desigualdade triangular (SKOPAL, 2006).

Os MAMs são construídos por meio de uma abordagem recursiva e são organizados em páginas de disco de tamanho fixo, similar ao proposto na árvore B (COMER, 1979) e fazem uso de elementos para particionamento do espaço em regiões definidas por um ou mais elementos de referência. De modo geral, os Métodos de Acesso Métrico são categorizados de diferentes maneiras, tais como:

- **Tipo de Resposta:** Em mecanismos de recuperação de informação são empregadas respostas exatas ou aproximadas. A resposta exata retorna o mesmo que uma consulta sequencial, ou seja, a resposta correta da consulta. Há métodos que otimizam, mas não garantem que algum elemento que deveria estar na resposta não esteja. Por exemplo, ao selecionar os k=10 mais próximos, pula um elemento e inclui o que seria o 11º como parte da resposta. Nesses casos, a justificativa em geral é que a otimização é muito boa, mas diz-se que a resposta é aproximada, por não ter garantias da precisão em relação à revocação.

- **Dinamicidade da Estrutura:** caracterizada como dinâmica ou estática. Se a estrutura for estática é necessário a existência *a priori* do conjunto de dados para ser indexado e não permite muitas das vezes a realização de atualizações posteriores. Para isso, é necessário a construção dos índices do zero, sendo estes as principais estruturas propostas VP-tree (YIANILOS, 1993), GNAT (BRIN, 1995) e MVP-tree (BOZKAYA; OZSOYOGLU, 1997). No caso de a estrutura ser dinâmica é possível realizar atualizações (inserções ou remoções) de elementos a qualquer momento sem que ocorra a reconstrução de toda a estrutura. Entre as principais estruturas estão, M-tree (CIACCIA; PATELLA; ZEZULA, 1997), Slim-Tree (TRAINA-JR et al., 2000), DBM-Tree (VIEIRA et al., 2010) e RLC (MAMEDE; BARBOSA, 2007).

- **Forma de Particionamento do Espaço de Busca:** Os principais tipos de particionamento são: particionamento por bola (UHLMANN, 1991), particionamento por hiperplano (NAVARRO; URIBE-PAREDES, 2011), e particionamento por meio do excluído (YIANILOS1, 1999). Como apresentado na Figura 3.

- **Tipo de Pivôs:** Pivôs locais e Pivôs globais. Os pivôs constituem-se de elementos que agem como representantes de regiões do espaço de busca e são utilizados para

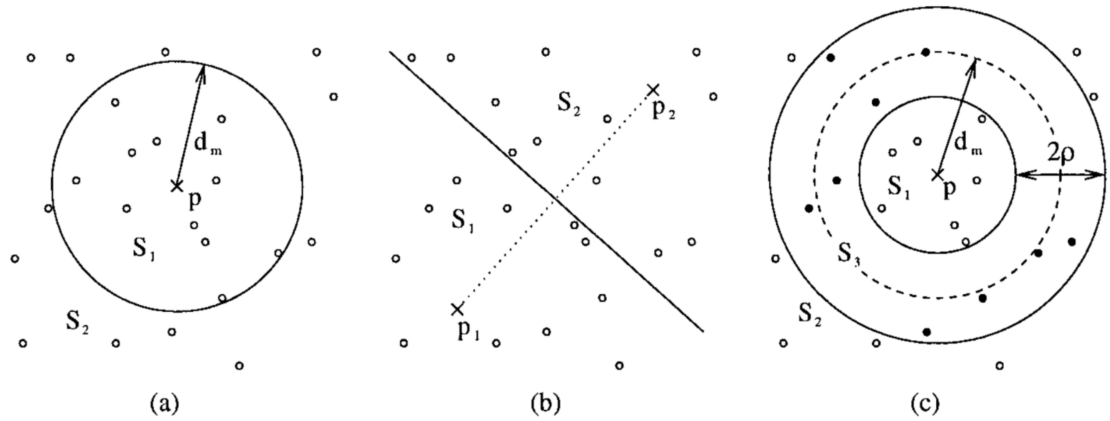

Figura 3 – Exemplos de formas de particionamento. Em (a) particionamento por bola, (b) particionamento por hiperplano e (c) particionamento por meio do excluído. Fonte: (ZEZULA et al., 2006).

podar elementos irrelevantes durante a execução de consultas. Um pivô é global quando todos os elementos do conjunto fazem referência a ele, enquanto um pivô local é referenciado apenas por uma parcela do conjunto de dados. Os pivôs globais podem ter sua dinamicidade comprometida pelo fato de eventuais atualizações necessitarem ser propagadas por toda a estrutura. Pivôs locais, por outro lado, permitem que a manutenção ocorra localmente ao preço de um menor poder de poda.

- **Tipo de Armazenamento:** O armazenamento em memória principal é empregado às estruturas como a k-d-tree (BENTLEY, 1975). A busca em memória principal é mais rápida do que em memória secundária (disco). No entanto, os métodos estão condicionados a especificidades como cardinalidade, dimensão e volume de dados. Sendo assim, em geral não é possível armazenar na memória principal grandes bases de dados. Os métodos propostos para trabalhar com memória secundária não sofrem da limitação expressa anteriormente, isso porque, fazem uso de armazenados dos dados em arquivos, pois dessa forma é possível armazenar grandes quantidades de dados. No entanto, o tempo de busca em disco é mais lento (DROZDEK; PAIVA, 2002), que por conseguinte torna a questão **E/S** uma preocupação, isto porque afeta negativamente o desempenho do método, devido ao número de leituras e escritas ao disco que são necessárias para recuperar elementos.

As estruturas estáticas não suportam operações de deleção e inserção de novos elementos, pois para permitir tais operações é necessário refazer todos os índices. Além disso, tem como desvantagem poder lidar apenas com pequenas bases de dados e, por isso, não serão utilizadas nesse trabalho. Desse modo, o escopo desta pesquisa contempla a utilização de Métodos de Acesso Métricos dinâmicos (ver seção 2.4.1) que são construídos incrementalmente ao serem utilizados por aplicações que necessitam de índices dinâmi-

cos em armazenamento secundário e que suportam operações de inserções e remoções de elementos.

2.4.1 Métodos de Acesso Métricos Dinâmicos

Em MAMs dinâmicos a medida em que os dados são inseridos ou removidos a estrutura deve atualizar corretamente os índices afim de representar adequadamente as consultas subsequentes.

A construção é realizada de baixo para cima, abordagem *bottom-up*, sendo que o índice pode iniciar vazio e ser construído por meio das sucessivas inserções. Na literatura há diversas métodos dinâmicos propostos dentre os principais estão *M-Tree* (CIACCIA; PATELLA; ZEZULA, 1997), a *Slim-Tree* (TRAINA-JR et al., 2000), a *DF-Tree* (TRAINA et al., 2002; TRAINA-JR et al., 2002b), a *DBM-Tree* (VIEIRA et al., 2010), a *OMNI-Family* (TRAINA-JR et al., 2007), a *Onion-Tree* (CARÉLO et al., 2011) e a *DSAT* (NAVARRO; REYES, 2016). Na próxima seção será descrito o funcionamento da MAM Slim-Tree o qual foi utilizado para testar e validar os resultados deste estudo. Os métodos aqui empregados podem ser generalizados para qualquer uma das suas variantes.

2.4.2 Slim-Tree

A Slim-Tree é um Método de Acesso Métrico dinâmico descendente da M-Tree, desenvolvido com a finalidade de melhorar o desempenho da sua construção do índice e de proporcionar consultas por similaridade mais eficientes na medida em que minimiza o número de acessos a disco e o número de cálculos de distância, preservando o balanceamento da estrutura. Além disso, é uma estrutura métrica explicitamente projetada com o intuito de reduzir o grau de sobreposição (TRAINA-JR et al., 2000). Em sua composição foi proposto um algoritmo chamado *Slim-Down* que busca diminuir a sobreposição existente entre os nós folhas. Além disso, foi proposta uma nova política de divisão de nós baseada na Árvore Geradora Mínima (*Minimal Spanning Tree – MST*) e uma nova estratégia de inserção fundamentada na ocupação mínima (quantidade mínima de elementos presentes nos nós). A árvore é balanceada e formada por nós índices (*index nodes*) e nós folhas (*leaf nodes*). Nos nós folhas são armazenados os elementos cobertos pelo nó e nos índices são armazenados os representantes dos nós e os respectivos raios de cobertura, como ilustrado na Figura 4.

A construção é realizada dinamicamente mediante as inserções de elementos. Assim, a inserção inicia-se pelo nó raiz, em que o algoritmo busca encontrar um nó que cobre o novo objeto. Se mais de um nó se qualificar, o algoritmo *ChooseSubTree* seleciona um dos nós para a inserção. Caso o novo objeto não seja coberto por nenhum nó, ou seja, se a distância do novo objeto ao representativo do nó está fora do raio de cobertura, o raio do nó é aumentado de modo a incluir o novo objeto. Além disso, a inserção pode causar

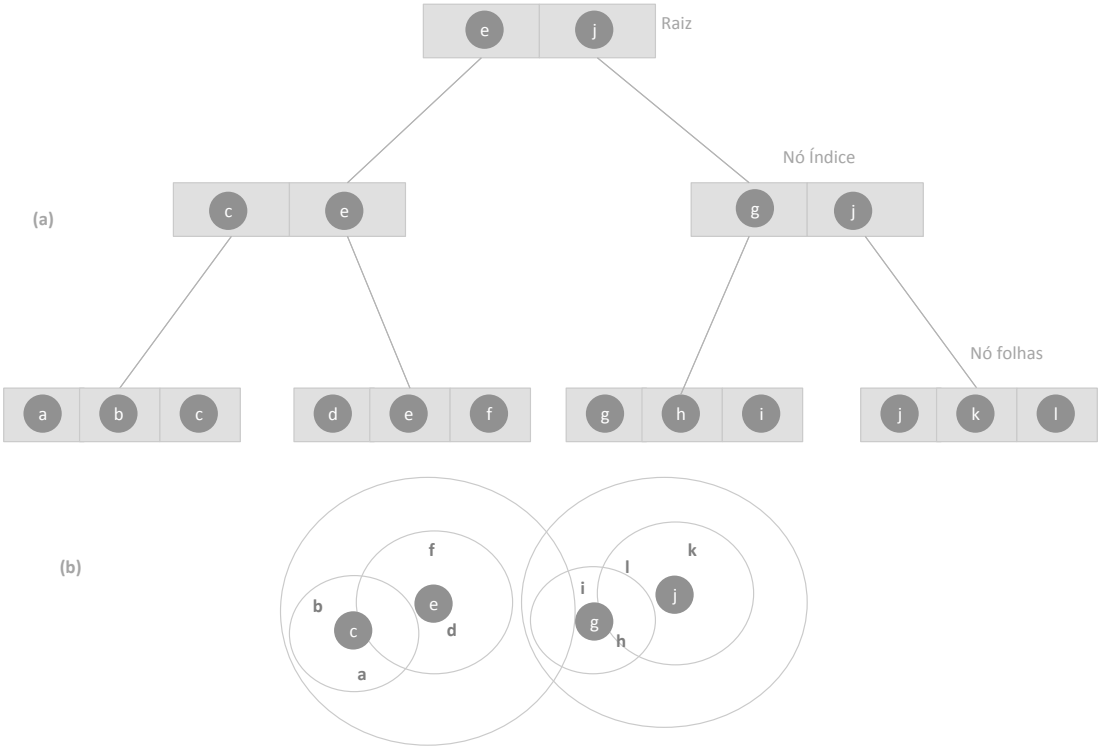

Figura 4 – Exemplo de um Slim-Tree indexando elementos bidimensionais utilizando a função de distância L2. (a) organização da árvore mediante a estrutura de nós; (b) árvore considerando apenas os raios de cobertura das subárvores. Adaptado de (CHINO, 2004).

a divisão de uma folha, desse modo, é necessário verificar se o nó não está cheio. Caso não esteja, basta inserir o novo objeto e atualizar o raio caso necessário. Se estiver cheio, os elementos deverão ser divididos em duas folhas. Esse método é conhecido por divisão do nó, realizado por meio do algoritmo de *splitting*. Esse processo será propagado para os nós superiores recursivamente. Com relação ao algoritmo *ChooseSubTree* a escolha do melhor nó é baseada em três critérios:

- **Aleatório:** escolhe aleatoriamente qualquer um dos ramos que se qualificam para inserir o novo objeto;
- **Distância Mínima:** escolhe o ramo cujo objeto representativo esteja mais próximo ao objeto inserido;
- **Ocupação Mínima:** escolhe o ramo cuja a taxa de ocupação seja mínima.

Assim sendo, em detrimento da escolha por um destes critérios mencionados anteriormente pode-se verificar algumas consequências, isto porque, ao se optar pelo algoritmo de distância mínima, verificou-se experimentalmente que resulta em árvores com taxa de

ocupação menor e menor grau de sobreposição, mas como desvantagem, as árvores terão alturas maiores. Ao se optar por ocupação mínima as árvores resultantes terão alturas menores com taxa de ocupação maior, porém com maior grau de sobreposição (TRAINA-JR et al., 2000). Uma outra proposta para o algoritmo que escolhe a subárvore é chamada de HCS (*Hybrid ChooseSubTree*) (YAMAMOTO; BIAJIZ; TRAINA-JR, 2003) que concilia a escolha da subárvore baseada na mínima distância e na ocupação do nó através do uso da lógica nebulosa.

2.4.3 Políticas de Divisão de Nós

Um dos procedimentos primordiais no processo de construção de métodos de acesso métricos é o responsável pela divisão de nós. Esse processo ocorre quando o nó no qual um novo elemento deveria ser inserido já se encontra completo. Nesse caso, são empregadas políticas de divisão ou também conhecidas como algoritmos de *splitting* para escolha de elementos representantes dos novos nós. As políticas de divisão de nós clássicas variam de acordo com os seguintes critérios:

- **Aleatório:** escolhe aleatoriamente dois elementos e os elegem como representativos.
- **MinMax:** realiza combinações de pares de elementos para descobrir quais deles configuram-se como melhores elementos representativos para os novos nós. Logo, o custo $O(n_3)$, em que n é a quantidade de elementos de um nó.
- **MST:** árvore de caminho mínimo para a seleção dos agrupamentos de elementos, com custo computacional $O(nlog(n))$, em que n é a quantidade de elementos de um nó.

A Figura 5 ilustra o funcionamento das políticas de divisão clássicas.

A Slim-Tree faz o particionamento do espaço por meio da técnica de particionamento por bola. Sendo assim, na medida que novos elementos são inseridos na estrutura os nós das subárvores vão aumentando seus raios de cobertura. Com o aumento do raio, também aumenta a probabilidade de ocorrer sobreposição entre as subárvores, como apresentado na Figura 6. Essa sobreposição é o principal fator de degradação de desempenho da estrutura no momento de realização de consultas por similaridade. Isto porque contribui para diminuição do número de podas por meio da propriedade da desigualdade triangular (ver seção 2.3).

Para mensurar o grau de sobreposição nos quais as estruturas estão expostas foi proposto o conjunto de medidas Fat-Factor que determina o grau/fator de sobreposição entre os nós das subárvores (TRAINA-JR et al., 2000).

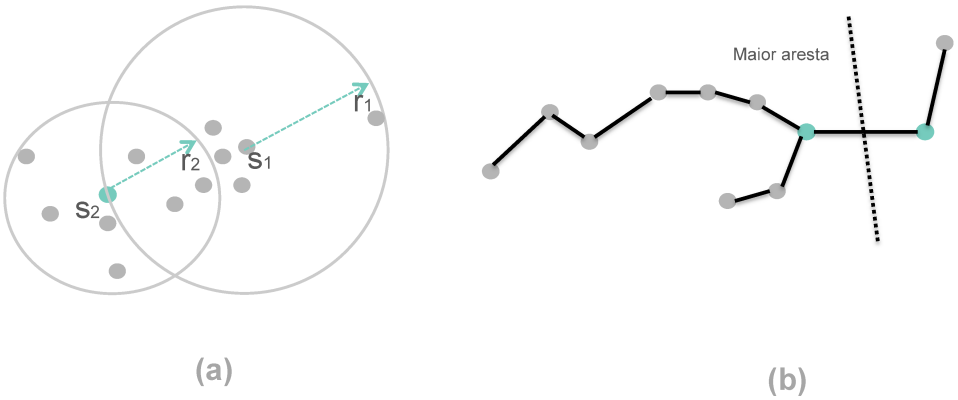

Figura 5 – Ilustração das políticas clássicas de divisão de nós. Em (a) política MinMax é representado a seleção de um par de elementos que são considerados candidatos e (b) política MST é representado a identificação da maior aresta a ser removida. Adaptado de (TRAINA-JR et al., 2000).

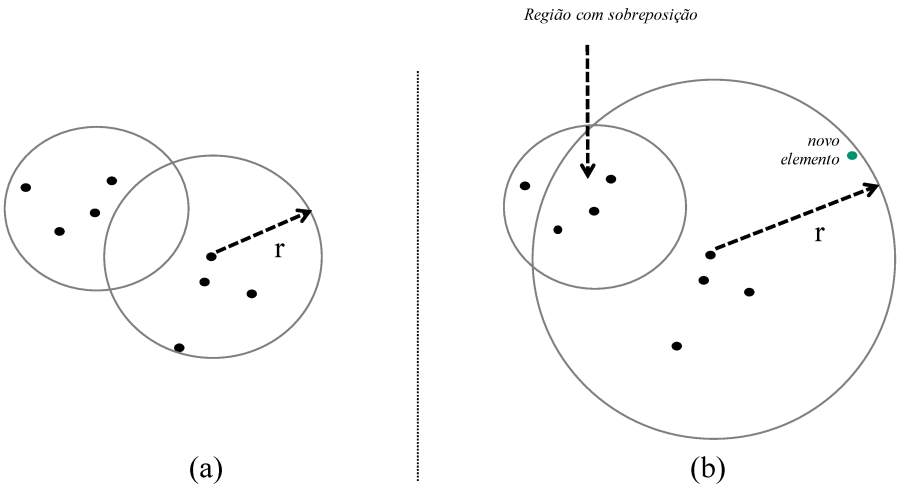

Figura 6 – Em (a) a estrutura sem sobreposição. Em (b) na cor azul está o novo elemento que chega ao modelo e, consequentemente, é representado o aumento do raio de cobertura que ocasiona a sobreposição nas subárvores.

2.4.4 O Fat-Factor

O Fat-Factor é um conjunto de medidas para avaliação do grau de sobreposição em que se permite realizar comparações entre árvores distintas (TRAINA-JR et al., 2000). Como não é possível calcular o volume das intersecções em espaços métricos genéricos, tomou-se a contagem de elementos nas intersecções como medida. Desse modo, o grau de sobreposição entre duas subárvores é dado por meio do número de elementos que se qualificam para pertencer a ambos os nós, divido pelo total de elementos de ambas as subárvores. É importante destacar que mesmo que os raios de cobertura se sobreponham caso não haja elementos na região de interseção, o valor do grau de sobreposição será igual a 0(zero), como representado na Figura 7 (b).

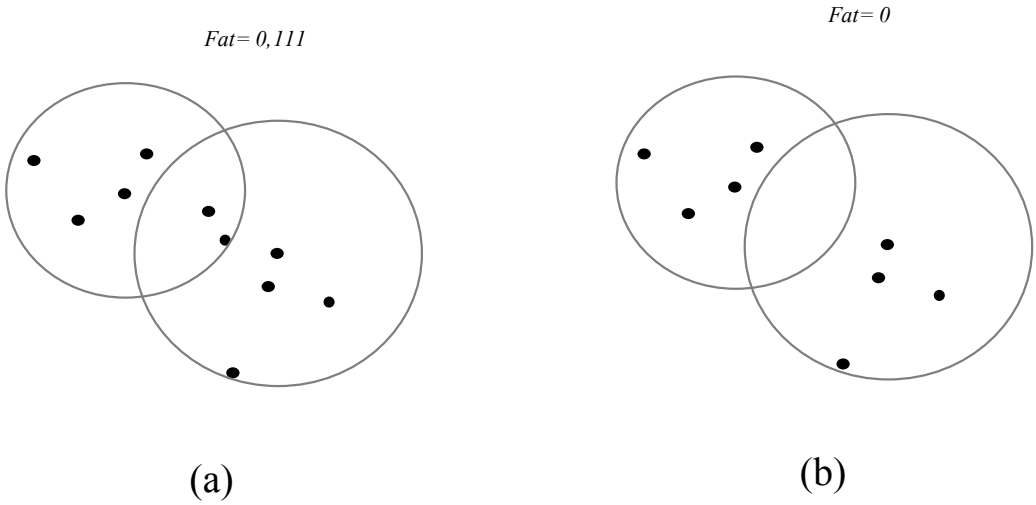

Figura 7 – Exemplo da aplicação do algoritmo Slim-Down em nó folhas. Em (a) temos dois nós antes a otimização e em (b) depois do processo de otimização.

O conjunto de medidas é dividido em **Fat-factor Absoluto** e **Fat-factor Relativo**.

O **Fat-Factor absoluto** é a medida geral de grau de sobreposição aplicado na comparação entre árvores que armazenam o mesmo conjunto de dados e com o mesmo número de nós. Essa medida vai avaliar o quão bom uma determinada árvore é em relação à sua sobreposição, independentemente de um possível desperdício de espaço em disco, decorrente de uma menor ocupação dos seus nós (TRAINA-JR et al., 2000). O valor resultante desta medida é um número que varia de [0, 1], sendo que 0 indica nenhuma sobreposição e 1 sobreposição total. Dado pela equação 5.

$$fat(T) = \frac{Ic - H.N}{N} \frac{1}{M - H} \tag{5}$$

onde, T é uma Slim-Tree, Ic é o número de acessos necessários para responder a uma consulta pontual, H é a altura árvore, M o número de nós, N o número total de elementos.

Nesse sentido, árvores que resultam em valores menores, contém menos regiões com interseções, implicado no fato de que ao realizar consultas, farão menos acesso a disco e menos cálculos de distâncias. Para árvores construídas por meio do mesmo conjunto de dados, porém que resultaram em diferentes números de nós (isso porque utilizou-se de diferentes algoritmos para *splitting* ou algoritmo de promoção), não é aplicado a mesma métrica. Isso porque, árvores com menor número de nós tendem a apresentar mais elementos em regiões de interseção, colaborando para gerar medidas de fat-factor com maior valor. Contudo, nesse caso o número de acesso a disco será menor, ou seja, menos nós a serem lidos, e consequentemente, melhora o tempo de execução do método. Assim sendo, a mesma medida fat-factor não seria justa para ser aplicada. Desse modo, para permitir comparações que expressam melhor a realidade, foi proposta a medida

Fat-Factor Relativo. Dado pela equação 6.

$$rfat(T) = \frac{Ic - Hmin.N}{N} \frac{1}{Mmin - Hmin} \tag{6}$$

em que, $Hmin$ denota a altura teórica mínima possível para a árvore, $Mmin$ o número teórico mínimo de nós, N é o número total de elementos e Ic é o número de acessos necessários para responder a uma busca pontal. Esta medida permite comparar duas árvores, considerando tanto o grau de sobreposições, quanto a ocupação eficiente dos nós. Sendo expresso por um o valor positivo maior que 0, o qual, possibilita mensurar que quanto menor o valor, menor será a sobreposição e, consequentemente, melhor será o desempenho da estrutura durante as consultas por similaridade.

2.5 Trabalhos Correlatos

Em diversos trabalhos encontrados na literatura correlata evidenciou-se o emprego de técnicas que têm como foco melhorar os mecanismos de particionamento (também conhecido como políticas de divisão de nós) quando os nós atingem sua capacidade máxima, como em (SOUZA; RAZENTE; BARIONI, 2014), ou ainda por meio de reinserções dinâmicas, como em (LOKOC; SKOPAL, 2008), no qual verifica-se a possibilidade de mover elementos entre os nós. Essas estratégias pretendem melhorar a distribuição hierárquica dos dados nas subárvores e auxiliar na redução sobreposição das estruturas. Desse modo, contribui diretamente para melhorar o desempenho das consultas, pois proporciona maior poder de poda por meio da propriedade da desigualdade triangular. Outras abordagens consistem em reorganizar as estruturas após a construção por meio de técnicas de *Bulk load* (VESPA; TRAINA-JR; TRAINA, 2010), proporcionando árvores mais compactas e, consequentemente, com menor grau de sobreposição. Uma outra abordagem recente é proposta em (OLIVEIRA; TRAINA-JR; KASTER, 2015) e se beneficia em otimizar o poder de poda das estruturas por meio da utilização de técnicas de adição de pivôs locais. Mas, as soluções apresentadas têm como principal desvantagem configurar-se como uma solução que ocorre localmente entre os nós. Dessa forma, apesar de trazerem bons resultados, durante a construção não são abrangentes.

Nesse sentido, uma solução global poderia ser adotada quando o grau de sobreposição das estruturas é elevado. Em (TRAINA-JR et al., 2000) foi proposto o algoritmo *Slim-Down* que consiste em reorganizar toda a estrutura de indexação. Esse algoritmo é aplicado sobre toda a árvore com o objetivo de reduzir a sobreposição entre os nós folhas. A ideia da técnica é mover pontos entre nós que sejam mais apropriados como finalidade de otimizar a estrutura ao garantir melhor distribuição dos dados. No processo de mover os elementos caso nenhum nó fique sem elementos nesse caso esse nó é removido. Além de reduzir a sobreposição entre os nós, contribui também para diminuir o número de subár-

vores, consequentemente, reduzindo o número de páginas de disco. A Figura 8 apresenta a ilustração da técnica proposta pelo algoritmo Slim-Down.

Apesar prover bons resultados, a abordagem tem um alto custo computacional, sendo ele $(O(nlogn) + O(n2))$, em que n é o tamanho da base de dados. Assim sendo, esse algoritmo não é adequado em base de dados que tem crescimento dinâmico (SKOPAL, 2006). E, por esse motivo, encontra-se na literatura uma série de trabalhos com propostas de métodos com custos computacionais menores, tais como (SKOPAL et al., 2003; LOKOC; SKOPAL, 2008).

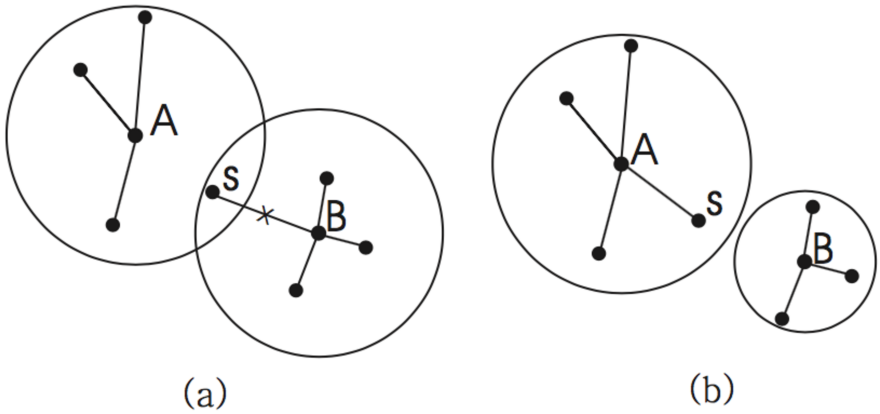

Figura 8 – Funcionamento do algoritmo *Slim-Down*. Em (a) Antes da aplicação (b) Após aplicação. Adaptado de: (TRAINA-JR et al., 2000).

No trabalho de Skopal (SKOPAL, 2006) (SKOPAL; LOKOČ, 2009) são propostos novas técnicas para a otimização do MAM M-tree por meio de reinserções forçadas. A reinserção forçada é uma técnica bem conhecida a partir de R*-tree (BECKMANN et al., 1990) em que a ideia é baseada na estratégia de remover alguns elementos de uma folha para outra com a finalidade de evitar uma operação de divisão e, em seguida, os elementos são reinseridos a folhas mais "adequadas". A motivação é a melhor ocupação do nó, ou seja, reinserções levam a nós mais completos. Além disso, funciona como o mecanismo que provê a oportunidade para mover alguns elementos "ruins" entre os nós folhas, ou seja, aqueles que provocam aumento da sobreposição nas estruturas a medida que novos elementos são inseridos. Os mesmos autores em (SKOPAL, 2006) apresentam quatros estratégias denominadas: *Pessimistic, Optimistic, Rev-pressimistic e Rev-optimistic*, que foram aplicadas no método M-Tree e permitiu a obtenção de árvores mais compactas. O método de comparação foi a própria estrutura M-tree padrão e apresentou melhores desempenhos na execução das consultas por similaridade. Apesar dos resultados, uma das desvantagens do emprego destas estratégias é a necessidade de limitar o número de reinserções. Isso porque, quanto maior o número de reinserções forçadas maior será custo de construção. Na Figura 9 apresenta uma ilustração da técnica proposta para reinserção forçada.

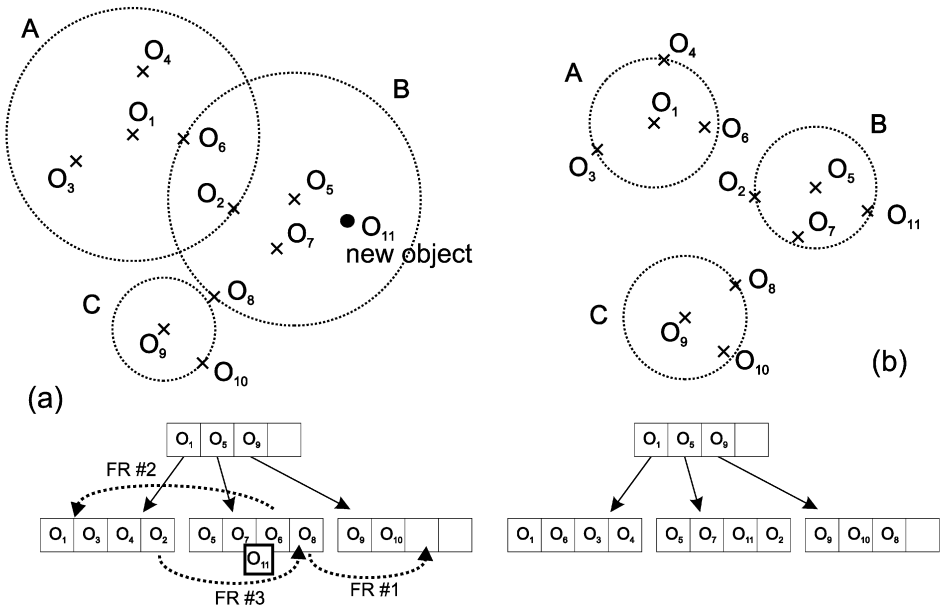

Figura 9 – (a) Antes da reinserção (b) Diminuição da sobreposição após 3 reinserções. Fonte: (SKOPAL; LOKOČ, 2009).

No trabalho de (SOUZA; RAZENTE; BARIONI, 2014) são propostas cinco novas heurísticas para as políticas de divisão de nós, com a finalidade melhorar a distribuição dos elementos nos nós das subárvores e assim otimizar o desempenho dos métodos de acesso métricos durante o processo de construção e consulta por similaridade. A ideia central consiste em melhorar a distribuição dos elementos entre os nós folhas e, consequentemente, reduzir o grau de sobreposição. Desse modo, foram propostas cinco novas políticas para melhorar o resultado do particionamento de nós, são elas: DM (Máxima Dissimilaridade), SDC (Soma das distâncias dos Caminhos), AEC (Agrupamento dos elementos no Caminho), ER (Elemento de Referência) e DE (Distribuição Equilibrada). Essas propostas foram avaliadas e comparadas com a estrutura Slim-Tree combinadas as suas políticas clássicas Aleatório, MinMax e MST (anteriormente apresentadas na Seção 2.4.2).

Os resultados encontrados são empíricos e foram validados experimentalmente, de modo a comprovar que o emprego das novas políticas, no processo de construção dos métodos provem gerar árvores com menores graus de sobreposição, comprovados pelo uso da medida fat-factor absoluto e relativo (ver Seção 2.4.4). Tais resultados contribuíram positivamente para otimizar o processo de construção das estruturas além de melhorar a performance das consultas por similaridade. Dentre as políticas propostas, a que apresentou melhores resultados com relação a tempo de construção para dados com alta dimensionalidade foi a política DM e, por esse motivo, esta será utilizada como parâmetro de comparação neste trabalho. A seguir será detalhado o funcionamento da política DM (Dissimilaridade Máxima). Essa abordagem parte do pressuposto que é possível obter nós com grau de sobreposição menor ao se dividir o conjunto de dados entre dois elementos

mais dissimilares. A complexidade de tempo é $O(n^2)$, em que n é a quantidade de elementos presentes no nó. Destaca-se que a complexidade é quadrática, inferior a da política MinMax ($O(n^3)$). Na Figura 10 pode-se observar uma ilustração da técnica proposta para divisão de nós.

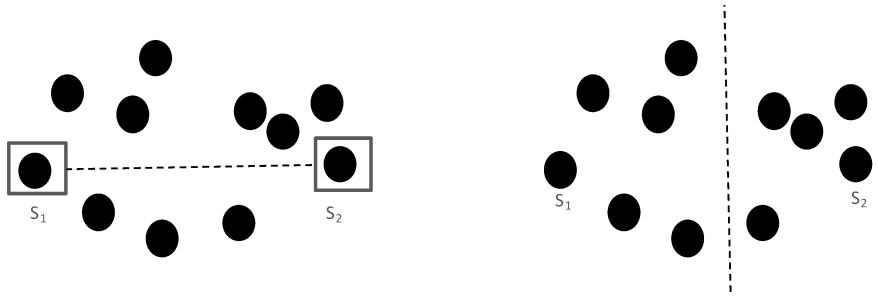

Figura 10 – Política de divisão de nós DM. Separa em dois grupos dos quais o par de elementos selecionados tenha a máxima distância. Adaptado de (SOUZA; RAZENTE; BARIONI, 2014).

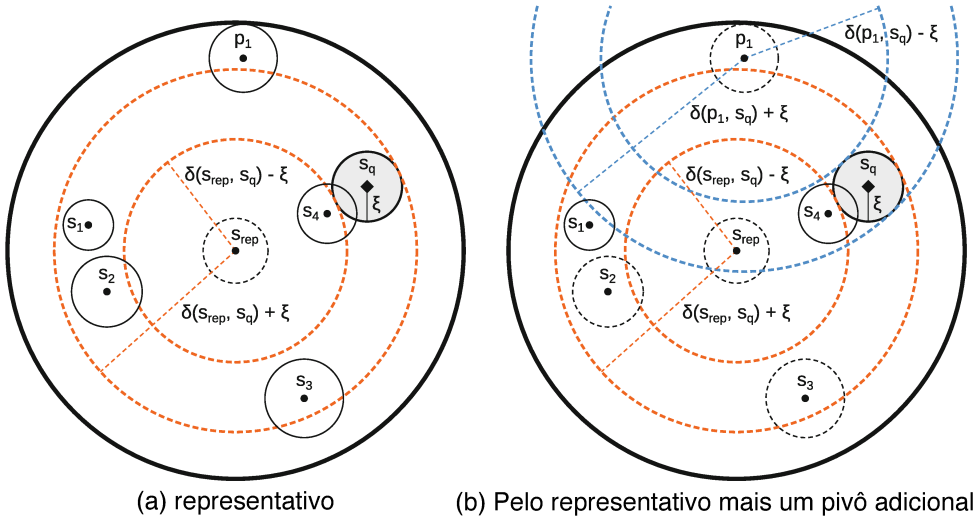

Figura 11 – Ilustração da poda da desigualdade triangular com um pivô local adicional. As circunferências pontilhas representam os elementos descartados. Fonte: (OLIVEIRA; TRAINA-JR; KASTER, 2015).

Com a mesma finalidade de acelerar consultas por similaridade sem comprometer a dinamicidade da estrutura, no trabalho de Oliveira (OLIVEIRA; TRAINA-JR; KASTER, 2015) são propostas duas novas técnicas que contribuem para melhorar o poder de poda em MAMs dinâmicos. A primeira utiliza-se de pivôs locais adicionais para reduzir os cálculos de distâncias e a segunda faz uso de informações antecipadas dos nós filhos para reduzir o número de acesso a disco desnecessários. A abordagem proposta foi avaliada e comparada com a Slim-tree e resultou em desempenhos satisfatórios no momento da realização de consultas por similaridade. No entanto, destaca-se como limitação da proposta o fato

Tabela 1 – Principais trabalhos que norteiam as estratégias de construção eficiente MAMs dinâmicos. Adaptado (SOUZA; RAZENTE; BARIONI, 2014).

Estratégia	**Trabalho**	**Tipo de Estratégia**
Slim-Down	(TRAINA-JR et al., 2002a)	Estático
Política de divisão de Nós	(LIM et al., 2006)	Dinâmico
Operação de *Bulk load*	(SKOPAL et al., 2003)	Estático
Operação de *Bulk load*	(SKOPAL, 2006)	Estático
Reinserção Dinâmica	(LOKOC; SKOPAL, 2008)	Dinâmico
Operação de *Bulk load*	(VESPA; TRAINA-JR; TRAINA, 2010)	Estático
Política de Divisão de nós	(SOUZA; RAZENTE; BARIONI, 2014)	Dinâmico
Pivôs locais Adicionais	(OLIVEIRA; TRAINA-JR; KASTER, 2015)	Dinâmico

do custo para a construção do método ser comprometido. A Figura 11 apresenta uma ilustração da técnica proposta com adesão de novos de pivôs locais.

Por fim, na Tabela 1 é representado um resumo dos principais trabalhos correlatos que nortearam o desenvolvimento desta dissertação. É possível notar que as técnicas foram criadas sobre duas perspectivas distintas, ou seja, o tipo de técnica do qual pode ser considerada dinâmica ou estática. A primeira permite otimizar dinamicamente os métodos, tais como, apresentados nos trabalhos (OLIVEIRA; TRAINA-JR; KASTER, 2015), (SOUZA; RAZENTE; BARIONI, 2014) e (LOKOC; SKOPAL, 2008), ou seja, a cada elemento sendo inserido no índice a técnica é aplicada. Já a segunda permite melhorar estaticamente um índice criado (pós-processamento), tais como, apresentados nos trabalhos de (TRAINA-JR et al., 2002a), (VESPA; TRAINA-JR; TRAINA, 2010), ou seja, nessa abordagem à partir de um índice criado, em um dado momento, é possível executar o algoritmo e melhorar a qualidade do índice. Com a inserção de novos elementos em caso do índice degradar é possível executar o algoritmo sobre o índice todo novamente.

2.6 Considerações

Neste capítulo foram apresentados conceitos fundamentais com relação às consultas por similaridade em métodos de acesso métrico, dos quais destacaram-se os métodos dinâmicos. Além disso, evidenciou-se um dos problemas ao se fazer uso do particionamento do espaço por bola, isto porque, no decorrer de novas inserções de elementos na estrutura de indexação pode ocorrer degradação devido à sobreposição das subárvores, fato que impacta diretamente no desempenho dos métodos.

Nesse sentido, é de suma importância mensurar e avaliar o grau de sobreposição ao qual os MAMs estão expostos, pois uma ampla gama de aplicações lida com base de dados dinâmicas e necessitam indexar e recuperar conjuntos de dados complexos, independentemente de suas particularidades, como volume de dados, dimensionalidade, cardinalidade e distribuição dos dados.

Foram apresentadas também no capítulo algumas das principais propostas encontradas na literatura das quais algumas baseiam-se em melhorar a particionamento dos nós folhas, quando estes estão cheios ou prestes a atingirem sua capacidade máxima. E outra técnica que utiliza pivôs locais para que permitam melhorar o poder de poda e assim melhorar o desempenho do método durante o processo de busca por similaridade. Contudo, apesar dos esforços empregados para minimizar a sobreposição, ainda há lacunas a serem resolvidas, principalmente porque existe um *trade off* com relação a melhora do desempenho das consultas e o tempo necessário para a construção dos métodos. Muitas das técnicas propostas conduzem a melhores resultados durante o processo de recuperação por similaridade, contudo, a construção do método é comprometida. Outro fator a ser considerado consiste no fato de que essas soluções têm como principal desvantagem configurar-se como uma solução que ocorre localmente entre os nós. Dessa forma, apesar de trazer bons resultados ao reduzir a sobreposição, essas soluções não são abrangentes.

Por fim, foi apresentado com detalhes o conjunto de medidas fat-factor relativo e absoluto que serão utilizadas para avaliação do grau de sobreposição.

Capítulo 3

Abordagens para Indexação Assistida por *short-term memory* com Capacidade Limitada

Este capítulo apresenta a estratégia desenvolvida nesta dissertação, que consiste em utilizar-se da estrutura *short-term memory* para auxiliar o processo de construção MAM dinâmicos otimizados. Na Seção 3.1 é apresentado o contexto da utilização da estrutura *short-term memory*. Na Seção 3.2 são descritas as contribuições aplicadas aos algoritmos responsáveis pelo processo de inserção Padrão MAM Slim-Tree e, em seguida são apresentados na Seção 3.2.1 os novos algoritmos desenvolvidos especificamente para inserção de nós folhas compostos por um representante e raio de cobertura. Na Seção 3.2.2 são descritos os algoritmos utilizados como estratégia para agrupar dados e criar nós folhas com elementos mais similares. Por fim, na Seção 3.3 são feitas as considerações sobre o capítulo.

3.1 Contexto de utilização da estrutura *short-term memory*

No contexto de indexação de dados complexos, as estruturas métricas dinâmicas tais como a M-tree e variantes são avaliadas de acordo com sua eficiência, sendo o tempo o principal parâmetro a ser considerado. Nesse caso, o tempo necessário para a construção e o tempo necessário para realizações de operações de busca por similaridade. As estruturas, como as citadas anteriormente, utilizam-se de armazenamento em memória secundária. Desse modo, têm o tempo diretamente ligado à quantidade necessária de acesso a páginas de disco e ao seu número de cálculos de distância.

Além disso, a eficiência ainda é impactada diretamente pela sobreposição existente entre os nós das subárvores. Isso porque, como visto na Seção 2.4.2, não é possível a

realização de podas no momento da busca colaborando para que um maior número de nós tenha que ser verificado, logo, ocasionando um aumento do número de cálculos de distância e de acessos a páginas de disco.

Sendo assim, torna-se primordial a investigação e aplicação de estratégias para construção de MAMs que minimizem a sobreposição resultante nas estruturas métricas. Para tratar o problema relacionado à sobreposição, na literatura são propostos algoritmos para a otimização das estruturas. Dentre estes, como visto na Seção 2.5, são propostos algoritmos para melhorar os mecanismos de particionamento quando os nós estão quase cheios (reinserções forçadas) ou ainda quando estes atingem sua capacidade máxima (políticas de divisão de nós), além disso, é proposta a adição de pivôs locais adicionais para melhorar o poder de poda da desigualdade triangular. No entanto, tais estratégias, de modo geral, tentam solucionar um problema que já aconteceu e sua solução ocorre localmente entre os nós. Além de que algumas destas abordagens comprometem o tempo necessário para a construção dos índices.

A abordagem proposta nessa dissertação fundamenta-se em suposições de adiamento da indexação com a finalidade de inserir no índice novos objetos compostos com distâncias próximas. A estratégia foi baseada em conceitos das áreas de *data streaming*, mais especificamente em soluções aplicadas à temática de classificação de dados. Em tais problemas, com a chegada de um novo elemento ao modelo classificador, este modelo deve ter a capacidade de classificar o elemento entre as classes já conhecidas ou ainda descartá-lo em caso desse não pertencer a nenhuma classe.

Para a utilização da abordagem de adiamento, aplica-se como uma solução temporária com o objetivo de mover os elementos que não possuem classe definida na estrutura auxiliar, que posteriormente deverá ser processada. A justificativa é prover a possibilidade de elementos que seriam descartados virem a receber elementos da mesma classe e formar uma nova classe, que será reconhecida como novidade e incorporada ao modelo de classificação, ideia similar apresentada no trabalho (FARIA et al., 2016a).

No contexto de indexação de dados serão movidos para uma estrutura auxiliar os elementos que produzem aumento dos raios de cobertura das subárvores. A ideia é conter os efeitos da sobreposição que causam degradação dos métodos. Sendo assim, inicia-se o trabalho por meio de uma pré-seleção temporária dos elementos a serem indexados na estrutura métrica, desse modo, é possível antecipar a informação de elementos que produzem aumento imediato aos raios das subárvores. Sendo assim, migram-se os “maus“ elementos que seriam inseridos no índice para a estrutura auxiliar *short-term memory*.

Dessa maneira, o processo tem como fatores positivos prover um ambiente mais abrangente para o processamento de elementos quando comparado ao processo que ocorre localmente entre nós e, colaborando para a descoberta de grupos de elementos com distribuição mais uniforme. Além disso, configura-se como um filtro temporário para reter os efeitos de elementos considerados *outliers* que provocam aumento imediato dos raios causando

mais sobreposição nas estruturas.

O uso da estratégia no que se refere a indexação e recuperação por similaridade fundamenta-se em prover raios de cobertura menores e mais compactos e, consequentemente, com menor grau de sobreposição por meio da melhoria da distribuição dos elementos entre os nós folhas. Por conseguinte, faz-se necessário criar e realizar inserções de nós folhas completos compostos por elementos mais similares na estrutura métrica. Assim, a estratégia contempla que os nós terão capacidade para receber novas inserções de elementos dentro da sua região de cobertura devido à fragmentação resultante da taxa de ocupação.

3.2 Contribuições às Estruturas de Indexação Métricas

Nessa seção apresenta-se as contribuições estruturais decorrentes do uso da *short-term memory* com capacidade limitada na abordagem de construção dos métodos de acessos métricos. A estrutura auxiliar foi implementada em memória principal (RAM) e armazena temporariamente apenas os elementos os quais não foram cobertos por algum nó folha. Sua estrutura está representada na Figura 12.

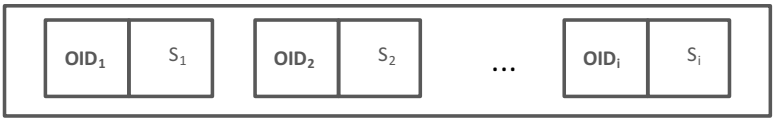

Figura 12 – Representação da estrutura *short-term memory*. Onde **OID***i* é o identificador do elemento; **s***i* é o próprio elemento, armazenado como um vetor de características.

O processo de inserção de um novo elemento s_n no MAM Slim-tree é feito do seguinte modo:

i) Verifica se há raiz, caso não houver cria o nó raiz e insere o elemento s_n;

ii) Caso a raiz tenha sido criada é necessário realizar uma travessia da raiz até as folhas que é feita por meio do algoritmo *InsertRecursive* que emprega o algoritmo *Choosesubtree* para escolher em qual subárvore descer;

iii) Caso nenhum nó esteja qualificado para armazenar s_n, então é escolhido o nó que tenha a distância mínima entre s_n e elemento representante deste nó, e o seu raio de cobertura deverá ser aumentado;

iv) Caso haja empate e mais de uma região seja candidata a receber s_n, o algoritmo *Choosesubtree* determina em qual página inserir de acordo com o critério estabelecido;

v) Caso haja *overflow*, ou seja, o nó atinge a capacidade máxima, um algoritmo de divisão é empregado, gerando uma nova folha, resultando na distribuição dos elementos entre o nó corrente e o novo nó. A inserção na folha pode propagar alterações recursivamente até a raiz. Esse mecanismo é conhecido como construção *bottom-up* e garante que a estrutura permaneça sempre balanceada (TRAINA-JR et al., 2000).

Como relação ao processo de inserção, especificamente no passo iii), ao aumentar o raio de cobertura, a medida que novos elementos são inseridos, muitas regiões vão sofrendo degradação pelo fato de ocorrer sobreposição entre as subárvores. Desse modo, a estratégia definida nesta dissertação propõe uma modificação no passo descrito, e passa a configurar-se como uma abordagem alternativa para processo de inserção de um elemento quando este gera aumento do raio de cobertura. Ela parte da suposição que ao permitir que um novo elemento, que não esteja coberto por um nó folha, aguarde na *short-term memory* para se agrupar com elementos próximos para formar um nó folha, contribuindo, assim, para minimização dos raios de cobertura e, consequentemente, evitar a sobreposição decorrente das estruturas resultantes, como apresentado na Figura 13.

Uma outra abordagem para minimizar a sobreposição seria um pós-processamento da estrutura para reconstruir a árvore considerando a similaridade entre todos os elementos o que certamente traria bons resultados, entretanto, o custo é proibitivo.

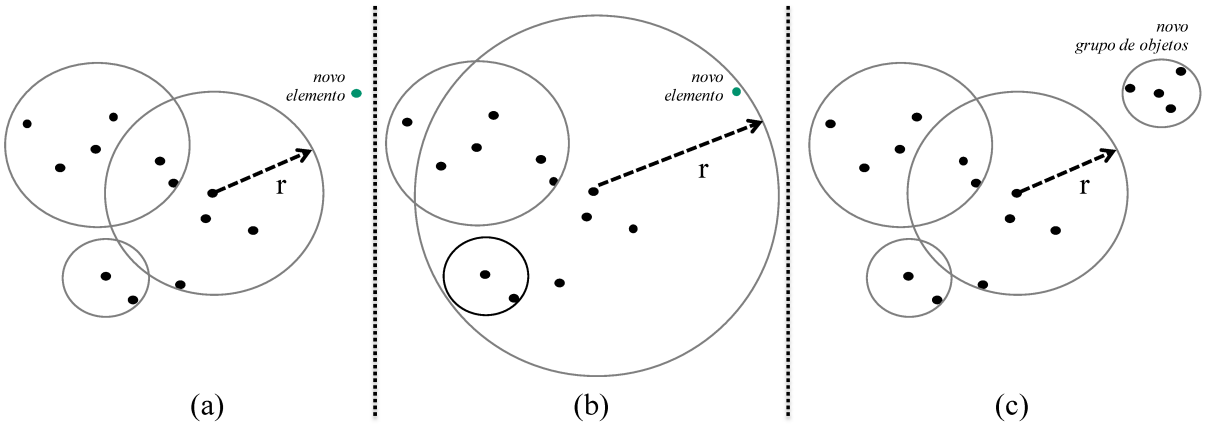

Figura 13 – Ilustração do comportamento dos métodos de acesso no que se refere ao processo de indexação de elementos. Em (a) Slim-Tree padrão, em (b) Slim-Tree ao receber um novo elemento resultou no aumento do raio de cobertura (r) e em (c) estruturas derivadas após aplicação da estratégia de adiamento.

No algoritmo padrão de inserção foi adicionado uma nova condição a ser verificada no passo iii), pois se a operação de inclusão no nó folha resultar em um aumento do raio da subárvore, o elemento deverá ser inserido na *short-term memory.* Além disso, adicionou outro passo que dever ser realizado após o retorno da recursão. Ou seja, após a inserção do novo elemento seja no índice ou na estrutura auxiliar, será necessário realizar uma avaliação para determinar se a *short-term memory* encontra-se com a capacidade

máxima. Caso verdadeira essa condição o método *ProcessShortMemory* (Algoritmo 1) é executado e irá processar os elementos de acordo com os parâmetros definidos pelo usuário no momento da criação do método.

Algoritmo 1: Processa Short-term Memory (Novo Método)

```
1  shortmemory ← conjuto de elementos presentes na short-term memory
2  c ← tamanho da partição
3  t ← taxa de ocupação do nó
4  f ← função de distância
5  I ← número de interações
6  metodo ← método escolhido para agrupamento
7  se metodo == "Random" então
8  |  newLeaf ← Random(shortmemory,c,t,f)
9  |  AddLeaf(newleaf)
10 fim
11 se metodo == "Density" então
12 |  newLeaf ← Density(shortmemory,c,t,f)
13 |  AddLeaf(newleaf)
14 fim
15 se metodo == "Cluster" então
16 |  newLeaves ← Cluster(shortmemory,c,t,f,I)
17 |  k ← quantidade de novas folhas
18 |  para i ← 0 até k − 1 faça
19 |  |  AddLeaf(newLeaves[i])
20 |  fim
21 fim
```

O Algoritmo 1 foi desenvolvido especificamente para o processamento dos dados da *short-term memory*. Nas linhas 7, 11 e 15 são testadas as condições para averiguar qual a estratégia escolhida para criação dos nós folhas. Nas linhas 9, 13 e 19 é chamado o Algoritmo 2 para inserção do nó *AddLeaf*, sendo este algoritmo similar ao proposto na Slim-Padrão, contudo, passa como parâmetro para os demais algoritmos envolvidos no processo de inserção um nó folha com representante e raio de cobertura definidos e não apenas um elemento.

Como resultado da aplicação do Algoritmo 1, um novo nó folha ou ainda um conjunto de nós folhas serão geradas de acordo com a abordagem adotada para agrupamento dos elementos, detalhados na Seção 3.2.2 , e respeitando a taxa de ocupação definida a priori. Em resumo as seguintes etapas são executadas quando a capacidade da *short-term memory* é atingida:

i) criação de um conjunto de nós folhas de tamanho fixo, onde cada nó folha terá um elemento representante (rep) e um raio de cobertura (r_i).

ii) remoção da *short-term memory* dos elementos dos nós folha gerados no passo i).

iii) inserção dos novos nós na estrutura de indexação.

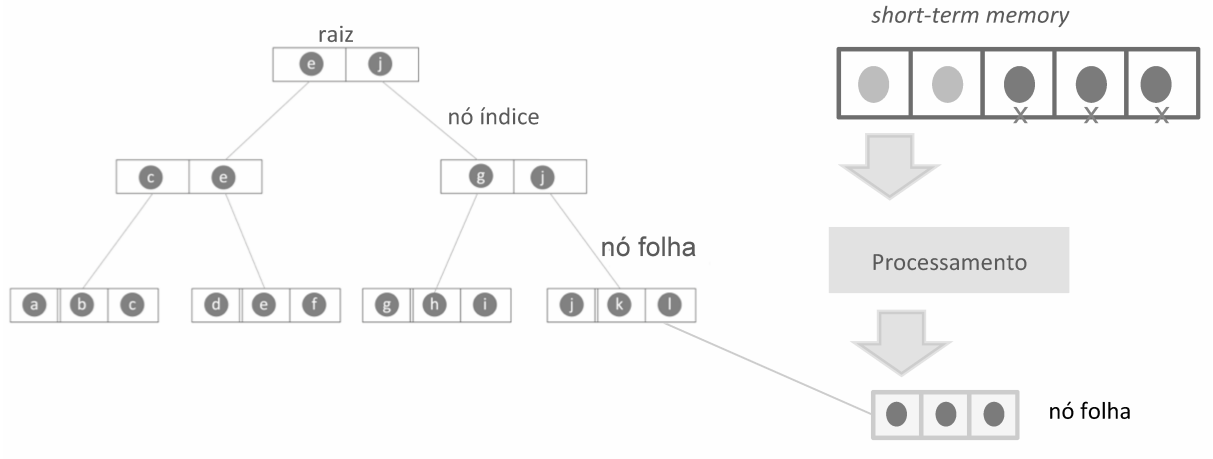

Figura 14 – Representação do processamento dos elementos presentes na *short-term memory* com inclusão de um nó folha na estrutura de indexação. Os algoritmos envolvidos no processo de inserção tiveram que ser reescritos, pois agora eles passam como parâmetro um nó folha completo, com centro (elemento representante do nó) e raio de cobertura.

3.2.1 Inserção de nó folha completo na estrutura de indexação

Para inserção dos nós folhas completos derivados da *short-term memory* foi necessário o desenvolvimento de três novos algoritmos. O primeiro para o processo de inserção denominado, *AddLeaf*, o segundo para navegar recursivamente na árvores *InsertLeafNode* e o terceiro para a escolha em qual ramo da árvore vai descer, denominado *ChooseLeafsubtree*. Tais algoritmos diferiam-se em alguns aspectos dos algoritmos padrões de MAM Slim-Tree, pois agora eles têm como parâmetro um nó folha completo, com um elemento representante do nó e raio de cobertura.

No Algoritmo 2 (*AddLeaf*) na linha 2 é verificado a condição do nó atual não ser raiz e caso verdadeira essa condição é chamado o Algoritmo 3 (*InsertLeafRecursivo*) que irá realizar o percurso recursivo até chegar na subárvore mais adequada para receberá o novo nó folha. Após o retorno da função na linha 4 é testado a condição para verificar se ocorreu ação do tipo Promoção e, em caso da condição ser verdadeira, chama o algoritmo *AddNewRoot* passando como parâmetros as informações para promover os elementos representantes daquele nível da árvore e esse processo irá ocorrer recursivamente para os demais níveis.

Na linha 2 é chamado o Algoritmo 4 (*ChooseLeafSubtree*) que irá escolher em qual índice deve descer na árvore. Na linha 17 é apresentado o momento em que se chegou ao nó que deve ser inserido o novo elemento. Desse modo, é necessário promover o

Algoritmo 2: Algoritmo de Inserção de Nó Folha. ADDLEAF (Novo Método)

Entrada: newLeaf: completo com *rep* (representante) e *r* raio de cobertura
Saída: status: True se inserção foi realizada ou false caso não seja

```
1 CurrNode ← recebe as informações sobre o nó corrente
2 se CurrNode.type != 0 então
3 |   acao ← executa o algoritmo InsertLeafRecursivo passando como parâmetro o
  |     índice corrente CurrNode, o novo newLeaf e informações sobre a subárvores
  |     em promo1 e promo2.
4 |   se acao == PROMOTION então
5 |   |   executa o algoritmo AddNewRoot passando com parâmetros as informações
  |   |     sobre o par de elementos representantes
6 |   fim
7 fim
```

elemento representante do novo nó, juntamente com seu raio de cobertura. Com relação a linha 12 quando a condição de "PROMOTION" é satisfeita então faz-se necessário promover o índice. Desse modo, certifica-se as condições das quais irão desencadear em processos recursivos para manutenção dos índices. Será possível realizar atualização nos raios de cobertura das subárvores, nos elementos representantes, atualização de entradas de elementos dos índices e operações de *slipt* (divisão de nós) para criação de novos nós. Tais operações estão inseridas nas ações denominadas "NO_ACT", "CHANGE_REP" ou "PROMOTION".

O Algoritmo 4 *ChooseLeafSubtree* foi desenvolvido especificamente para encontrar o índice que irá cobrir nó folha. Nas linhas 6 - 22 é imposto um *loop* que deve ser realizado para cada entrada de índice do nível atual. As linhas 9 e 15 cobrem as condições que devem ser satisfeitas para a inserção do nó folha.

Após satisfeitas estas condições será eleito o índice que irá receber o nó folha completo, ou seja, aquele que tiver a menor distância. A condição foi baseada na heurística que utiliza o critério *mindistance* utilizando como padrão para a construção do MAM Slim-Tree.

Contudo a diferença consiste em calcular a mínima distância não apenas para um único elemento mas para o nó folha completo de acordo com a distância para o representante e considerando o raio de cobertura do mesmo (linhas 9 e 15 do O Algoritmo 4).

Por fim, foram descritos as modificações e novos algoritmos criados que fazem parte do processo de inserção, seja de um elemento ou de um nó folha completo. Na Figura 15 é apresentado um resumo de todo processo realizado durante a construção do método de indexação.

Na Seção 3.2.2 serão detalhas as estratégias empregadas para a agrupar os elementos da *short-term memory*.

Algoritmo 3: InsertLeafRecursivo (Novo Método)

Entrada: identificador da página $CurrNode$, novo nó folha $newLeaf$ e informação do nó correntes $promo1$ e $promo2$.
Saída: acao

```
1  se CurrNode.type == "INDEX" então
2  |  subtree ← ChooseLeafSubTree(CurrNode,newLeaf)
3  |  acao ← InsertLeafRecursive(subtree.PageID, newLeaf, promo1,promo2)
4  |  se acao == "NOACT" então
5  |  |  atualiza raio e conta quantidades de entradas
6  |  |  ....
7  |  fim
8  |  se acao == "CHANGEREP" então
9  |  |  troca elemento representativo
10 |  |  ....
11 |  fim
12 |  se acao == "PROMOTION" então
13 |  |  atualiza os índices da subárvore
14 |  |  ....
15 |  fim
16 fim
17 senão
18 |  LeafNode ← recebe informações sobre o nó folha como quantidades de
   |   objetos, representante e raio de cobertura.
19 |  promo1 ← recebe todas as informações sobre LeafNode
20 |  promo2 ← recebe todas as informações sobre newLeaf
21 |  acao← "PROMOTION"
22 fim
23 return acao
```

3.2.2 Abordagens para agrupamento de dados

A técnica permite que os elementos se agrupem com outros elementos próximos formando uma nova folha, que tem capacidade para receber novas inserções de elementos dentro da sua região de cobertura devido à fragmentação resultante da taxa de ocupação. Além disso, os nós folhas criados a partir da SM não apresentam estrutura de informações diferentes dos gerados pelo processo padrão da Slim-Tree. Ou seja, o nó folha apresenta a mesma estrutura, tal como ilustrado na Figura 16.

Para realizar o agrupamento dos dados são propostos três abordagens distintas que são ilustradas na Figura 17.

3.2.2.1 Abordagem *Random*

O agrupamento chamado *random* consiste em selecionar aleatoriamente um elemento da *short-term memory*, e elegendo-o como elemento representante do nó. E a partir desse elemento é computada a distância para todos os demais elementos do conjunto. Serão

Algoritmo 4: Algoritmo ChooseLeafSubTree (Novo Método)

Entrada: newLeaf: nó folha completo
Saída: idx: Identificação do Índice

```
1  newLeafRep ← recebe informações sobre objeto representante e quantidades de
     elementos no nó
2  newleafRadius ← recebe informação sobre o raio de cobertura do nó
3  mindistance ← +∞
4  numberOfEntries ← quantidades de entradas do nó índice
5  stop = (idx >= numberOfEntries)
6  while (!stop) do
7  |  CurrIndexNode ← informações sobre índice corrente
8  |  distance ← GetDistance(CurrIndexNode, newleafRep)
9  |  se distance < CurrIndexNode.Radius + newLeafRadius então
10 |  |  minDistance = distance
11 |  |  stop = true
12 |  |  minIndex = idx
13 |  fim
14 |  senão
15 |  |  se distance - CurrIndexNode.Radius - newLeafRadius < minDistance então
16 |  |  |  minDistance = distance - CurrIndexNode.Radius - newLeafRadius
17 |  |  |  minIndex = idx
18 |  |  fim
19 |  fim
20 |  idx ++
21 |  stop = stop || (idx >= numberOfEntries)
22 |  return minIndex
23 end
```

selecionados os elementos mais similares, ou seja, tenham suas distâncias mais próximas ao representante para compor um nó folha de tamanho fixo, de acordo com as restrições impostas pela taxa de ocupação e tamanho da página de disco. E por fim, este nó será inserido na estrutura de índice durante a fase de construção. A abordagem é representada pelo Algoritmo 5. Essa estratégia será utilizada em conjunto com uma política de divisão de nós para a construção da estrutura métrica denominada SMRandom (SMR).

3.2.2.2 Abordagem *Density*

O agrupamento chamado *density* é uma heurística baseada no *random*, a ideia é encontrar um conjunto de elementos ao redor de um representante cuja distância dos elementos para o representante seja mínima, de maneira, a não comprometer a construção dos métodos. A diferença é que utiliza-se de uma constante para configurar o número de iterações em que será selecionado aleatoriamente o representante do nó. Com essa abordagem, é possível selecionar entre um conjunto de possíveis nós folhas aquele que tenha o seu raio de cobertura minimizado. O funcionamento é apresentado em Algoritmo 6. Essa estraté-

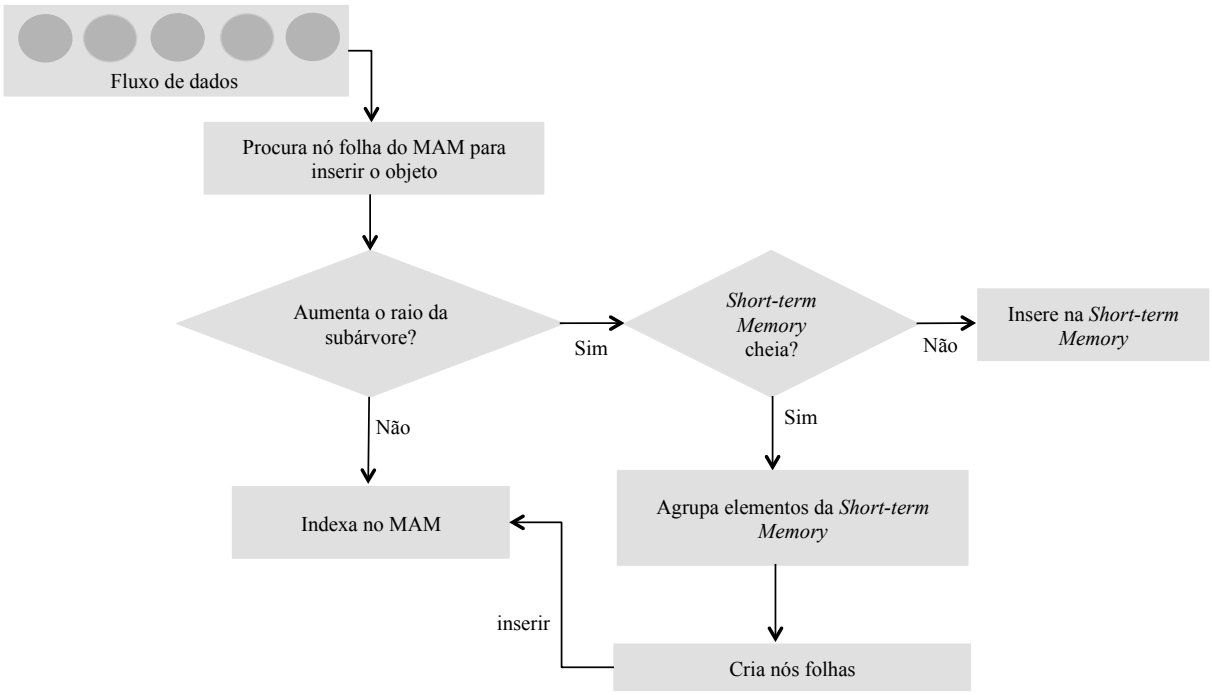

Figura 15 – Fluxo do processo de construção das estruturas métricas assistidas por *short-term memory*. Note que a diferença está no passo em que a inclusão de um novo elemento resulta em um aumento do raio da subárvore. Desse modo, a abordagem proposto é inclusão do elemento na *short-term memory* temporariamente até que esta tenha sua capacidade máxima atingida e seja necessário processar os nós.

Figura 16 – Representação da estrutura do nó folha completo, onde **OID**i é o identificador do elemento, $\mathbf{s}i$ é o próprio elemento, armazenado como um vetor de características e $\delta(si, rep)$ é a distância entre $\mathbf{s}i$ e o representante.

gia será utilizada em conjunto com uma política de divisão de nós para a construção da estrutura métrica denominada SMDensity (SMD).

3.2.2.3 Abordagem *Cluster*

A abordagem nomeada *cluster* utiliza-se do algoritmo Clarans (*Clustering Large Applications basedon Randomized Search*) proposta em (NG; HAN, 2002), assim como apresentado no Algoritmo 8. A ideia é gerar um conjunto de k agrupamentos. O algoritmo retorna um conjunto de k medoids encontrados. Cada medoid desse conjunto será eleito o elemento representante de uma página. E a partir de cada representante é computado a distância para todos os demais elementos do conjunto. E, então seleciona-se os elementos

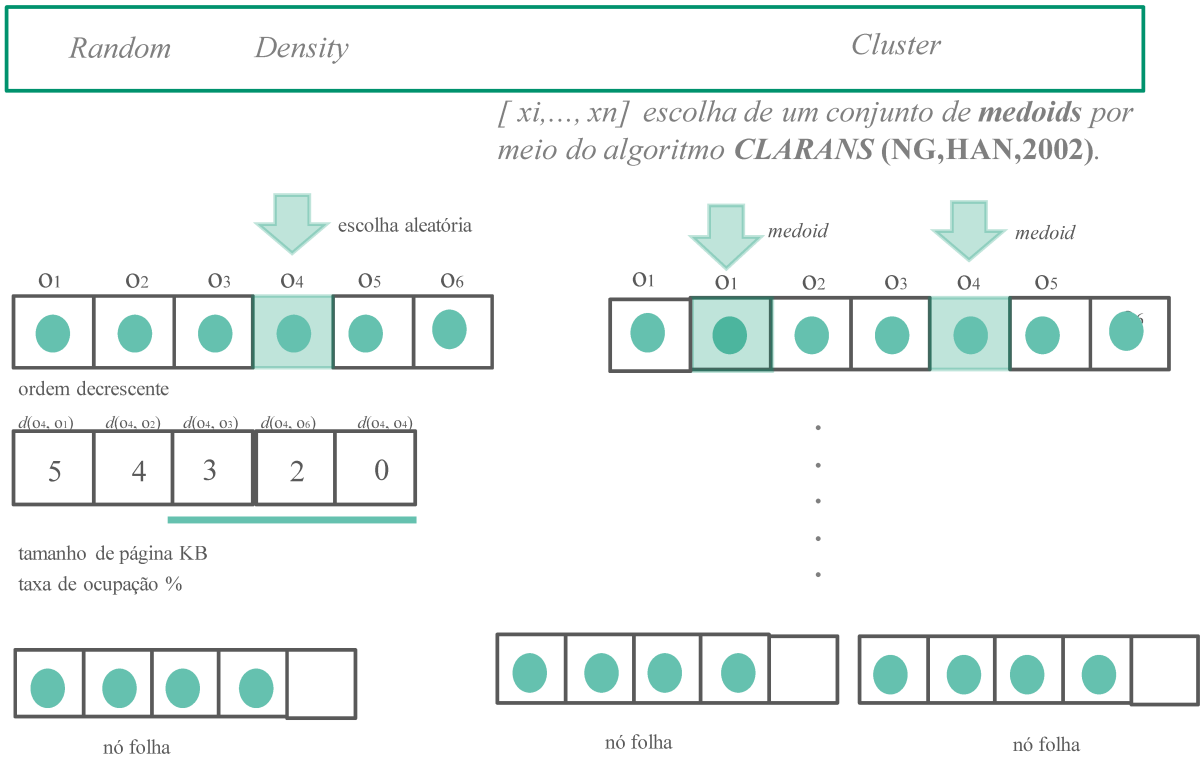

Figura 17 – Representação das técnicas empregadas para agrupamento dos dados provenientes da *short-term memory*.

Algoritmo 5: ABORDAGEM PARA AGRUPAMENTO *Random*

Entrada: Conjunto de elementos S, tamanho partição c, taxa de ocupação t e métrica $dist()$
Saída: Subconjunto de elementos s

1 aleatoriamente selecione um elemento $x_i \in S$ como representante rep
2 **para** $x_i \in S$ **faça**
3 $\quad d[i] \leftarrow dist(rep, x_i)$
4 **fim**
5 ordene d e S de acordo com a distância para o representante rep em ordem decrescente
6 $s \leftarrow$ selecione um subconjunto $x_1, x_2, ..., x_n \in$ d, tamanho c e limitado por t
7 remova os $s \in S$
8 retorne s

mais próximos de cada medoid para formar cada uma das k páginas folhas até o limite da ocupação definida. Essa estratégia será utilizada em conjunto com uma política de divisão de nós para a construção da estrutura métrica denominada SMCluster (SMC).

Para a definição do k, foi empregada a razão do tamanho da SM pela quantidade máxima de elementos de um nó, limitado em 5 grupos. Esse algoritmo de agrupamento hierárquico é uma heurística que encontra um bom agrupamento, no entanto não é exaustivo, pois o problema tem complexidade computacional proibitiva. Outras abordagens

Algoritmo 6: Abordagem para agrupamento *Density*

Entrada: Conjunto de elementos S, tamanho partição c, taxa de ocupação t, métrica $dist()$ e número máximo de iterações I
Saída: Subconjunto de elementos s

```
1  mindist = ∞ best_s = {}
2  para k ≤ I faça
3  |  aleatoriamente selecione um elemento x_i ∈ S como representante rep
4  |  para x_i ∈ S faça
5  |  |  d[i] ← dist(rep, x_i)
6  |  fim
7  |  ordene d e S de acordo com a distância para o representante rep em ordem
   |   decrescente
8  |  s ← selecione um subconjunto x_i ∈ d do último para o primeiro (de acordo c e
   |   t)
9  |  sumdist ← soma das distâncias dos elementos de s em relação rep
10 |  se sumdist < mindist então
11 |  |  mindist ← dist e best_s ← s
12 |  fim
13 fim
14 remova os best_s ∈ S
15 retorne s
```

poderiam contemplar a utilizar de algoritmos mais eficientes para a escolha dos elementos representantes, como DBSCAN (ESTER et al., 1996) que é um algoritmo hierárquico mais rápido, porém, é restrito a espaços multidimensionais.

3.3 Considerações

Este capítulo apresentou as contribuições desta dissertação para a criação de novas estruturas métricas dinâmicas, nomeadas SMRandom, SMDensity e SMCluster, derivadas da Slim-Tree. As três variações desenvolvidas levam em consideração a estratégia de adiamento da indexação para permitir que novos elementos, que não são cobertos por um nó folha, aguardem na *short-term memory* e posteriormente sejam agrupados com elementos vizinhos mais próximos com a finalidade de formar um novo nó folha completo, que será inserido no índice principal e terá a capacidade de receber novas inserções devido à taxa de ocupação.

É importante salientar que um dos procedimentos considerados chaves no processo de construção de MAM's é o responsável pela divisão de nós, que ocorre quando o nó no qual um novo elemento deveria ser inserido já se encontra completo. Dessa maneira, para avaliação mais abrangente das estruturas propostas, adotou-se distintas políticas de divisão, sendo elas, MinMax, MST, DM (mais detalhes na Seção 2.5) em conjunto com abordagens apresentadas nessa seção para a construção das estruturas métricas que serão

Algoritmo 7: Abordagem para agrupamento *Cluster*

Entrada: Conjunto de elementos S, tamanho partição c, taxa de ocupação t, métrica $dist()$, número máximo de vizinhos analisados a cada passo $maxneighbor$, número de iterações de busca $numlocal$ e $kcluster$ quantidade de representantes

Saída: Subconjunto de elementos s

```
1  medoids ← CLARANS(S,maxneighbor,numlocal,kcluster)
2  para kcluster faça
3  |  rep ← medoids[i]
4  |  para x_i ∈ S faça
5  |  |  d[i] ← dist(rep, x_i)
6  |  fim
7  |  ordene d e S de acordo com a distância para o representante rep em ordem
   |   decrescente
8  |  s ← selecione um subconjunto x_1, x_2, ..., x_n ∈ d, tamanho c e limitado por t
9  |  remova os s ∈ S
10 |  newLeaves.add(s)
11 fim
12 retorne newLeaves
```

avaliadas na próxima Seção 4. Além disso, cada uma das variações desenvolvidas traz as suas particularidades, mas independentemente disso, a estratégia pode ser aplicada na construção de estruturas métricas ou multidimensionais, como a M-tree (CIACCIA; PATELLA; ZEZULA, 1997) e suas variantes e a R-tree (GUTTMAN, 1984).

Por fim, espera-se com as abordagens propostas reduzir o crescimento da área de cobertura do nó a ser inserido, assim, minimizando a sobreposição e, consequentemente, acelerar a execução de consultas por similaridade sobre dados complexos.

Algoritmo 8: Algoritmo *Clarans* (NG; HAN, 2002)

Entrada: Conjunto de elementos S, número máximo de vizinhos analisados a cada passo $maxneighbor$;número de iterações de busca $numLocal$ e k número de medoids.

Saída: Subconjunto de elementos **medoids**.

```
1  mincost = + ∞
2  para  i ∈ [0, numLocal] faça
3      aleatoriamente selecione um conjunto de medoids currentMedoidSet ∈ S de
        tamanho k
4      currentCost ← custo diferencial de currentMedoidSet
5      j ← 0
6      while j < maxneighbor do
7          randomMedoidSet ← vizinhos aleatorios de currentMedoidSet
8          randomCost ← custo diferencial randomMedoidSet
9          se randomCost < currentCost então
10             currentCost ← randomCost
11             currentMedoidSet ← randomMedoidSet
12             j ← 0;
13         fim
14         senão
15             j++
16         fim
17     end
18     se currentCost < bestCost então
19         bestCost ← currentCost
20         seleciona o conjunto bestMedoidSet
21     fim
22 fim
```

Capítulo 4

Experimentos e Análise dos Resultados

Este capítulo apresenta os resultados experimentais obtidos com o desenvolvimento desta pesquisa. Na Seção 4.1 são descritas as bases de dados utilizadas para os experimentos. Na Seção 4.2 são apresentados os resultados experimentais para validação dos métodos.

4.1 Bases de dados

Para os experimentos realizados neste capítulo foram utilizadas base de dados reais e artificiais. Na Tabela 2 são apresentadas as características das bases de dados empregadas para os experimentos realizados nesse trabalho. As especificações com relação as bases de dados artificiais SynD e SynEDC são apresentadas a seguir na Seção 4.1.1.

Tabela 2 – Descrição das bases de dados utilizadas nos experimentos

Conjunto	**Nº Instâncias**	**Nº Dimensões**	**Tipo**
Pendigitis	10.992	16	Real
Eigenfaces	11.900	16	Real
Letter	20.000	16	Real
Nasa	40.150	20	Real
Corel	68.040	32	Real
SynD	250.000	10	Sintética
SynEDC	400.000	40	Sintética
Covertype	581.102	54	Real
Colour Structure	1.000.000	64	Real

4.1.1 Base de artificiais

Estas bases foram utilizados nos trabalhos (FARIA et al., 2016b; FARIA; CARVALHO; GAMA, 2016; MASUD et al., 2011), para avaliar métodos de classificação e agrupamento de fluxos contínuos de dados simulando distribuição não estacionária de dados.

- **SynD - Synthetic data with only concept-drift:** A base encena mudança de conceito, gerada a partir de hiperplanos que se movem. Ela é composta por vetores de características com 250.000 instâncias e 10 dimensões (cada atributo está no intervalo [0,1]), e de modo aleatório é introduzido ruído na base.
- **SynEDC - Synthetic data with concept-drift and novel-class:** A base é composta de 400.000 instâncias, 40 atributos e 20 classes. Todos os exemplos foram gerados utilizando uma distribuição gaussiana com diferentes médias, variando de -5.0 a $+5.0$ e variâncias por classes 0.5 a 6. Algumas classes aparecem e desaparecem ao longo do tempo, sendo que a distribuição de probabilidade varia ao longo do tempo. A fim de inserir mudança de conceito, os valores médios de uma certa porcentagem dos atributos são alterados constantemente. Os valores de todos os atributos estão no intervalo [0,1].

4.1.2 Base de dados reais

A base de dados *Colour Structure* é um subconjunto com 1.000.000 de elementos do COPhIR *Structure Colour* que contém características extraídas de 100 milhões de imagens, disponível em (BOLETTIERI et al., 2009). As demais bases de dados reais que foram utilizadas neste trabalho são provenientes do repositório de dados UCI Machine Learning Repository (LICHMAN, 2013).

4.2 Experimentos

A técnica foi implementada na linguagem C++ e utiliza a biblioteca Arboretum[1]. Os experimentos foram executados em um microcomputador com sistema operacional Linux 64 bits, i7-4770 @3.40GHz, 8GB de RAM e disco rígido de 1TB. Para a realização dos experimentos os parâmetros gerais das árvores testadas foram:

- Função de distância: Euclidiana ou L_2;
- Tamanho da SM: definido de acordo com Tabela 3;
- Tamanho das páginas de disco: definido de acordo com Tabela 3;
- Critério *min-distance* para algoritmo *Choosesubtree*;
- Políticas divisão de nós: MinMax, MST e DM;
- Taxa de ocupação(elementos/página) de 75% dos nós criados a partir da SM;
- Escolha aleatória de 100 elementos do conjunto para realização das consultas.

[1] Disponível em http://gbdi.icmc.usp.br/arboretum

Para a escolha do tamanho das páginas de disco das árvores e a quantidade máxima de elementos suportados na *short-term memory*, foi utilizado como critério a quantidade de dimensões, bem como a cardinalidade dos conjuntos de dados. Na tabela 2 são apresentadas com detalhes os critérios adotados.

Tabela 3 – Descrição dos tamanhos das páginas de disco (bytes) e número de elementos da *short-term memory* (SM) utilizadas nos experimentos de acordo com o número de dimensões

Nº de Dimensões	**Tamanho da Página (bytes)**	**Nºde elementos SM**
8 - 10	256	100 - 500
11 - 20	1024	100 - 500
25 - 31	2048	500 -1000
32 - 63	4096	500 - 1000
64 - 127	8192	500 - 5000

O tamanho da *short-term memory* (SM) deve ser ponderado, isto porque, quanto maior o tamanho atribuído, maior será o número de cálculos de distância adicionais para processar os elementos, o que por ventura pode vir a comprometer o tempo total de construção do método. Desse modo, para testar o impacto decorrente do número de elementos que compõem a estrutura *short-term memory*, foram realizados experimentos onde variou-se o tamanho atribuído de acordo com o apresentado na Tabela 3. E, por meio dos resultados foi possível constatar que ao aumentar o tamanho da SM é possível uma melhora na distribuição dos elementos entre os nós, dependendo da base de dados, e consequentemente, melhora a qualidade das estruturas resultantes (mensurada por meio do fator relativo). No entanto, esse ganho de qualidade não é significativo, pois compromete muito o tempo de construção das árvores. A *short-term memory* foi implementada inicialmente para funcionar em memória principal. A expressão 7 apresenta como calcular a quantidade de memória necessária para funcionamento da estrutura:

$$D * (E + I) * Q. \tag{7}$$

onde D representa a dimensionalidade e Q a quantidade de elementos presentes na estrutura, E tamanho do elemento em bytes (4 *bytes*), I identificado do elemento *objetoID* em bytes (4 *bytes*).

4.2.1 Considerações sobre as políticas de divisão de nós

Ao optar por diferentes políticas de divisão de nós para a construção das estruturas de indexação ocorrem algumas implicações das quais impactam nas operações de consultas por similaridade. Por exemplo, o uso da política *MinMax* produz árvores com o menor grau de sobreposição entre os nós, contudo, o número necessário de cálculos de distâncias é elevado e, consequentemente, o tempo de execução também. Isso porque a complexidade

do algoritmo é $O(n^3)$, no qual n é o número de elementos do nó. Com o uso da política *MST* é possível construir árvores boas com um menor número de cálculos de distância, porém, isso não garante uma distribuição uniforme dos elementos entre nós (TRAINA-JR et al., 2002a). É importante salientar que o custo computacional dessa política é $O(n^2 log(n))$, em que n representa a quantidade de elementos presentes no nó. No trabalho (SOUZA; RAZENTE; BARIONI, 2014) foram propostas novas políticas de divisão de nós, dentre elas a *DM* (conforme exposto na Seção 2.5). Essa política permite gerar árvores com um custo computacional inferior quando comparado com *MinMax*, pois tem-se complexidade $O(n^2)$, onde n representa a quantidade de elementos presente em cada nó, enquanto mantém a qualidade do índice para execução de consultas por similaridade. Assim, este trabalho emprega as diferentes políticas de divisão de nós (MinMax, MST e DM), juntamente com as técnicas propostas nesta dissertação, objetivando construir novas estruturas métricas dinâmicas, derivadas da Slim-Tree.

As estruturas derivadas foram denominadas SMRandom, SMDensity e SMCluster. Elas foram construídas variando algoritmos de agrupamentos de dados aplicados a criação de nós folhas completos. O método de comparação foi o mesmo empregado nos trabalhos correlacionados, no qual, é feita a comparação com a própria estrutura Slim-Tree proposta em (TRAINA-JR et al., 2000). Além disso, os experimentos foram divididos nas três etapas descritas abaixo:

1. Na primeira etapa o objetivo é avaliar os parâmetros relacionados à construção dos MAMs, sobre a perspectiva das diferentes políticas de divisão de nós. Os parâmetros avaliados são: número de cálculos de distância, número de acessos a páginas de disco e tempo total em segundos.

2. Na segunda etapa a finalidade é avaliar a qualidade das estruturas resultantes do processo de construção, sobre a perspectiva das diferentes políticas de divisão de nós. Os parâmetros avaliados são as medidas fator relativo e fator absoluto e o número de nós resultantes.

3. Na terceira etapa objetiva-se avaliar o desempenho durante processo de consultas por similaridade, sobre a perspectiva das diferentes políticas de divisão de nós, sendo que os parâmetros considerados são: número médio de cálculos de distância, número médio de acessos a páginas de disco e tempo total em segundos para realização de consultas.

Para estas etapas, foi feita a comparação entre os métodos que foram construídos por meio do mesmo algoritmo de divisão de nós (comparação intragrupos) [2] e, posteriormente, uma análise mais amplas considerados os resultados para cada estrutura individualmente (comparação intergrupos) [3].

[2] Comparação intragrupos: MAMs construídos com as mesmas políticas de divisão de nós.
[3] Comparação intergrupos: MAMs construídos com diferentes políticas de divisão de nós.

4.2.2 Parâmetros de desempenho: análise do comportamento dos métodos durante o processo de construção

Nesta seção serão apresentados os parâmetros relacionados ao desempenho das árvores resultantes por meio da análise do número de cálculos de distância, número de acessos a páginas de disco e tempo total para construção. Além disso, será direcionada a responder ao seguinte questionamento: *"Q1: É possível gerar árvores com custo de construção inferior com o uso short-term memory ?"*.

Inicialmente foram realizados experimentos utilizando o conjunto de dados *Pendigits*, composto por 10.992 elementos e 16 dimensões. Além disso, os parâmetros de configuração utilizados nos experimentos foram: Tamanho *short-term memory* = 100 elementos e Tamanho página de disco = 1024 *bytes*, conforme critério adotado na Tabela 3.

Por meio dos resultados apresentados na Tabela 4, ao realizar comparação intragrupos, ou seja, MAMs construídas com a mesma política de divisão de nós, nota-se que todas as estruturas propostas SMRandom, SMDensity e SMCluster, combinadas a diferentes políticas de divisão de nós DM, MinMax e MST, apresentaram maior número de cálculos de distância quando comparadas com a estrutura Slim-Tree original, exceto para a estrutura SMRandom construída por meio da política MinMax que apresentou cerca de 43% menos cálculos de distância. Ainda considerando o número de cálculos de distância, verifica-se uma diferença notável para as estruturas SMCluster, pois estas apresentam um número muito elevado, pois possuem um alto custo computacional para criação de nós folhas. Desse modo, a técnica SMCluster não é recomendada para grandes bases de dados, principalmente, se combinada com a política MinMax.

Quando comparamos o número de acessos a páginas de disco, em todos os resultados as estruturas SMRandom, SMDensity e SMCluster apresentam quantidades maiores. Contudo, o aumento é discreto em torno de 6%, 4% e 2% para estruturas construídas por meio das políticas DM, MinMax e MST, respectivamente. Com esses dois fatores, maior número de cálculos de distância e maior número de acessos a disco, o tempo total médio de construção das estruturas propostas foram maiores em comparação com Slim-tree, como apresentado na Tabela 4. Contudo, entre as estruturas Slim-Tree e SMRandom construídas pela política MinMax observou-se uma diferença muito pequena (6% aproximadamente), para o tempo total de construção. Apesar de estas estruturas propostas apresentarem maiores tempos para a construção dos índices, ao considerar os ganhos com relação à qualidade das árvores resultantes e desempenho em consultas por similaridade (Seções 4.2.4 e 4.2.3, respectivamente), o contexto *"custo vs. benefício"* compensa o aumento nos tempos. A mesma ideia não pode ser aplicada ao SMCluster, devido o alto custo de construção e o pouco de ganho resultante das consultas.

Tabela 4 – Resultados dos experimentos para a construção com conjunto de dados *Pendigists*. Parâmetros comparação: Cálculos de distância, Acessos a páginas de disco e Tempo total em segundos. Destaca-se na cor cinza o resultado no qual o método proposto atinge menor número de cálculos de distância.

Parâmetros	**Política DM**				**Política MinMax**				**Política MST**			
	Slim-Tree	**SMR**	**SMD**	**SMC**	**Slim-Tree**	**SMR**	**SMD**	**SMC**	**Slim-Tree**	**SMR**	**SMD**	**SMC**
Acesso	105.964	112.544	112.191	111.675	108.005	112.732	112.693	112.785	110.930	113.531	112.924	112.830
Distância	320.952	568.616	2.923.912	93.325.614	2.400.564	1.359.905	5.275.412	146.254.696	350.100	624.378	4.507.510	140.135.618
Tempo (s)	0,19	0,66	4,67	23,28	1,04	1,11	7,62	36,22	0,21	0,82	7,22	34,75

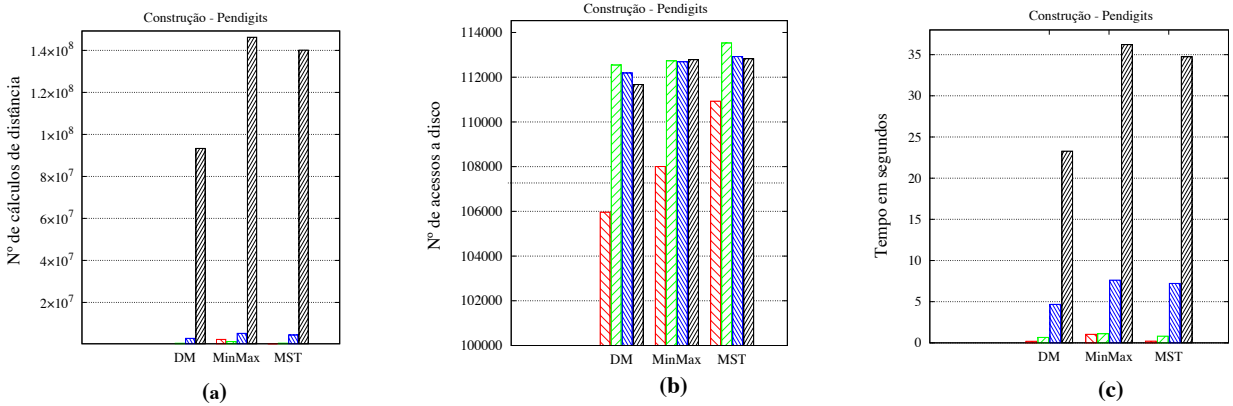

Figura 18 – Construção com a base *Pendigits*. Em (a) Número de cálculos de distância, (b) Número de acessos a disco e (c) Tempo em segundos.

O método SMCluster possui um alto custo associado ao número de cálculos distância executados, Figura 19 a. Assim, para os demais experimentos não será utilizado.

Tabela 5 – Resultados dos experimentos para a construção com os conjuntos de dados avaliados. Parâmetros comparação: Cálculos de distância, Acessos a páginas de disco e Tempo total em segundos. Destacados na cor cinza estão os resultados nos quais os métodos propostos atingem menor número de cálculos de distância.

Base	**Parâmetros**	**Política DM**			**Política MinMax**			**Política MST**		
		Slim-Tree	**SMR**	**SMD**	**Slim-Tree**	**SMR**	**SMD**	**Slim-Tree**	**SMR**	**SMD**
Eigenfaces	Acessos a disco	80.015	78.842	78.968	80.355	78.001	77.783	80.981	77.502	77.749
	Cálculo de distância	701.486	702.908	843.965	33.525.949	**28.433.045**	**27.458.742**	780.692	696.567	1.080.989
	Tempo total (s)	0,23	0,23	0,4	9,45	**8,1**	**8,08**	0,24	0,26	0,77
Letter	Acessos a disco	136.842	136.592	136.966	137.420	134.723	134.605	137.683	136.237	135.723
	Cálculo de distância	1.202.718	1.216.820	1.229.243	50.956.378	**47.936.889**	**48.847.727**	1.286.962	1.282.344	1.538.463
	Tempo total (s)	0,35	0,36	0,37	14,36	**13,68**	**14,33**	0,4	0,42	0,75
Nasa	Acessos a disco	282.192	290.380	286.839	288.848	299.311	295.096	303.025	304.732	309.191
	Cálculo de distância	2.548.475	2.579.338	2.950.408	74.101.695	**60.442.379**	**58.838.641**	2.731.786	2.379.956	4.285.652
	Tempo total (s)	0,9	0,95	1,37	23,93	**19,66**	**21,28**	1,02	1,12	3,5
Corel	Acessos a disco	586.650	584.566	583.386	589.577	576.636	573.403	597.126	573.645	576.365
	Cálculo de distância	3.600.541	3.661.502	5.542.587	59.531.206	**41.750.704**	**46.016.152**	3.916.017	3.671.411	12.312.253
	Tempo total (s)	1,79	1,93	4,11	26,01	**19,27**	27,01	2,01	2,75	12,46
SynD	Acessos a disco	2.715.268	2.952.228	2.926.265	2.725.742	3.091.953	3.116.102	3.020.465	3.427.213	3.169.964
	Cálculo de distância	10.965.550	11.503.276	16.447.883	89.757.446	**50.833.056**	117.067.245	11.957.272	13.876.728	72.607.701
	Tempo total (s)	2,11	2,50	5,06	7,82	**7,80**	39,21	2,48	5,28	33,01
Covertype	Acessos a disco	5.585.020	5.857.413	5.850.491	5.692.087	5.828.782	5.807.488	5.872.511	5.856.555	5.840.448
	Cálculo de distância	41.153.774	40.620.378	98.606.861	843.716.826	**370.713.534**	**432.488.631**	44.801.134	35.891.535	106.244.661
	Tempo total (s)	5,52	7,50545	32,4091	60,98	30,89	59,62	6,12	8,16	38,42
ColourStructure	Acessos a disco	9.905.524	10.071.951	10.292.015	9.976.561	10.427.336	10.429.538	10432362	10471241	10499623
	Cálculo de distância	75.599.828	75.514.490	80.363.793	1.001.897.477	**842.740.440**	**891.412.145**	75.808.904	68.024.169	153.386.414
	Tempo total (s)	85	87	100	1.037	**890**	**1.017**	87	93	282
SynEDC	Acessos a disco	4.095.448	4.203.754	4.201.159	4.122.745	4.175.541	4.174.561	4.489.356	4.300.489	4.188.072
	Cálculo de distância	25.720.654	27.592.561	77.576.859	271.442.135	**93.562.192**	**178.098.270**	40.768.783	26.208.676	121.817.759
	Tempo total (s)	15,88	23,91	102,00	141,89	**59,01**	185,65	26,01	**24,48**	158,68

O próximo resultado que será apresentado corresponde ao conjunto de dados sintéticos *SynD*, composto por 250.000 elementos com 10 atributos numéricos. Os parâmetros de configuração utilizados foram: tamanho SM = 500 elementos e tamanho página de disco = 1024 *bytes*, conforme critério adotado na Tabela 3.

Ao analisar o comportamento intragrupo das estruturas, dados da Tabela 5, em comparação com o MAM Slim-Tree para o processo de construção, nota-se que:

- Política DM: os métodos propostos SMRandom e SMDensity apresentaram aproximadamente 19% e o dobro dos tempos totais para a construção, respectivamente. O desempenho inferior da SMDensity ocorreu pois esta exige aproximadamente o dobro do número de cálculos de distância para construção. Além disso, os métodos propostos requerem quantidades maiores de acessos a páginas de disco.

- Política MinMax: a estrutura SMRandom obteve quantidade muito menor de cálculos de distância, aproximadamente 43%, enquanto para SMDensity ocorre um aumento de cerca de 30%. Em relação ao número de acessos a página de disco estas estruturas resultaram em maiores quantidades. Desse modo, as estruturas propostas não foram construídas em tempo menor. Contudo, a SMRandom obteve valores próximos a Slim-Tree sendo a diferença inferior a 1% do tempo de execução.

- Política MST: a estrutura SMRandom obteve 16% mais número de cálculos de distância comparado com Slim-Tree. Enquanto a SMDensity apresenta um resultado ruim, de cerca de cinco vezes mais número de cálculos de distância. Além disso, ambas as estruturas apresentaram maiores quantidade de acessos a disco, colaborando para obterem maiores tempos totais para a construção comparadas a Slim-Tree.

Com relação aos resultados apresentados na Tabela 5 para o conjunto de dados *SynD* é importante observar que mesmo com os cálculos de distância para processamento da *short-term memory*, o total de cálculos de distância para a construção dos índices na perspectiva do uso da política MinMax para a estrutura SMRandom apresentou resultados inferiores, colaborando para melhorar a eficiência na construção do método. O resultado é decorrente do fato de melhorar a distribuição dos elementos entre os nós folhas e reduzir o número necessário de operações de divisão de nós. Por fim, na comparação intergrupos as estruturas propostas geradas por meio da política DM são as que constroem árvores com menor tempo de execução.

A seguir serão apresentados os resultados dos experimentos para o conjunto *Colour Structure*, composto por 1.000.000 elementos com 64 atributos numéricos. Os parâmetros utilizados nos experimentos foram: tamanho SM = 500 elementos e tamanho página de disco = 8192 *bytes*, conforme critério adotado na Tabela 3.

[1] Comparação intragrupos: consiste em avaliar os MAMs construídas com a mesma política de divisão de nós.

Para o processo de construção, quando analisado o comportamento das estruturas intragrupo, ou seja, analisar as diferentes estruturas construídas com as mesmas política de divisão de nós, é possível notar com base na Tabela 5 que:

- Política DM: os métodos propostos SMRandom e SMDensity apresentam maiores tempos totais, aproximadamente 2% e 17%, respectivamente, comparados com a Slim-Tree. Tal resultado ocorreu devido os métodos demandarem cerca de 1% menos e 6% mais em cálculos de distância, e também maiores quantidades de acessos a disco aproximadamente 2% e 4%, respectivamente.

- Política MinMax: os métodos propostos SMRandom e SMDensity resultam em redução significativa do cálculo de distância, aproximadamente 16% e 11%, respectivamente, comparados com Slim-Tree. No entanto, o número de acesso a páginas de disco foi maior cerca de 5% para ambas as estruturas propostas. Desse modo, o tempo total para a construção das árvores de indexação foi mais rápido que a Slim-Tree em aproximadamente 14% e 2% para SMRandon e SMDensity, respectivamente.

- Política MST: o tempo total construção dos métodos propostos são maiores cerca de 8% para a SMRandom e, de até duas vezes mais para SMDensity. Esse último ocorre devido ao alto custo relacionado ao número de cálculos de distância necessário para a criação dos nós folhas derivados da *short-term memory*.

Ainda com relação aos resultados da Tabela 5, para conjunto de dados *Colour Structure* faz-se necessário destacar que o número de acessos a disco foi similar entre todas as políticas apresentadas. Além disso, é possível identificar que as estruturas que apresentam menor tempo total para a construção são Slim-tree e SMRandom, construídas por meio da política DM. Por fim, é importante mencionar os bons resultados com relação a redução do número de cálculos de distância das estruturas propostas ao serem construídas com política MinMax. Tais resultados permitiram em alguns casos prover estruturas com tempo de construção inferior a Slim-Tree, quando analisada sobre a mesma política de divisão de nós.

4.2.3 Parâmetros de qualidade: Comparação do comportamento dos métodos durante o processo de construção

Nesta seção serão apresentados os parâmetros relacionados à qualidade das árvores resultantes por meio da análise das medidas fator absoluto e fator relativo e número de nós. O objetivo principal é a avaliação do grau de sobreposição decorrente do processo de indexação entre os nós das árvores resultantes a partir de diferentes políticas de divisão

de nós. Os testes foram direcionados com a finalidade de responder ao seguinte questionamento: *"Q2: É possível gerar árvores com grau de sobreposição inferior ao fazer uso short-term memory no processo de construção da estrutura de indexação?"*.

Com relação à qualidade das árvores resultantes na Tabela 6 e 7 é possível notar que as técnicas propostas nessa dissertação provêm bons ganhos de qualidade para todas as estruturas proposta e avaliadas. Enquanto a Slim-Tree apresenta melhor fator relativo de 0,22 para conjunto *Covertype* utilizando a política MinMax, as estruturas propostas SMRandom e SMDensity apresentam para mesmo conjunto os melhores resultados que foram de 0,15 para ambas as estruturas.

Considerando os resultados para o conjunto *Pendigits*, Tabela 6, com relação a medida fator relativo em comparação com a estrutura Slim-tree padrão, destaca-se que:

- Política DM: as estruturas propostas possuem um ganho de qualidade de aproximadamente 35%, 27% e 29% (SMRandom, SMDensity e SMCluster, respectivamente) em comparação a Slim-Tree.
- Política MinMax: o ganho é de aproximadamente 38%, 37% e 36% para SMRandom, SMDensity e SMCluster, respectivamente, em comparação a Slim-Tree.
- Política MST: o ganho é de aproximadamente 49%, 50% e 48% para as estruturas SMRandom, SMDensity e SMCluster, respectivamente, quando comparadas com a Slim-Tree.

Analisando os resultados em relação ao fator absoluto, Tabela 7, os resultados sugerem que as estruturas propostas construídas por meio das políticas DM,MinMax e MST são consideradas aceitáveis [4], isso porque estão entre 0,1 a 0,3 (fator absoluto). Enquanto a Slim-tree apresenta árvore considerada aceitável apenas para a estrutura construída por meio da política MST. Desse modo, os resultados apresentados possibilitaram inferir que houve uma melhora significativa na qualidade das árvores de indexação por meio da estratégia de adiamento da inserção.

[4] Uma árvore boa deve apresentar fator absoluto entre 0 a 0,1, entre 0,1 a 0,3 tem-se uma árvore aceitável e maior que 0,3 tem-se uma árvore ruim (TRAINA-JR et al., 2000).

Tabela 6 – Parâmetros de qualidade das estruturas resultantes para os conjunto de dados *Pendigits*.

Base	Parâmetros	Política DM				Política MinMax				Política MST			
		Slim-Tree	SMR	SMD	SMC	Slim-Tree	SMR	SMD	SMC	Slim-Tree	SMR	SMD	SMC
Pendigits	Fator Relativo	0,54	0,35	0,39	0,38	0,49	0,30	0,31	0,31	0,55	0,28	0,27	0,28
	Fator Absoluto	0,38	0,21	0,23	0,22	0,31	0,18	0,18	0,19	0,28	0,17	0,16	0,16
	Quantidades de nós	1.213	1.406	1.422	1.450	1.346	1.397	1.432	1.414	1.636	1.400	1.457	1.467

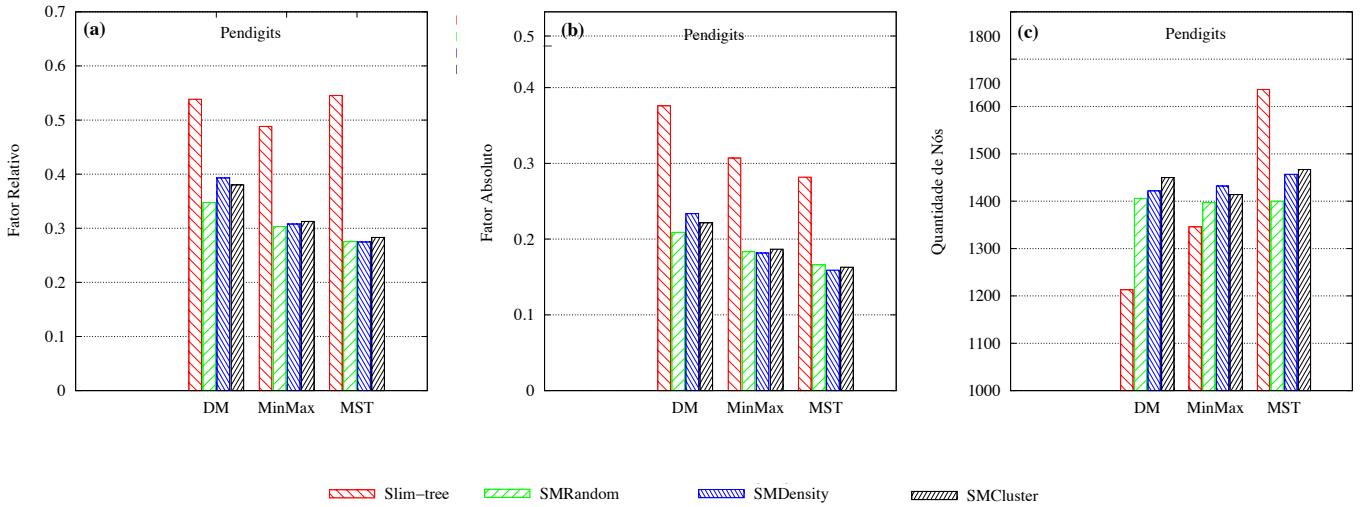

Figura 19 – Qualidade das estruturas resultantes para o conjunto de dados *Pendigits*. Em (a) Fator Relativo, (b) Fator Absoluto e (c) Número nós.

Tabela 7 – Parâmetros de qualidade das estruturas resultantes para os conjuntos de dados avaliados. Destacados na cor cinza estão os resultados nos quais os métodos propostos atingem menor quantidade de nós.

Base	**Parâmetros**	**Política DM**			**Política MinMax**			**Política MST**		
		Slim-Tree	**SMR**	**SMD**	**Slim-Tree**	**SMR**	**SMD**	**Slim-Tree**	**SMR**	**SMD**
Eigenfaces	Fator Relativo	0,35	0,29	0,29	0,27	0,15	0,17	0,27	0,17	0,18
	Fator Absoluto	0,25	0,20	0,20	0,18	0,10	0,11	0,16	0,10	0,10
	Quantidades de nós	284	305	290	307	319	312	352	355	355
Letter	Fator Relativo	0,90	0,87	0,89	0,60	0,38	0,36	0,56	0,49	0,50
	Fator Absoluto	0,69	0,66	0,67	0,44	0,26	0,24	0,34	0,31	0,29
	Quantidades de nós	445	448	451	464	500	510	558	546	571
Nasa	Fator Relativo	0,65	0,57	0,55	0,37	0,25	0,21	0,37	0,26	0,26
	Fator Absoluto	0,51	0,42	0,41	0,27	0,16	0,14	0,22	0,16	0,15
	Quantidades de nós	1091	1163	1164	1154	1312	1333	1400	1430	1450
Corel	Fator Relativo	0,74	0,61	0,53	0,54	0,24	0,21	0,50	0,23	0,25
	Fator Absoluto	0,63	0,50	0,41	0,43	0,17	0,14	0,29	0,14	0,15
	Quantidades de nós	2.747	2.897	2.985	2.951	3.439	3.472	4.018	**3.752**	**3.734**
SynD	Fator Relativo	0,84	0,77	0,76	0,63	0,37	0,37	0,90	0,46	0,45
	Fator Absoluto	0,52	0,45	0,45	0,34	0,18	0,17	0,33	0,21	0,20
	Quantidades de nós	18.440	19.269	19.277	20.842	23.298	24.309	31.434	**25.295**	**25.217**
Covertype	Fator Relativo	0,36	0,25	0,19	0,22	0,15	0,15	0,26	0,15	0,15
	Fator Absoluto	0,23	0,13	0,10	0,12	0,08	0,08	0,12	0,08	0,08
	Quantidades de nós	20.535	24.711	25.495	23.280	23.780	24.185	28.769	**24.647**	**24.370**
ColourStructure	Fator Relativo	0,66	0,59	0,59	0,43	0,22	0,22	0,33	0,20	0,23
	Fator Absoluto	0,66	0,59	0,59	0,43	0,22	0,22	0,33	0,20	0,23
	Quantidades de nós	37.141	37.933	37.831	38.969	44.111	44.499	54.551	**53.644**	**53.871**
SyEDC	Fator Relativo	0,44	0,40	0,39	0,30	0,29	0,30	1,68	0,31	0,29
	Fator Absoluto	0,28	0,25	0,43	0,21	0,21	0,22	0,31	0,20	0,20
	Quantidades de nós	27.547	28.130	28.249	24.161	24.165	24.179	93.140	**26.639**	**25.516**

A seguir serão apresentados os resultados para as bases *SynD* e *Colour Structure*. É importante destacar que não foram realizados testes com a estrutura SMCluster nestes experimentos, visto seu alto custo computacional que demandaria para bases de dados robustas.

Com relação a qualidade das árvores resultantes para a base *SynD*, Tabela 7, comparado com a Slim-Tree o ganho foi até 8% e 9% para árvores construídas com a política DM, de 41% para ambas as árvores construídas pela política MinMax e 49% para as ambas as árvores construídas com a política MST para estruturas SMRandom e SMDensity, respectivamente. Os resultados com relação ao fator absoluto, Tabela 7 (base *SynD*), sugerem que as árvores construídas por meio da política DM são considerada ruins, ou seja, possuem valores maiores que 0,3.

A quantidade de nós, Tabela 7 - SynD, é maior para as estruturas propostas SMRandom e SMDensity nas políticas DM (4% e 5%) e MinMax (12% e 17%), respectivamente. No entanto, para a política MST é possível obter uma redução de aproximadamente 20% número de nós. Esse resultado é decorrente do fato da técnica proporcionar nós folhas completos com distribuição uniforme o que colabora para otimização das estruturas resultantes. Visto que na literatura correlata o maior problema da técnica MST relacionava-se ao fato da política gerar nós folhas com distribuição não uniforme.

Para o conjunto *Colour Structure*, Tabela 7, é possível notar bons resultados com relação ao fator relativo sendo que o ganho de qualidade quando comparado com a Slim-Tree utilizando a política DM é de aproximadamente 11% e 10% (SMRandom e SMDensity, respectivamente). A política MinMax representa os melhores resultados de otimização, aproximadamente 50% para ambas as estruturas SMRandom e SMDensity. Já na política MST, apresenta o resultado de 41% e 31% (SMRandom e SMDensity, respectivamente) em comparação à estrutura Slim-Tree.

Analisando os resultados com relação a fator absoluto, (Tabela 7 - *Colour Structure*), verifica-se que árvores construídas por meio da política DM são consideradas ruins, ou seja, com alta taxa de sobreposição, o que implica em um mal desempenho durante as consultas por similaridade. No entanto, as árvores propostas ao serem construídas por meio das políticas MinMax e MST são consideradas aceitáveis, isso por que estão entre 0,1 a 0,3. Vale destacar ainda que as estruturas SMRandom e SMDensity possuem um fator absoluto muito inferior a Slim-Tree, cerca de 50%, assim, é possível inferir que são estruturas melhores. Nota-se que todas as estruturas possuem a mesma altura da hierarquia gerada igual à 5.

Além disso, notou-se que política MST, Tabela 7 - *Colour Structure*, gera árvores com maiores quantidade de nós, assim como já discutidos na literatura correlata. Porém quando combinada com a técnica proposta nesta dissertação é possível constatar uma redução significativa do número de nós, colaborando para obter árvores com melhor qualidade, inclusive melhores do que a Slim-tree, quando combinada a política MinMax.

Enquanto para as demais políticas o oposto ocorre, ou seja, todas têm um aumento da quantidade de páginas de disco (quantidade de nós), como apresentados na Tabela 6 e 7.

Por fim, vale ressaltar que os resultados encontrados nessa seção são importantes visto que a qualidade das árvores impacta diretamente no desempenho das consultas por similaridade. Desse modo, árvores com maior qualidade permitem consultas por similaridade mais eficientes, como será apresentado na seção 4.2.4 a seguir.

4.2.4 Parâmetros de desempenho: Análise do comportamento dos métodos durante o processo de consultas por similaridade

Esta seção apresenta a avaliação do impacto decorrente das estratégias adotadas para a construção dos métodos de acesso métricos dinâmicos, no que se refere ao desempenho no processo de consultas por similaridade, sobre a perspectiva das diferentes políticas de divisão de nós. Os parâmetros considerados para avaliação são: número médio de cálculos de distância, número médio de acessos a páginas de disco e tempo total para consulta (em segundos). Além disso, os testes têm como intuito responder ao seguinte questionamento: *"Q3: É possível obter consultas por similaridade mais eficientes ao utilizar-se da estrutura short-term memory no processo de construção dos métodos?"*.

É necessário esclarecer que para a realização dos experimentos a *short-term memory* inicia vazia e, toda vez que é necessário realizar consultas são descarregados os elementos para a estrutura de indexação. Dessa maneira, não é necessário criar um mecanismo diferente para busca. Os conjuntos de dados são os mesmos apresentados nos experimentos anteriores, ou seja, *Pendigits*, *SynD* e conjunto *Colour Structure*. O algoritmo utilizado para a consulta é o k-vizinhos mais próximos (k-nn), no qual, os valores de k variam de 10 a 100 elementos e, considerou-se a média para 100 consultas.

Com relação aos resultados dos experimentos para o conjunto de dados *Pendigits*, Figura 20, é importante destacar que:

- Política DM: quando aplicada a SMRandom, SMDensity e SMCluster (comparação intragrupos), foi possível reduzir em até 44%, 39% e 40% o número de cálculos de distância (Figura 20 g), 37%, 31% e 32% o número de acessos a páginas de disco (Figura 20 d) colaborando para uma redução no tempo total de execução 41%, 36% e 37% (Figura 20 a), respectivamente.

- Política MinMax: quando aplicada a SMRandom, SMDensity e SMCluster (comparação intragrupos) foi possível reduzir em até 51%, 52% e 48% o número de cálculos de distância (Figura 20 h), 44%, 44% e 42% o número de acessos a páginas de disco (Figura 20 e) colaborando para uma redução no tempo total de execução 48%, 48%e 45% (Figura 20 b), respectivamente.

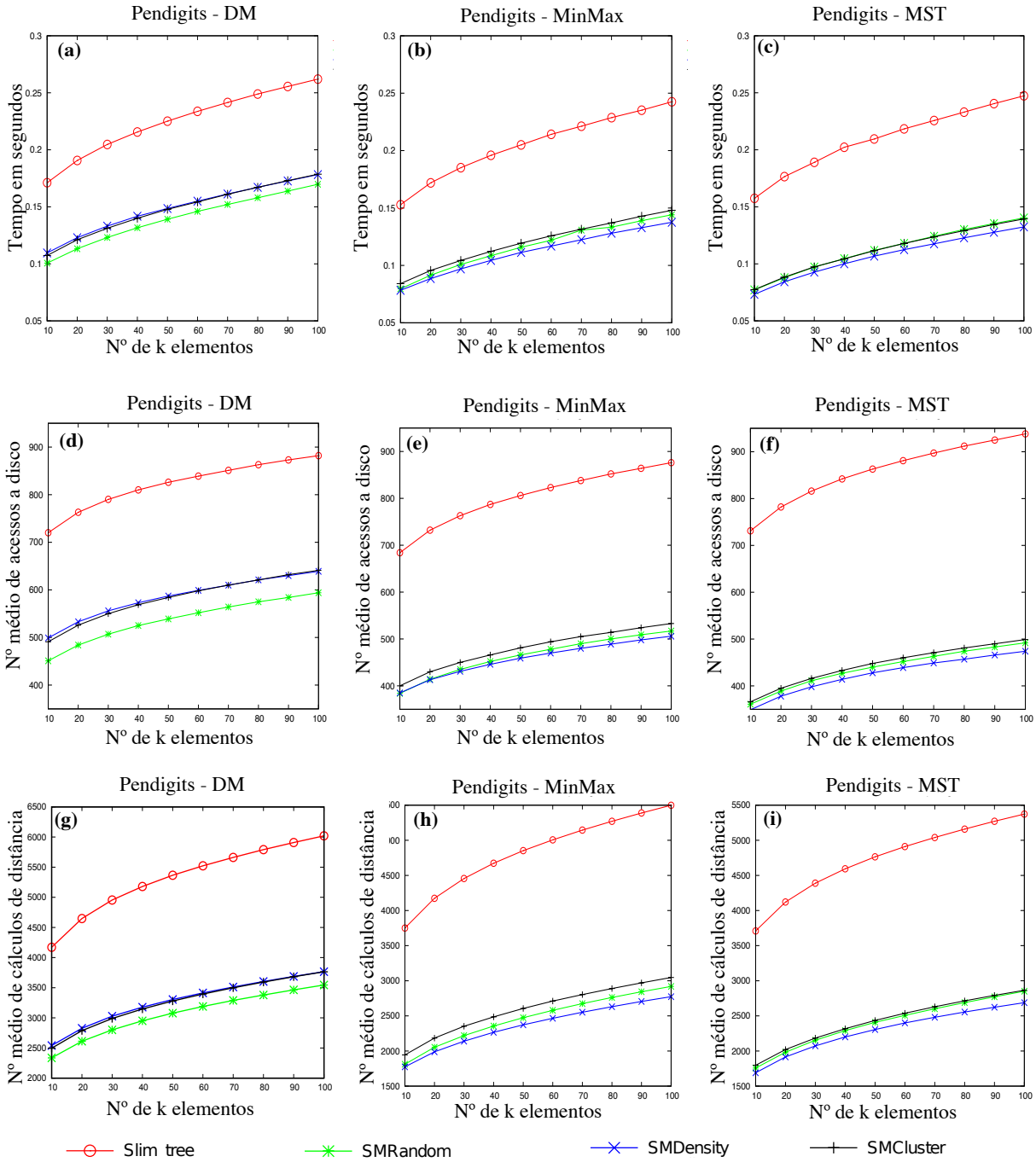

Figura 20 – Comparação intragrupos para as consultas ao k-vizinhos mais próximos, conjunto *Pendigits*. Em (a),(b) e (c) Tempo médio em segundos; (d),(e) e (f) Número médio de acessos a disco e (g),(h) e (i) Número médio de cálculos de distância.

- Política MST: quando aplicada a SMRandom, SMDensity e SMCluster foi possível reduzir em até 52%, 54% e 51% o número de cálculos de distância (Figura 20 i), 50%, 52% e 49% o número de acessos a páginas de disco (Figura 20 f) colaborando para uma redução no tempo total de execução 51%,53% e 50% (Figura 20 c), respectivamente.

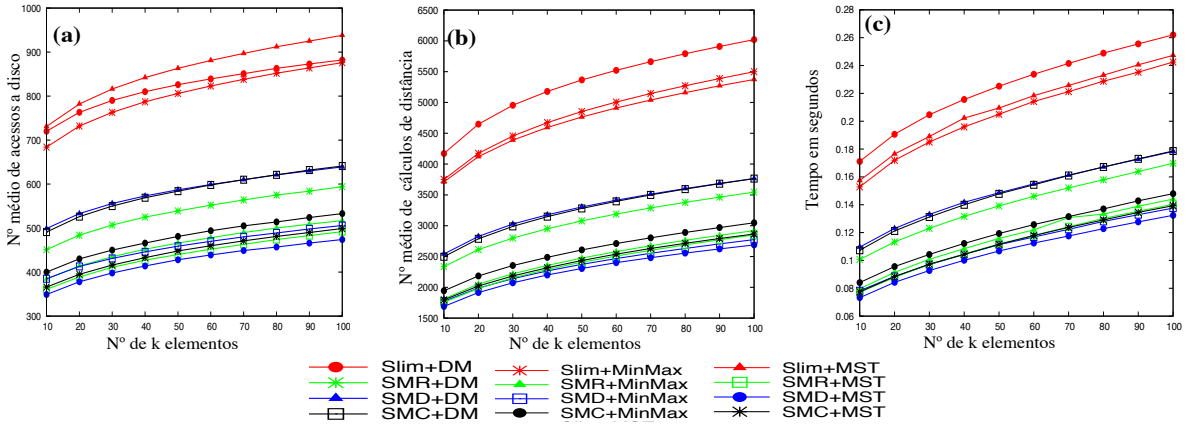

Figura 21 – Comparação intergrupos para as consultas ao k-vizinhos mais próximos, conjunto *Pendigits*. Em (a) Número médio de acessos a disco, (b) Número médio de cálculos de distância e (c) Tempo total em segundos. Em cor vermelha estão representadas estruturas Slim-tree padrão; em verde as estruturas SM-Random; em azul as estruturas SMDensity e; em preto as estruturas SM-Cluster.

Apesar das estruturas SMRandom e SMDensity construídas por meio da política DM não superar as demais estruturas propostas em tempo de execução, elas atingem bons resultados quando comparadas as estruturas Slim-Tree sobre a perspectivas de diferentes políticas de divisão de nós (Figura 21 c). Com relação, a política MinMax e MST quando aplicadas as estruturas propostas, geram árvores com praticamente o mesmo desempenho nas operações de consulta por similaridade. Além disso, apesar dos bons resultados das estruturas SMCluster construídas sobre a perspectiva de diferentes políticas, nota-se que seu ganho de desempenho com relação as demais estruturas propostas não compensa o seu alto custo de construção.

Com relação aos resultados dos experimentos para conjunto de dados *SynD*, Figura 22, quando comparados com as estruturas Slim-Tree, ao serem constituídas por meio de diferentes políticas, nota-se que:

- Política DM: quando aplicada as estruturas SMRandom e SMDensity, foi possível reduzir em até 4% o número médio de cálculos de distância (Figura 22 g). No entanto, o número médio de acessos a páginas de disco foi maior, cerca de 1%, (Figura 22 d), colaborando para uma pequena redução no tempo total de execução 3% e 2% (Figura 22 a), respectivamente.

- Política MinMax: quando aplicada as estruturas SMRandom e SMDensity foi possível reduzir em até 47% e 44% o número médio de cálculos de distância (Figura 22 h), 32% e 31% menos número médio de acessos a páginas de disco (Figura 22 e) colaborando para uma redução no tempo total de execução 40% e 38% (Figura 22 b), respectivamente.

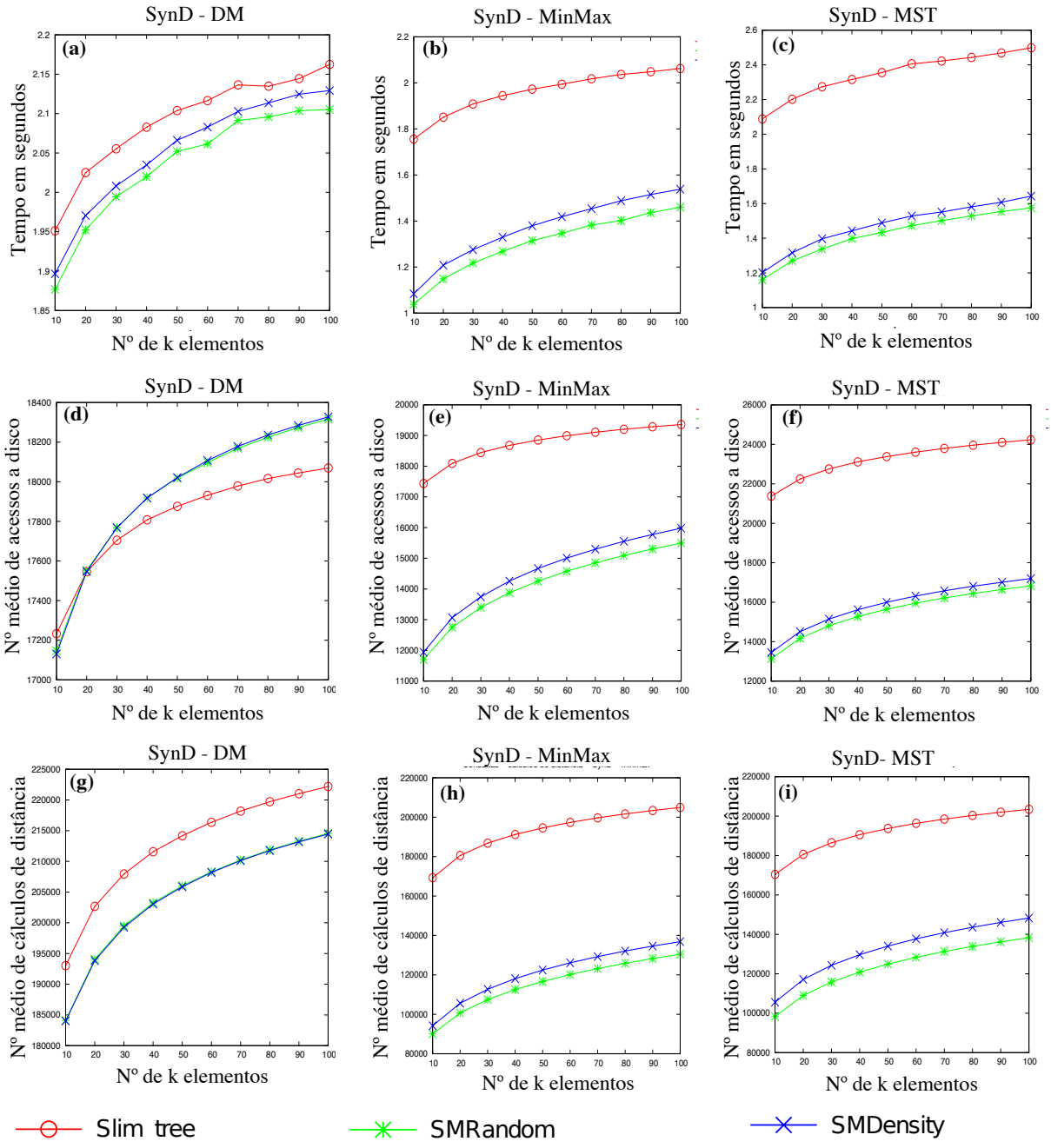

Figura 22 – Comparação intragrupos para as consultas ao k-vizinhos mais próximos, conjunto *SynD*. Em (a),(b) e (c) Tempo total em segundos;(d),(e) e (f) Número médio de acessos a disco e (g),(h) e (i) Número médio de cálculos de distância.

- Política MST: apresentou o maior desempenho ao analisar intragrupos, pois para estruturas SMRandom foi possível reduzir em até 42% o número médio de cálculos de distância, porém, a SMDensity apresenta maiores quantidades cerca 7% (Figura 22 i). Para número médio de acessos a páginas de disco as estruturas reduzem em até 38% e 37% (Figura 22 f) colaborando para uma redução no tempo total de execução 44% e 42% (Figura 22 c), respectivamente.

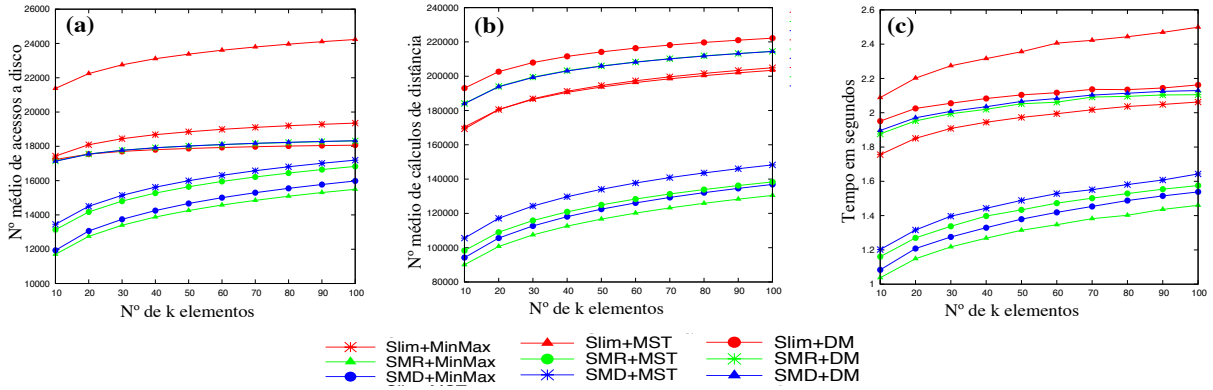

Figura 23 – Comparação intergrupos para as consultas ao k-vizinhos mais próximos, conjunto *SynD*. Em (a) Número médio de acessos a disco, (b) Número médio de cálculos de distância e (c) Tempo total em segundos. Em cor vermelha estão representadas estruturas Slim-tree padrão; em verde as estruturas SMRandom e; em azul as estruturas SMDensity.

Em comparação intergrupos, ou seja, MAMs construídos com diferentes políticas alcançou resultados semelhantes ao apresentado para base *Pendigits*. Assim, notou-se que as estruturas SMRandom e SMDensity construídas por meio da política DM apesar alcançarem bom desempenho, ainda não conseguem superar as políticas de divisão clássicas em tempo de execução. Observou-se que política MinMax aplicadas da construção da Slim-tree, apresenta o melhor tempo de execução das consultas quando comparada a mesma estrutura utilizando a política MST. No entanto, para as estruturas propostas utilizando-se das mesmas políticas, notou-se praticamente o mesmo desempenho, como pode ser observado na Figura 23 c, sendo que a política MinMax possui desempenho superior ao MST em apenas 10%, aproximadamente. Desse modo, para esse conjunto de dados a política MinMax, continua a ser a melhor estratégia para a construção de árvores com melhores desempenhos para consultas, no entanto, demanda maior custo para construção quando comparadas as demais políticas.

Com relação aos resultados dos experimentos para o conjunto de dados *Colour Structure* ao realizar a comparação intragrupos com base nos resultados apresentados na Figura 24, nota-se que:

- Política DM: para ambas as estruturas SMRandom e SMDensity, foi possível reduzir apenas 2% o número médio de cálculos de distância (Figura 24 g). O número médio de acessos a páginas de disco (Figura 24 d) não foi significativo para ambas as estruturas menos de 1%. Desse modo, o tempo total de execução das consultas não apresentou resultados significativos quando comparados com a estrutura Slim-Tree padrão. Isso porque o ganho foi de aproximadamente 3% (Figura 24 a).

- Política MinMax: Para as estruturas SMRandom e SMDensity, foi possível reduzir em até 16% e 15% o número médio de cálculos de distância (Figura 24 h), respectivamente. No entanto, o número médio de acessos a páginas de disco (Figura 24 e)

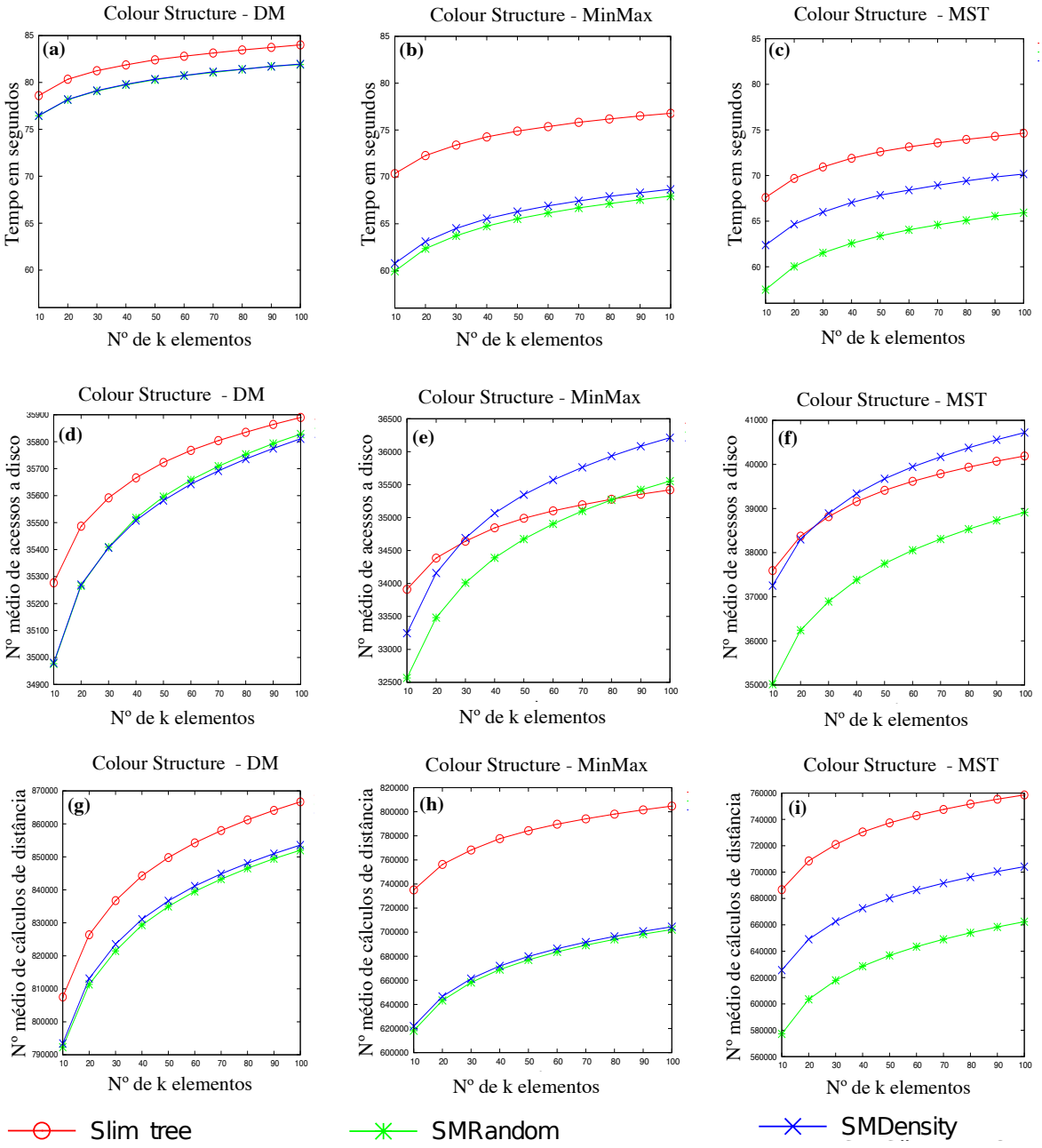

Figura 24 – Comparação intragrupos para as consultas ao k-vizinhos mais próximos, conjunto *Colour Structure*. Em (a),(b) e (c) Tempo total em segundos;(d),(e) e (f) Número médio de acessos a disco e (g),(h) e (i) Número médio de cálculos de distância.

para a SMRandom foi reduzido apenas em 7%. Além disso, para SMDensity ganho não foi significativo, apenas 1%. Contudo, o tempo total de execução das consultas manteve-se melhor para as estruturas propostas, ou seja, 15% e 14% (Figura 24 b), SMRandom e SMDensity, respectivamente, quando comparada com Slim-Tree.

- Política MST: para as estruturas SMRandom e SMDensity, foi possível reduzir em até 15% e 9% o número médio de cálculos de distância (Figura 24 i), respectiva-

mente. No entanto, o número médio de acessos a páginas de disco (Figura 24 f) para a SMRandom foi reduzido apenas 4%. Além disso, para SMDensity ganho não foi significativo, apenas 2%. Contudo, o tempo total de execução das consultas manteve-se melhor para as estruturas propostas, ou seja, 15% e 8% (Figura 24 c), SMRandom e SMDensity, respectivamente, quando comparada com Slim-Tree.

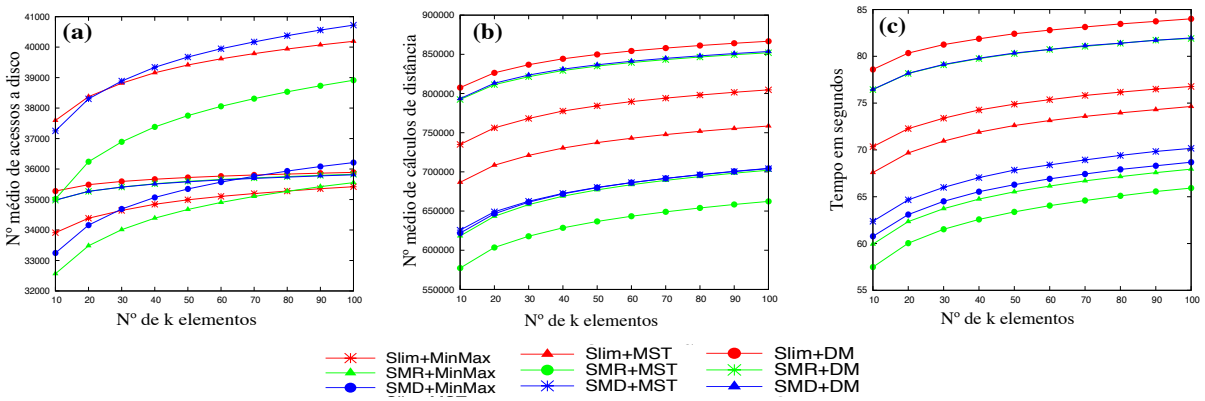

Figura 25 – Comparação intergrupos para as consultas ao k-vizinhos mais próximos, conjunto *Colour Structure*. Em (a) Número médio de acessos a disco, (b) Número médio de cálculos de distância e (c) Tempo total em segundos. Em cor vermelha estão representadas estruturas Slim-tree padrão; em verde as estruturas SMRandom e; em azul as estruturas SMDensity.

Na comparação intergrupos notou-se que as estruturas SMRandom construídas por meio da política MinMax e MST possuem os menores tempos totais em consultas (Figura 25 c), sendo que esta última apresenta melhor desempenho. Além disso, a SMDensity construída por meio da política MinMax, apresenta melhor desempenho quando comparadas a mesma estrutura construída pela política MST. No que se refere ao emprego da política DM, notou-se que esta não apresentou melhoras significativa que justifique a utilização para otimizar as consultas por similaridade para este conjunto de dados em específico. Além disso, as estruturas propostas apresentam maiores quantidades de acesso a disco durante as consultas, como apresentado na Figura 25 a. Este resultado é decorrente das estruturas produzirem árvores com maiores números de nós. Outro fator a ser considerado é que os métodos propostos são impactados por características intrínsecas do conjunto de dados. Nesse caso, esta base é pouco discriminativa, pois é afetada pela maldição da alta dimensionalidade.

Resultados similares aos apresentados nas três etapas dos experimentos, foram encontrados para outros conjuntos de dados, tais como: *Eigenfaces*, *Letter*, *Nasa*, *Corel*, *SynEDC*, *Covertype*. Como estes resultados não conduzem a novas descobertas, preferiu-se não detalha-los, visto o teor aprofundado das análises apresentadas anteriormente neste capítulo. Contudo, as Tabelas 8 e 9 apresentam um resumo de todos os resultados encontrados nos experimentos, o que irá permitir responder as questões formulas em cada uma

das etapas que conduziram os experimentos. As perguntas formuladas foram: *"Q1: É possível gerar árvores com custo de construção inferior com o uso short-term memory ?"*; *"Q2: É possível gerar árvores com grau de sobreposição inferior ao fazer uso short-term memory no processo de construção da estrutura de indexação?"* e *"Q3: É possível obter consultas por similaridade mais eficientes ao utilizar-se da estrutura short-term memory no processo de construção dos métodos?"*.

Tabela 8 – Resumo dos resultados encontrados no que tange os questionamentos levantados nas epatas que direcionaram os experimentos, para os conjuntos de dados *Pendigits*.

Base de dados	**Questões**	**Política DM**			**Política MinMax**			**Política MST**		
		SMR	**SMD**	**SMC**	**SMR**	**SMD**	**SMC**	**SMR**	**SMD**	**SMC**
Pendigits	*Q1*	não	não	não	não	não	não	não	não	não
	Q2	sim	sim	sim	sim	sim	sim	sim	sim	sim
	Q3	sim	sim	sim	sim	sim	sim	sim	sim	sim

Com relação aos resultados apresentados nas Tabelas 8 e 9, as estruturas propostas demandam maior custo para a construção. No entanto, quando construídas por meio da política MinMax, na maioria dos experimentos realizados, estas conseguem reduzir significativamente o número cálculos de distância e, desse modo, reduzir o tempo total de construção superando nesses casos a Slim-Tree. Com relação a sobreposição das estruturas resultantes foi possível notar que as novas estruturas derivadas para todos os casos resultaram na minimização da sobreposição e, consequentemente contribuiu para melhorar o desempenho das consultas por similaridade em todos os casos quando comparadas as estruturas construídas por meio da mesma política.

4.3 Considerações

Este capítulo apresentou os resultados dos experimentos e as análises realizadas para os principais parâmetros considerados na literatura. Avaliou-se o processo de construção e consultas por similaridade sobre a perspectiva de diferentes políticas de divisão de nós, no que se refere a eficiência e qualidade dos métodos propostos quando comparados com a estrutura Slim-Tree original. Para tal, foi utilizado uma série de bases de dados com peculiaridades distintas e o emprego de consultas k-nn, variando o parâmetro k. É importante salientar que como descrito no Capítulo 3, ao modificar o processo de inserção padrão para os elementos que foram movidos para *short-term memory*, há um custo adicional para inseri-los posteriormente, isto porque requer cálculos de dissimilaridade que não são requeridos pelo índice original. Além disso, requer travessia no índice resultando em acessos a disco adicionais para a construção dos nós folhas completos. No entanto, como apresentado neste capítulo, esse custo foi compensado mediante a redução do número de

Tabela 9 – Resumo dos resultados encontrados no que tange os questionamentos levantados nas epatas que direcionaram os experimentos, para os conjuntos de dados avaliados .

Base de dados	**Questões**	**Política DM**		**Política MinMax**		**Política MST**	
		SMR	**SMD**	**SMR**	**SMD**	**SMR**	**SMD**
Eigenfaces	*Q1*	não	não	sim	sim	não	não
	Q2	sim	sim	sim	sim	sim	sim
	Q3	sim	sim	sim	sim	sim	sim
Letter	*Q1*	não	não	sim	não	não	não
	Q2	sim	sim	sim	sim	sim	sim
	Q3	sim	sim	sim	sim	sim	sim
Nasa	*Q1*	não	não	sim	sim	não	não
	Q2	sim	sim	sim	sim	sim	sim
	Q3	sim	sim	sim	sim	sim	sim
Corel	*Q1*	não	não	sim	não	não	não
	Q2	sim	sim	sim	sim	sim	sim
	Q3	sim	sim	sim	sim	sim	sim
SynD	*Q1*	não	não	não	não	não	não
	Q2	sim	sim	sim	sim	sim	sim
	Q3	sim	sim	sim	sim	sim	sim
Covertype	*Q1*	não	não	sim	sim	não	não
	Q2	sim	sim	sim	sim	sim	sim
	Q3	sim	sim	sim	sim	sim	sim
Colour Structure	*Q1*	não	não	sim	sim	não	não
	Q2	sim	sim	sim	sim	sim	sim
	Q3	sim	sim	sim	sim	sim	sim
SynEDC	*Q1*	não	não	sim	não	não	não
	Q2	sim	sim	sim	sim	sim	sim
	Q3	sim	sim	sim	sim	sim	sim

operações de divisão de nós e, assim as políticas com custo maior nestas operações foram mais beneficiadas. Nesse sentido a política MinMax aplicada a construção das estruturas propostas teve seu número de cálculos de distância na criação dos índices reduzidos significativamente, principalmente para bases onde a cardinalidade e dimensionalidade eram maiores, por exemplo, *SynD* e *Colour Structure*. Essa redução, impactou positivamente para melhorar o tempo de construção das árvores sendo possível construir árvores mais rápidas do que Slim-Tree padrão em alguns casos. Além disso, a qualidade das árvores foi melhorada em todos os casos avaliados quando comparadas as medidas fator-relativo Tabela 7 e 6 .

Os métodos propostos apresentaram maiores quantidades de acessos a páginas de disco durante a construção, exceto para as estruturas construídas pela política MST. Em relação a essas estruturas algumas conseguem igualar a quantidade de acessos realizados

pela Slim-Tree, devido a diminuição na quantidade de nós resultante, o que contribui para menos acessos a disco tanto durante a construção quanto durante as operações de consultas por similaridade.

Os tempos de construção das estruturas propostas SMRandom e SMDensity para a base de dados *Pendigits* foram maiores do que a Slim-Tree, porém, ao considerar os ganhos de desempenho em consultas por similaridade, o contexto "custo vs. benefício" compensa o aumento nos tempos de construção. A mesma ideia não pode ser aplicada ao SMCluster visto o alto custo necessário para a construção dos índices e o pouco ganho resultante nas consultas quando comparadas com as demais estruturas propostas.

Os resultados do desempenho das consultas por similaridade em todos os casos foram melhorados com o emprego das técnicas propostas na construção das estruturas métricas SMRandom, SMDensity e SMCluster em comparação com a estrutura Slim-Tree.

Por meio dos bons resultados apresentados nos experimentos, conclui-se que a estrutura *short-term memory* pode ser utilizada na construção de métodos de acesso métricos mais eficientes para acelerar o processo de consultas por similaridade e, também em alguns casos acelerar o processo de construção das estruturas.

Capítulo 5

Conclusão

O desenvolvimento de métodos de acessos métricos dinâmicos sobre dados complexos mais eficientes é fundamental para o processamento de consultas por similaridade e devem satisfazer a requisitos mínimos como: baixo tempo para realização de consultas, pouco espaço para manutenção dos índices e qualidades dos resultados.

Há diversas publicações sobre a temática MAM dinâmicos em que são propostas soluções consideradas como estratégias do tipo dinâmicas ou estáticas. Com relação às soluções dinâmicas a maioria destas têm como foco melhorar os mecanismos de particionamento quando os nós estão quase cheios (técnicas de reinserção forçadas) ou ainda quando estes atingem sua capacidade máxima (políticas de divisão de nós). Essas soluções têm como principal desvantagem tentar solucionar um problema que já ocorreu (nesse caso as subárvores já possuem nós sobrepostos), dessa maneira, o que é realizado é a tentativa de reorganizar os elementos entre os nós com a finalidade de obter melhor distribuição dos dados. Apesar de trazer bons resultados ao reduzir a sobreposição, a solução ocorre localmente entre os nós.

Nesse sentido, a proposta desta dissertação configurou-se como uma solução alternativa aos estudos anteriores ao propor uma nova estratégia baseada em suposições de adiamento da indexação, conceitos derivados áreas de *data streams*, para promover um ambiente mais abrangente de processamento de elementos, possibilitando a descoberta de melhores grupos de elementos e com distribuição mais uniforme. Assim, este estudo foi direcionado e desenvolvido com o intuito de validar a hipótese formulada na Seção 1.3 e, desse modo, foi definido como objetivo geral, propor e desenvolver métodos de acesso métrico dinâmicos derivados da Slim-Tree assistidos por *short-term memory*, na perspectiva de diferentes políticas de divisão de nós com intuito de otimizar as operações de consulta por similaridade. Além disso, para auxiliar a satisfazer o objetivo geral, delimitaram-se os objetivos específicos explicitados na Seção 1.2.

Com relação a estes objetivos, a compreensão do Capítulo 2, especificamente das Seções 2.4.1 Métodos de Acesso Métricos Dinâmicos e Seção 2.5 Trabalhos correlatos, foram fundamentais para que os quatro primeiros objetivos específicos i), ii), iii) e iv)

fossem propostos e alcançados, sendo primordial o entendimento do funcionamento geral MAM Slim-Tree e das peculiaridades referentes ao processo de divisão de nós no que se tange às implicações decorrentes adoção de diferentes políticas de divisão de nós. Além disso, foi necessário formular estratégias de agrupamento de dados para criação de nós folhas completos, respeitando o tamanho das páginas de disco e a taxa de ocupação dos nós. E ainda, a implementação de novos algoritmos para inserção de nós folhas que não comprometa o balanceamento da árvore de indexação.

Os demais objetivos foram atingidos por meio de formulações e pela realização de experimentos, como descritos no Capítulo 3. No que se refere ao objetivo específico v), os experimentos foram realizados com uma quantidade considerável de dados reais e sintéticos. Cada conjunto de dados foi testado e executado no mínimo três vezes para obtenção de resultados mais precisos. Os objetivos subsequentes tiveram por finalidade avaliar os parâmetros de desempenho e a qualidade das estruturas resultantes. Como método de comparação foi utilizado a estrutura Slim-Tree padrão diversificando o algoritmo de divisão de nós.

Segundo os resultados obtidos nos experimentos, notou-se que o uso das abordagens propostas para a construção proporciona ganhos expressivos em todos os cenários apresentados e avaliados neste trabalho. Especificamente ao que se refere ao desempenho em operações de consultas por similaridade Seção 4.2.4, as novas estruturas desenvolvidas são mais eficientes quando comparadas a Slim-tree. Na comparação intragrupos (MAMs construídos com as mesmas políticas de divisão de nós), é possível notar que as estruturas que alcançaram melhores resultados foram as construídas por meio da política MST. Esse resultado é decorrente do fato da técnica proporcionar nós folhas completos com distribuição uniforme, sendo que na literatura o principal problema da política MST consiste em gerar nós com distribuição não uniforme e, por isso colaborou para otimização da estrutura resultante.

Nas comparações intergrupos (MAMs construídos com diferentes políticas de divisão de nós) é possível notar que o melhor desempenho ocorre para estruturas construídas por meio da política MinMax, contudo, as estruturas geradas por meio da política MST teve seu desempenho em consultas muito próximos a estas. Além disso, foi possível observar em todos os casos testados e avaliados que apesar da política DM resultar em um tempo melhor de construção, ela não atinge resultados expressivos no que se refere a qualidade das árvores resultantes e, consequentemente, as operações de consultas por similaridade não são eficientes quando forem comparadas às demais estruturas construídas por diferentes políticas.

É importante salientar que os bons resultados das consultas são decorrentes dos ganhos expressivos no que refere-se à qualidade das estruturas resultantes construídas por meio estratégia de adiamento da indexação. Os melhores resultados no que se refere a construção, foram para experimentos sobre o conjunto cuja cardinalidade e dimensionalidade

eram maiores e a bases eram mais discriminativas, ou seja, os dados estavam melhores distribuídos.

Mesmo que alcançados todos os objetivos do estudo é possível citar algumas limitações no processo de desenvolvimento da pesquisa. Dentre elas, destacam-se as limitações em relação à técnica proposta, onde verificou-se por meio dos experimentos que o número de acessos a disco e de cálculos de distâncias em alguns casos para realizar a construção dos métodos é maior quando comparado com a Slim-Tree.

Outro importante fator limitante da pesquisa está relacionado ao tempo para realização dos testes, pois conforme aumentava a quantidade de dados dos conjuntos, o tempo de espera estimado das execuções era longo e por isso, não houve tempo hábil para explorar com maiores detalhes as peculiaridades das bases de dados, tais como: cardinalidade, dimensionalidade e distribuição de dados. Sendo que esse fator pode ser determinante para melhorar os resultados das técnicas propostas.

As limitações apresentadas anteriormente, juntamente com outras ideias para melhorar os estudos que permearam essa pesquisa, são apresentadas como sugestões para trabalhos futuros na Seção 5.2. A dissertação teve como foco tratar o problema da sobreposição decorrente das estruturas métricas dinâmicas M-Tree, Slim-Tree e derivadas e, diferencia-se das demais ao propor uma solução mais abrangente para melhorar a distribuição hierárquica dos dados, por meio da estratégia de adiamento da indexação ao fazer uso da estrutura *short-term memory* para otimizar o processo de construção de novas estruturas de indexação. Os resultados foram significativos no que se refere à qualidade das árvores resultantes e que, consequentemente, impactaram em melhor eficiência nas operações de consultas por similaridade.

5.1 Principais Contribuições

Esta pesquisa contribuiu para ampliar o conhecimento dos estudos em relação a técnicas aplicadas a Métodos de Acesso Métricos dinâmicos inserida na área de estrutura de dados. Especificamente, as principais contribuições deste trabalho podem ser resumidas em:

1. Extensão MAM Slim-Tree para inclusão da estrutura *short-term memory* no processo de construção de novos métodos derivados otimizados;

2. Três novas abordagens para criação de nós folhas completos;

3. Uma nova abordagem para inserção de nós folhas completos;

4. No que se refere a construção da estrutura: redução na quantidade de cálculos de distância; redução no tempo total de construção e, redução da sobreposição entre nós folhas das subárvores;

5. No que se refere a consultas por similaridade: redução na quantidade total de acessos a páginas de disco; redução na quantidade de cálculos de distância e, redução no tempo de execução de consultas;

6. Otimização significativa no processo de construção e consultas das estruturas métricas que fazem uso da política MST para divisão de nós.

Além disso, outro ponto importante relacionados as abordagens propostas é que elas não exigem inclusão de novas informações nos métodos de acesso, ou seja, não há alterações nas estruturas de dados. As técnicas foram integradas ao método Slim-Tree e podem ser integradas facilmente a outros métodos dinâmicos.

5.2 Trabalhos Futuros

Como sugestões de trabalhos futuros, torna-se possível almejar a construção de métodos de acesso métricos assistidos por *short-term memory* mais amplo, tais como:

- Utilizar novos algoritmos para processar os elementos da *short-term memory* com um custo inferior ao método *Cluster* que seja viável a utilização em base de dados volumosas;

- Explorar novas estratégias de divisão de nós que se beneficie das técnicas propostas;

- Propor e desenvolver algoritmos de inserção de nó folhas por meio do uso de abordagem *top-down* quando necessário. A motivação é resolver o problema decorrente das sobreposições não apenas no nível das folhas como é proposto, mas também no nível dos índices;

- Implementar a estrutura *short-term memory* persistente em disco;

- Um estudo aprofundado para desenvolvimento de outras estratégias para permitir reduzir ainda mais o número de acesso a página de disco, visto que esse parâmetro é um dos pontos fracos da abordagem empregada na construção dos métodos.

5.3 Contribuições em Produção Bibliográfica

- Publicados:
  SOUSA, R. M. S.; RAZENTE, H. L. ; BARIONI, M. C. N. "Explorando o Uso de Short-term Memory na Construção de Métodos de Acesso Métricos Mais Eficientes". In: 31o Simpósio Brasileiro de Banco de Dados, 2016, Salvador, p. 163-168.

- Submetidos:SOUSA, R. M. S.; RAZENTE, H. L. ; BARIONI, M. C. N. “Exploring the use of a Short-term Memory in the Construction of Efficient Metric Access Methods”. In: DEXA, 2017.

Referências

BECKMANN, N. et al. The r*-tree: an efficient and robust access method for points and rectangles. In: ACM. **ACM Sigmod Record**. [S.l.], 1990. v. 19, n. 2, p. 322–331.

BENTLEY, J. L. Multidimensional binary search trees used for associative searching. **Communications of the ACM**, ACM, v. 18, n. 9, p. 509–517, 1975.

BOLETTIERI, P. et al. CoPhIR: a test collection for content-based image retrieval. **CoRR**, abs/0905.4627v2, 2009. Disponível em: <http://cophir.isti.cnr.it>.

BOZKAYA, T.; OZSOYOGLU, M. Distance-based indexing for high-dimensional metric spaces. In: ACM. **ACM SIGMOD Record**. [S.l.], 1997. v. 26, n. 2, p. 357–368.

BRIN, S. Near neighbor search in large metric spaces. In: **VLDB'95, Proceedings of 21th International Conference on Very Large Data Bases, September 11-15, 1995, Zurich, Switzerland.** [s.n.], 1995. p. 574–584. Disponível em: <http://www.vldb.org/conf/1995/P574.PDF>.

BUGATTI, P. H.; TRAINA, A.; TRAINA-JR, C. Assessing the best integration between distance-function and image-feature to answer similarity queries. In: ACM. **Proceedings of the 2008 ACM symposium on Applied computing**. [S.l.], 2008. p. 1225–1230.

CARÉLO, C. C. M. et al. Slicing the metric space to provide quick indexing of complex data in the main memory. **Information Systems**, Elsevier, v. 36, n. 1, p. 79–98, 2011.

CHÁVEZ, E. et al. Searching in metric spaces. **ACM computing surveys (CSUR)**, ACM, v. 33, n. 3, p. 273–321, 2001.

CHINO, F. J. T. **Visualizando a organizaçao e o comportamento de estruturas métricas: Aplicaçoes em consultas por similaridade1**. Tese (Doutorado) — Instituto de Ciências Matemáticas e de Computaçao, 2004.

CIACCIA, P.; PATELLA, M.; ZEZULA, P. M-tree: An efficient access method for similarity search in metric spaces. In: **VLDB**. Atenas, Grécia: [s.n.], 1997. p. 426–435.

COMER, D. Ubiquitous b-tree. **ACM Computing Surveys (CSUR)**, ACM, v. 11, n. 2, p. 121–137, 1979.

DROZDEK, A.; PAIVA, L. S. de C. **Estrutura de Dados e Algoritmos em C++**. [S.l.]: Pioneira Thomson Learning, 2002.

ESTER, M. et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In: **Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96), Portland, Oregon, USA**. [s.n.], 1996. p. 226–231. Disponível em: <http://www.aaai.org/Library/KDD/1996/kdd96-037.php>.

FARIA, E. R.; CARVALHO, A. C. P. de Leon Ferreira de; GAMA, J. MINAS: multiclass learning algorithm for novelty detection in data streams. **Data Min. Knowl. Discov.**, v. 30, n. 3, p. 640–680, 2016. Disponível em: <http://dx.doi.org/10.1007/s10618-015-0433-y>.

FARIA, E. R. et al. Novelty detection in data streams. **Artificial Intelligence Review**, Springer, v. 45, n. 2, p. 235–269, 2016.

FARIA, E. R. et al. Novelty detection in data streams. **Artif. Intell. Rev.**, v. 45, n. 2, p. 235–269, 2016. Disponível em: <http://dx.doi.org/10.1007/s10462-015-9444-8>.

GUTTMAN, A. R-trees: A dynamic index structure for spatial searching. In: **SIGMOD**. Boston, Massachusetts: [s.n.], 1984. p. 47–57.

LICHMAN, M. **UCI Machine Learning Repository, University of California, Irvine, School of Information and Computer Sciences, http://archive.ics.uci.edu/ml**. 2013.

LIM, S.-H. et al. A node split algorithm reducing overlapped index spaces in m-tree index. In: IEEE. **International Conference on Data Engineering Workshop (ICDEW)**. [S.l.], 2006. p. 15–15.

LOKOC, J.; SKOPAL, T. On reinsertions in m-tree. In: IEEE. **International Conference on Data Engineering Workshop (ICDEW)**. [S.l.], 2008. p. 410–417.

MAMEDE, M.; BARBOSA, F. Range queries in natural language dictionaries with recursive lists of clusters. In: IEEE. **Computer and information sciences, 2007. iscis 2007. 22nd international symposium on**. [S.l.], 2007. p. 1–6.

MASUD, M. M. et al. Classification and novel class detection in concept-drifting data streams under time constraints. **IEEE Trans. Knowl. Data Eng.**, v. 23, n. 6, p. 859–874, 2011. Disponível em: <http://dx.doi.org/10.1109/TKDE.2010.61>.

NAVARRO, G.; REYES, N. New dynamic metric indices for secondary memory. **Information Systems**, v. 59, p. 48–78, 2016.

NAVARRO, G.; URIBE-PAREDES, R. Fully dynamic metric access methods based on hyperplane partitioning. **Information Systems**, Elsevier, v. 36, n. 4, p. 734–747, 2011.

NG, R. T.; HAN, J. Clarans: A method for clustering objects for spatial data mining. **IEEE transactions on knowledge and data engineering**, IEEE, v. 14, n. 5, p. 1003–1016, 2002.

OLIVEIRA, P.; TRAINA-JR, C.; KASTER, D. Improving the pruning ability of dynamic metric access methods with local additional pivots and anticipation of information. In: SPRINGER. **ADBIS, LNCS 9282**. Poitiers, França, 2015. p. 18–31.

SKOPAL, T. On fast non-metric similarity search by metric access methods. In: SPRINGER. **EDBT, LNCS 3896**. Munique, Alemanha, 2006. p. 718–736.

SKOPAL, T.; LOKOČ, J. New dynamic construction techniques for m-tree. **Journal of Discrete Algorithms**, Elsevier, v. 7, n. 1, p. 62–77, 2009.

SKOPAL, T. et al. Revisiting m-tree building principles. In: SPRINGER. **East European Conference on Advances in Databases and Information Systems**. [S.l.], 2003. p. 148–162.

SOUZA, J.; RAZENTE, H.; BARIONI, M. C. Optimizing metric access methods for querying and mining complex data types. **Journal of the Brazilian Computer Society**, Springer, v. 20, n. 1, p. 1, 2014.

TEIXEIRA, J. W. **Processamento de consultas analíticas com predicados de similaridade entre imagens em ambientes de data warehousing**. Tese (Doutorado) — Universidade de São Paulo, 2013.

TRAINA, A. et al. The metric histogram: A new and efficient approach for content-based image retrieval. In: **Visual and Multimedia Information Management**. [S.l.]: Springer, 2002. p. 297–311.

TRAINA-JR, C. et al. The omni-family of all-purpose access methods: a simple and effective way to make similarity search more efficient. **The VLDB Journal**, Springer, v. 16, n. 4, p. 483–505, 2007.

TRAINA-JR, C. et al. Fast indexing and visualization of metric data sets using slim-trees. **IEEE Transactions on Knowledge and Data Engineering**, IEEE, v. 14, n. 2, p. 244–260, 2002.

TRAINA-JR, C. et al. How to improve the pruning ability of dynamic metric access methods. In: ACM. **Proceedings of the eleventh international conference on Information and knowledge management**. [S.l.], 2002. p. 219–226.

TRAINA-JR, C. et al. Slim-trees: High performance metric trees minimizing overlap between nodes. In: **Advances in Database Technology - EDBT 2000, 7th International Conference on Extending Database Technology, Konstanz, Germany, March 27-31, 2000, Proceedings**. Springer, 2000. p. 51–65. Disponível em: <http://dx.doi.org/10.1007/3-540-46439-5_4>.

UHLMANN, J. K. Satisfying general proximity/similarity queries with metric trees. **Information processing letters**, Elsevier, v. 40, n. 4, p. 175–179, 1991.

VESPA, T. G.; TRAINA-JR, C.; TRAINA, A. Efficient bulk-loading on dynamic metric access methods. **Information Systems**, Elsevier, v. 35, n. 5, p. 557–569, 2010.

VIEIRA, M. R. et al. Dbm-tree: A dynamic metric access method sensitive to local density data. **Journal of Information and Data Management**, v. 1, n. 1, p. 111–128, 2010.

YAMAMOTO, C. H.; BIAJIZ, M.; TRAINA-JR, C. Aumento da eficiência das estruturas de indexação métricas com uso de conceitos da lógica nebulosa. In: **XVIII Simpósio Brasileiro de Bancos de Dados (SBBD), 6-8 de Outubro, Manaus, Amazonas, Brasil, Anais/Proceedings.** [S.l.: s.n.], 2003. p. 185–199.

YIANILOS, P. N. Data structures and algorithms for nearest neighbor search in general metric spaces. In: SOCIETY FOR INDUSTRIAL AND APPLIED MATHEMATICS. **Proceedings of the fourth annual ACM-SIAM Symposium on Discrete algorithms**. [S.l.], 1993. p. 311–321.

YIANILOS1, P. N. Excluded middle vantage point forests for nearest neighbor search. In: CITESEER. **In DIMACS Implementation Challenge, ALENEX'99**. [S.l.], 1999.

ZEZULA, P. et al. **Similarity search: the metric space approach**. [S.l.]: Springer Science & Business Media, 2006. v. 32.