UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

ORTree: Aumentando a eficiência de buscas por similaridade diversificadas por meio de particionamento de dados

João Victor de Oliveira Novaes

Dissertação de Mestrado do Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional (PPG-CCMC)

ICMC USP
SÃO CARLOS

João Victor de Oliveira Novaes

ORTree: Aumentando a eficiência de buscas por similaridade diversificadas por meio de particionamento de dados

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

Área de Concentração: Ciências de Computação e Matemática Computacional

Orientador: Prof. Dr. Caetano Traina Júnior

USP – São Carlos
Janeiro de 2023

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

N935o

Novaes, João Victor de Oliveira
ORTree: Aumentando a eficiência de buscas por similaridade diversificadas por meio de particionamento de dados / João Victor de Oliveira Novaes; orientador Caetano Traina Júnior. -- São Carlos, 2023.
94 p.

Dissertação (Mestrado - Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional) -- Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2023.

1. Buscas por similaridade. 2. Busca por similaridade com diversidade. 3. Busca em espaços métricos. 4. Métodos de acesso métrico. I. Traina Júnior, Caetano, orient. II. Título.

Bibliotecários responsáveis pela estrutura de catalogação da publicação de acordo com a AACR2:
Gláucia Maria Saia Cristianini - CRB - 8/4938
Juliana de Souza Moraes - CRB - 8/6176

João Victor de Oliveira Novaes

ORTree: Tuning diversified similarity queries using data partitioning

Dissertation submitted to the Instituto de Ciências Matemáticas e de Computação – ICMC-USP – in accordance with the requirements of the Computer and Mathematical Sciences Graduate Program, for the degree of Master in Science. *FINAL VERSION*

Concentration Area: Computer Science and Computational Mathematics

Advisor: Prof. Dr. Caetano Traina Júnior

USP – São Carlos
January 2023

Este trabalho é dedicado a todos que acreditam e tornam a ciência possível.

AGRADECIMENTOS

À Deus.

Ao meu orientador, Prof. Dr. Caetano Traina Júnior, agradeço todo o apoio, confiança e conselhos dados durante o desenvolvimento do mestrado.

Ao professor e amigo, Lúcio Fernandes Dutra Santos, por ter conduzido o encontro com o Prof. Caetano e, por todos os conselhos, conversas e sugestões.

À Alane Ferreira de Almeida, pelo amor, apoio incondicional, paciência, incentivo e, que mesmo na distância se fez sempre presente.

Aos meus pais, pelo amor, incentivo e por serem exemplos de tudo que quero ser.

Ao meu irmão Vinícius Jhônata de Oliveira Novaes, que mesmo eu não estando presente, sempre me deu apoio, amor e incentivo.

Aos amigos de república Endi Daniel Coelho Silva e Thiago de Jesus Oliveira Durães, por todos os momentos compartilhados durante o mestrado.

Aos amigos do GBdI, especialmente, Afonso Matheus Sousa Lima, José Maria Clementino, Maxwell Sampaio dos Santos e Natália de Fátima Martins. Aos demais amigos do grupo, muito obrigado por todas as oportunidades de compartilhar ideias e momentos com vocês.

Agradeço o fundamental apoio financeiro da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

Aos funcionários e professores do ICMC/USP.

“O mais corajoso dos atos ainda é pensar com a própria cabeça.”
(Coco Chanel)

RESUMO

NOVAES, J. V. O. **ORTree: Aumentando a eficiência de buscas por similaridade diversificadas por meio de particionamento de dados**. 2023. 94 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2023.

A complexidade dos dados aumenta conforme as aplicações vão evoluindo, sendo sempre necessário desenvolver novas técnicas para o seu armazenamento e recuperação. Neste sentido, as buscas por similaridade têm se mostrado uma das melhores formas de se comparar/recuperar dados complexos. Contudo, ao serem aplicados em grandes conjuntos de dados, os operadores fundamentais de busca por similaridade têm sua expressividade reduzida, e os elementos recuperados tendem a ser muito similares entre si. Para solucionar este problema, vários pesquisadores têm considerado a inclusão de diversidade nas buscas por similaridade. O objetivo deste tipo de busca é encontrar um conjunto de elementos que sejam similares ao elemento de consulta ao mesmo tempo que sejam o mais diversos possível entre si. Enquanto uma busca por similaridade pode ser feita de forma simples, uma busca por similaridade com diversidade tende a ser mais complexa, pois se torna necessário comparar os elementos da resposta entre si e, portanto executar um número maior de comparações, o que torna a busca mais lenta e custosa. Na literatura são encontradas abordagens que visam reduzir os custo dessas buscas, uma delas é a de selecionar elementos candidatos. Neste caso, ao invés de utilizar todos elementos do conjunto de dados, apenas uma pequena amostra do conjunto é de fato utilizada pelos algoritmos de diversidade. O foco principal dessa dissertação é desenvolver abordagens de seleção de candidatos que sejam escaláveis e que permitam selecionar elementos candidatos de alta qualidade. Neste sentido, são apresentadas: uma nova estrutura de indexação baseada em particionamento hierárquico de dados; e três abordagens de seleção de elementos candidatos, que utilizam o particionamento gerado pela estrutura para encontrar de forma rápida elementos candidatos adequados.

Palavras-chave: Buscas por similaridade, Busca por similaridade com diversidade, Busca em espaços métricos, Métodos de acesso métrico.

ABSTRACT

NOVAES, J. V. O. **ORTree: Tuning diversified similarity queries using data partitioning**. 2023. 94 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2023.

The complexity of data increases as the applications evolve, and it is always necessary to develop new techniques for its storage and retrieval. In this sense, similarity search have been shown to be one of the best ways to compare/recover complex data. However, when applied to large data sets, the fundamental similarity search operators have their expressiveness reduced and the retrieved elements tend to be very similar to each other. To solve this problem, several researchers have considered including diversity in the similarity searches. The objective of this type of search is to find a set of elements that are similar to the query element while being as diverse as possible from each other. While a search for similarity can be done in a simple way, a search for similarity with diversity tends to be more complex, as it becomes necessary to compare the elements of the answer with each other and, therefore, perform a greater number of comparisons, which makes the search slower and more expensive. In the literature are found approaches that aim to reduce the cost of these searches. One of them is to select candidate elements. In this case, instead of using all elements of the dataset, only a small sample of the set is actually used by the diversity algorithms. The main focus of this dissertation is to develop candidate selection approaches that are scalable and allow the selection of high-quality candidate elements. In this sense, the following results obtained are described: a new indexing structure based on hierarchical data partitioning; and three candidate element selection approaches, which use the partitioning generated by the structure, to quickly find candidate elements.

Keywords: Similarity queries, Similarity with diversity queries, Query in metric spaces, Metric access methods.

LISTA DE ILUSTRAÇÕES

LISTA DE ALGORITMOS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

s_q — Elemento central de consulta

s_i, s_j — Elemento da base de dados

δ — Função de distância métrica

δ_{sim} — Função de avaliação de similaridade

δ_{div} — Função de avaliação de diversidade

$\mathbb{S}$ — Domínio de dados

$\mathbb{M} =< \mathbb{S}, \delta >$ — Espaço métrico

Rq — Busca por abrangência

ξ — Limiar de distância (dissimilaridade)

$k-NNq$ — Busca aos k-vizinhos mais próximos

R — Conjunto Reposta

$\mathfrak{D}$ — Dimensão fractal

ξ_k — Limiar de distância, com k elementos

Rd_r — Intervalo de Busca para o espaço de características

λ — Preferência de Diversidade

$I(s_i, s_j)$ — Influência entre dois elementos

SUMÁRIO

CAPÍTULO

1

INTRODUÇÃO

1.1 Contexto

Os Sistemas de Gerenciamento de Banco de Dados (SGBD) foram inicialmente projetados para armazenar e recuperar de forma eficiente grandes conjuntos de dados. Tipicamente, esses sistemas suportavam apenas dados como números e cadeias de caracteres. Com a evolução das formas de aquisição e das aplicações dos dados, tornou-se necessário armazenar e recuperar dados mais complexos, tais como: coordenadas geográficas, áudios, imagens, vídeos, dados vetoriais, séries temporais. Entretanto, os SGBDs continuam sendo baseados nas relações de identidade (RI - $=, \neq$) e de ordem (RO - $<, \geq, >\leq$), as quais nem sempre podem ser aplicadas em dados complexos, e nos casos em que podem ser aplicadas têm sua serventia reduzida, pois, geralmente os elementos complexos estão representados em domínio de dados que não atendem as propriedades de RO. Por conta disso, a busca por elementos complexos é geralmente executada comparando-se atributos (características) que os descrevem ao invés de se comparar os objetos diretamente (SANTOS, 2017; BÊDO, 2017; SOUZA, 2018; AVALHAIS, 2012).

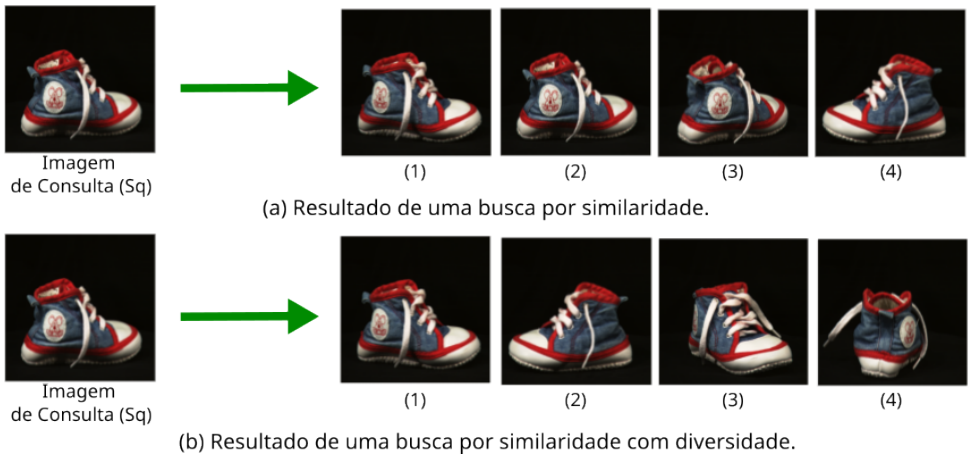
Uma técnica que associa atributos a elementos complexos é a de recuperação por conteúdo, como por exemplo, a Recuperação de Imagens por Conteúdo (CBIR do inglês – *Content-Based Image Retrieval*). Essa abordagem utiliza algoritmos semi-automáticos ou automáticos para gerar as características dos elementos complexos. Tais características são representadas por meio de dados estruturados, comumente chamados de vetores de características. Para recuperar os elementos, essa técnica utiliza funções de distância que quantificam o quão próximos dois vetores de características são; assumindo que quanto mais próximos dois vetores de características são, mais similares os elementos complexos correspondentes são. Desta forma, temos o conceito de busca por similaridade, que recupera os elementos baseando-se no quão similar eles são em relação a um dado elemento de referência, chamado de elemento central consulta (s_q) (SANTOS; et. al, 2018; NOVAES; et. al, 2019; SOUZA, 2018).

No entanto, existem problemas que podem reduzir a qualidade dos resultados gerados por uma busca por similaridade, sendo que o aumento da densidade e do volume dos dados é um deles. Conforme a base de dados cresce, as buscas por similaridade tendem a perder expressividade, retornando muitos elementos similares ao elemento central de consulta, que também são muito similares entre si. Isso acontece, principalmente, por que conforme o número de elementos aumenta, a quantidade de elementos muito similares entre si na base de dados também tende a aumentar e, por consequência, muitos dos elementos retornados pelas buscas por similaridade tendem a ser similares entre si. O problema com muitos elementos similares é que eles não agregam valor ao resultado gerado e, podem induzir o usuário a reformular a busca com frequência, a fim de encontrar outro resultado mais interessante (SANTOS, 2017; ZHENG *et al.*, 2017; DROSOU; PITOURA, 2014; SKOPAL *et al.*, 2009). Para solucionar tal problema, muitos pesquisadores/trabalhos têm considerado a inclusão do conceito de diversidade nas buscas por similaridade. As técnicas de diversificação de resultados consideram que os elementos presentes na resposta da busca não devem ter apenas similaridade em relação ao elemento central de consulta, mas também devem ter um certo grau de dissimilaridade entre eles, garantindo assim, uma visão mais holística dos elementos na base de dados (NOVAES; et. al, 2019; SANTOS, 2017; ZHENG *et al.*, 2017; SKOPAL *et al.*, 2009). Na literatura podem ser encontradas diferentes abordagens de diversificação de resultados: as baseadas em atributos, que utilizam informações externas (rótulos, taxonomias, log de buscas, etc.) (ZHENG *et al.*, 2017; AGRAWAL *et al.*, 2009) e as baseadas em conteúdo, que utilizam a relação de distância entre os próprios elementos (ZHENG *et al.*, 2017; VIEIRA *et al.*, 2011). A qualidade das respostas geradas pelas abordagens que utilizam informações externas dependem da disponibilidade e conformidade dessas informações, além de terem custo computacional mais elevado (VIEIRA *et al.*, 2011). Já as baseadas na similaridade entre os elementos, podem utilizar os mesmos vetores de características utilizados durante a busca por similaridade, para gerar as respostas diversificadas.

Na literatura podem ser encontradas diferentes aplicações que utilizam o conceito de diversidade para melhorar o resultados apresentados aos seus usuários, por exemplo, os sistemas de recomendação (ZHENG *et al.*, 2017; KUNAVER; POŽRL, 2017) e os sistemas de recuperação (NOVAES; et. al, 2019; SANTOS; et. al, 2018; DROSOU; PITOURA, 2013). Em ambos os casos o objetivo é recuperar dados que sejam relevantes para o usuário, mas que possuam algumas informações diferentes entre os dados retornados. Por exemplo em um sistema de recuperação, como um CBIR, o principal objetivo é recuperar imagens que são similares a uma imagem de consulta, mas que tenham informações visuais diferentes(diversas) entre si. A Figura 1 ilustra esse processo utilizando algumas imagens da base de dados Aloi[1]. Na Figura 1(a) é exibido o resultado de uma busca por similaridade convencional, como pode ser observado algumas das imagens retornadas (1, 2 e 3) são muito similares entre si e quase não agregam novas informações ao resultado da busca. Já na Figura 1(b) é exibido o resultado de uma busca por

[1] Disponível em: <https://aloi.science.uva.nl/>, acessado em: 24/01/2023.

Figura 1 – Ilustração do resultado de buscas por similaridade sem e com diversidade utilizando imagens da base de dados Aloi. (a) Resultado de uma busca por similaridade. (b) Resultado de uma busca por similaridade com diversidade.



Fonte: Elaborada pelo autor.

similaridade com diversidade, neste caso, todas as imagens recuperadas são similares a imagem de consulta mas apresentam novas informações sobre objeto presente nas imagens. Dessa forma, as buscas por similaridade com diversidade conseguem aumentar a qualidade das informações retornadas pelas buscas.

Por ser um processo mais exploratório (mais elementos tendem a ser analisados), as buscas por similaridade com diversidade tendem a ter um custo maior que uma busca por similaridade convencional. Isso se deve, principalmente, ao fato de mais elementos terem de ser carregados/comparados e à necessidade de se comparar não apenas cada elemento ao centro de consulta, mas também entre si. Além disso, alguns modelos de diversidade consideram o problema de diversificação como um problema de otimização que tem complexidade NP-Difícil, onde se tenta encontrar um conjunto de elementos que maximize a similaridade em relação ao elemento central de consulta e a diversidade entre os elementos escolhidos. Por conta disso, muitos algoritmos aproximados e/ou reformulações do problema foram desenvolvidas (SANTOS *et al.*, 2013; VIEIRA *et al.*, 2011; SKOPAL *et al.*, 2009; CARBONELL; GOLDSTEIN, 1998). Entretanto, mesmo considerando essas abordagens, os algoritmos podem ser ineficientes quando aplicados em grandes conjuntos de dados. Com isso, o tempo necessário para gerar uma resposta diversificada pode ser muito alto, o que pode reduzir a aplicabilidade dos modelos de diversificação. Uma abordagem bastante utilizada para tentar reduzir o tempo de execução dos algoritmos de diversificação é a de selecionar um sub-conjunto menor de elementos candidatos da base de dados que é então entregue para o algoritmo de diversificação (NOVAES *et al.*, 2021; VIEIRA *et al.*, 2011). Contudo, algumas das abordagens de seleção de candidatos podem apresentar problemas de escalabilidade e, em alguns casos, a qualidade das respostas tende a ser reduzida (WANG; MELIOU; MIKLAU, 2018; SANTOS *et al.*, 2015).

O principal objetivo de uma busca por similaridade com diversidade é aumentar a

qualidade dos resultados gerados e por conta disso, uma etapa importante do processo de diversificação é a de metrificar/calcular a qualidade do resultados. Neste trabalho, assim como em outros trabalhos da literatura (NOVAES *et al.*, 2021; ZHENG *et al.*, 2017; VIEIRA *et al.*, 2011), a qualidade dos resultados será calculada utilizando uma função objetivo de diversidade que mede o quão similares, em relação ao elemento central de consulta, e diversos entre si os elementos recuperados são.

Visando melhorar o tempo de execução e a qualidade das respostas geradas pelas abordagens atuais de seleção de candidatos, o objetivo principal dessa dissertação é desenvolver novas abordagens de seleção de candidatos que sejam rápidas, escaláveis e que consigam retornar elementos candidatos que possuem qualidade equivalente às demais abordagens da literatura.

1.2 Objetivo Geral

Objetivo geral dessa dissertação é definir abordagens de seleção de candidatos que possam ser utilizadas para reduzir os custos dos algoritmos de seleção por similaridade com diversidade baseados em otimização. Para isso foram desenvolvidas abordagens baseadas em métodos de acesso métrico, particionamento de dados e em algoritmos de diversificação de resultados. Neste sentido, as abordagens foram projetadas para extraírem, de um conjunto de partições da base de dados, elementos candidatos diversos que estejam dentro de um limiar de dissimilaridade, em relação ao elemento central de consulta. Para particionar a base de dados, foi desenvolvida uma nova estrutura de dados, chamada *Omni-Range Tree* (**ORTree**), baseada na *Omni-Techinique* e na *Range Tree* que além de particionar os dados, permite selecionar de forma eficiente as partições contidas dentro de um limiar de dissimilaridade.

1.3 Organização do Documento

Além deste capítulo introdutório, este trabalho está estruturado da seguinte forma:

- O Capítulo 2 apresenta os conceitos básicos relacionados a recuperação por conteúdo e métodos de acesso métrico.
- O Capítulo 3 apresenta as definições de buscas por similaridade com diversidade e os conceitos relacionados aos principais algoritmos de diversificação de resultados.
- O Capítulo 4 apresenta algumas abordagens que tentam agilizar as buscas por similaridade com diversidade.
- O Capítulo 5 apresenta as abordagens de seleção de candidatos desenvolvidas nessa dissertação, bem como, uma nova estrutura de indexação e particionamento dos dados, que serve como base para algumas abordagens que tentam agilizar as buscas por similaridade com diversidade, utilizando a abordagem de seleção de candidatos.

- O Capítulo 6 apresenta o ambiente de testes utilizado para validar as abordagens desenvolvidas e, descreve os resultados dos experimentos realizados, comparando as abordagens desenvolvidas com as abordagens encontradas na literatura.
- O Capítulo 7 apresenta as conclusões, contribuições e trabalhos futuros dessa dissertação.

CAPÍTULO

2

CONCEITOS BÁSICOS

2.1 Considerações Iniciais

As buscas por similaridade com diversidade têm se mostrado ser uma das melhores abordagens para aumentar a qualidade das buscas por similaridade em grandes bases de dados (*big data*). Uma busca por similaridade com diversidade deve retornar os elementos mais similares ao elemento central de consulta mas que também sejam diversos entre si, baseando-se em uma medida de comparação (NOVAES *et al.*, 2021; DROSOU *et al.*, 2017b; SANTOS, 2017).

Este capítulo apresenta alguns dos aspectos relacionados às buscas por similaridade com diversidade em espaços métricos. A Seção 2.2 apresenta algumas das medidas de similaridade mais comuns. Na Seção 2.4 são apresentadas as principais buscas por similaridade. Na Seção 2.5 são apresentados alguns métodos de acesso métricos. Na Seção 2.6 é apresentada a Range Tree, uma abordagem de particionamento de dados, e as conclusões são apresentadas na Seção 2.7.

2.2 Medidas de Similaridade

Por definição, uma busca por similaridade recupera os elementos de acordo com a sua similaridade em relação a um dado elemento central de consulta (s_q). Uma das etapas mais importantes de um processo de busca por similaridade é calcular o quanto dois elementos são similares. Tipicamente, uma função de distância δ pode ser utilizada para medir o quão similar dois elementos são. Quanto menor é a distância entre os elementos, maior é a similaridade entre eles. Consequentemente, quanto maior é a distância, maior a dissimilaridade entre os elementos (ZEZULA *et al.*, 2010; SOUZA, 2018; AVALHAIS, 2012).

Assim como as propriedades das RI e RO podem ser utilizadas para construir estruturas que permitem recuperar e armazenar os dados de forma mais eficiente, uma função δ que respeite

as propriedades dos espaços métricos pode ser utilizada para que algumas estruturas de indexação sejam utilizadas durante uma busca por similaridade. Desta forma, no lugar das propriedades de RI e RO, as propriedades dos espaços métricos são utilizadas para construir estruturas que agilizam as buscas por similaridade (SANTOS, 2017; SOUZA, 2018; ZEZULA *et al.*, 2010; TRAINA *et al.*, 2007). Um espaço métrico pode ser descrito conforme a Definição 1.

Definição 1. Espaço Métrico $\mathbb{M} =< \mathbb{S}, \delta >$. Dado o domínio de dados $\mathbb{S}$ e uma função de distância δ: $\mathbb{S}$ x $\mathbb{S} \longrightarrow \mathbb{R}^+$, o par $< \mathbb{S}, \delta >$ pode ser chamado de espaço métrico se as seguintes propriedades forem atendidas, para quaisquer elementos $s_i, S_j, s_k \in \mathbb{S}$:

- **Simetria**: $\delta(s_i, s_j) = \delta(s_j, s_i)$.
- **Não-negatividade**: $0 \leq \delta(s_i, s_j) < \infty$.
- **Identidade**: $\delta(s_i, s_i) = 0$.
- **Desigualdade Triangular**: $\delta(s_i, s_k) \leq \delta(s_i, s_j) + \delta(s_j, s_k),\ \forall s_i, s_j, s_k \in \mathbb{S}$.

Mesmo que uma função não satisfaça as propriedades listadas anteriormente, ela ainda pode ser utilizada em diversas aplicações. Contudo, algumas otimizações de busca em espaços métricos, por exemplo os métodos de acesso métrico (Seção 2.5), só podem ser adotadas caso a função atenda as propriedades de simetria, não-negatividade e desigualdade triangular (TRAINA *et al.*, 2007; BEYGELZIMER; KAKADE; LANGFORD, 2006; JR. *et al.*, 2000; CIACCIA; PATELLA; ZEZULA, 1997).

Uma grande variedade de funções de distância podem ser encontradas na literatura (DEZA; DEZA, 2009), sendo algumas delas mais apropriadas para uma situação do que outras. A família de funções Minkowski ou L_p é uma das mais conhecidas para espaços multidimensionais. Dados dois elementos de dimensão d, pode-se definir a família L_p da seguinte forma :

$$L_p(s_i, s_j) = \sum_{k=1}^{d} (|s_{ik} - s_{jk}|^p)^{1/p}. \tag{2.1}$$

Os membros mais conhecidos desta família são:

- City-Block ou Manhattan:$L_1(s_i, s_j) = \sum_{k=1}^{d} |s_{ik} - s_{jk}|$.
- Euclidiana: $L_2(s_i, s_j) = \sum_{k=1}^{d} (|s_{ik} - s_{jk}|^2)^{1/2}$.
- Chebychev: $L_\infty = max_{k=1}^{d} |s_{ik} - s_{jk}|$.

Além da família Minkowski, outras funções de distância bastante conhecidas são: a de Bray-Curtis, Canberra, a distância de Jaccard para espaços de conjuntos e a função de edição para espaços de cadeias de texto de Levenshtein (NOVAES; et. al, 2019; BÊDO, 2017; ZHENG *et al.*, 2017).

2.3 Extração de Características

Nem sempre é possível aplicar uma função de distância diretamente sobre os dados complexos. Por exemplo, não é possível aplicar uma métrica L_p sobre duas imagens. Neste sentindo, é comum converter ou extrair dos dados um conjunto de atributos em que se possa aplicar a métrica desejada. Tipicamente, é utilizado um processo de extração de características para extrair dos elementos complexos as informações necessárias. O resultado esperado deste processo é um arranjo, chamado de vetor de características, que representa as informações extraídas. Uma descrição do que é um extrator de características é dada pela definição 2 (BÊDO, 2017; SOUZA, 2018; NOVAES; BENEDITO; SANTOS, 2019; CAZZOLATO, 2019).

Definição 2. Extrator de Características.

Seja $\mathbb{S}$ um domínio de dados e $\mathbb{V}$ um domínio de vetores de características. Um extrator de características $\mathscr{E}$ pode ser modelado como uma função $\mathscr{E} : \mathbb{S} \Rightarrow \mathbb{V}$, ou seja, dado um elemento $s_i \in \mathbb{S}$, a função $\mathscr{E}(s_i)$ gera um elemento $v_i \in \mathbb{V}$. Uma função $\mathscr{E}$ pode ser definida como uma função espelho, neste caso, $s_i = \mathscr{E}(s_i)$.

Assim como as funções de distância, os extratores de características são desenvolvidos para um objetivo. Por exemplo, para imagens existem diferentes extratores baseados nas características de cor, textura ou forma, cada um deles tendo como objetivo coletar um tipo de informação (CAZZOLATO, 2019; MARTINEZ; KOENEN; PEREIRA, 2002). Portanto, a escolha da função de distância e do extrator de características deve ser feita levando em consideração a aplicação, pois essa escolha implica diretamente na qualidade dos resultados gerados (NOVAES; et. al, 2019). O par função de distância e extrator de características tem sido frequentemente dado o nome de "descritor" (Definição 3). Neste trabalho, sempre que um elemento $s_i \in \mathbb{S}$ for referenciado, será considerado que um processo de extração de características foi previamente utilizado e, para as comparações por similaridade, será considerado o uso de um descritor.

Definição 3. Descritor $\mathbb{D} =< \mathbb{S}, \delta > (s_i, s_j)$. Dada uma função de distância δ, um extrator de características $\mathscr{E}$ e dois elementos s_i e $s_j \in \mathbb{S}$, podemos definir o par $< \delta, \mathscr{E} >$ como um descritor d, tal que:

$$d(s_i, s_j) = \delta(\mathscr{E}(s_i), \mathscr{E}(s_j)) \tag{2.2}$$

2.4 Buscas por Similaridade

Um operador de busca por similaridade recupera os elementos da base de dados que atendem a algum critério de comparação por similaridade previamente definido, sendo que os elementos recuperados dependem do elemento central de consulta ($s_q \in \mathbb{S}$). Existem dois critérios fundamentais de busca por similaridade: os que expressam as buscas por abrangência (*Rq* do

inglês - *Range Query*) e os que expressam buscas aos k-vizinhos mais próximos (k-NNq do inglês - *k-nearest neighbor query*).

2.4.1 Busca por Abrangência

Uma busca por abrangência (Rq) recupera todos os elementos da base de dados que estejam a até um limiar de distância (ξ) do elemento central de consulta (s_q). Uma busca por abrangência pode ser definida da seguinte forma (SANTOS, 2017):

Definição 4. Busca por Abrangência - $Rq(s_q,\xi)$. Dado um espaço métrico $\mathbb{M} =< \mathbb{S}, \delta >$, um conjunto de dados $S \subseteq \mathbb{S}$, um elemento central de consulta $s_q \in \mathbb{S}$ e um limiar de distância $\xi \in \mathbb{R}^*$, uma busca por abrangência retorna todos os elementos $s_j \in S$, tal que, $\delta(s_q, s_j) \leq \xi$, ou seja, todos elemento em S cuja distância em relação a s_q seja menor ou igual que ξ.

$$R = \{s_i | \forall s_j \in S : \delta(s_q, s_j) \leq \xi\} \tag{2.3}$$

A Figura 2(a) ilustra um busca por abrangência em um espaço Euclidiano bi-dimensional. No caso da distância euclidiana, o limiar ξ define o raio de uma circunferência onde s_q é o centro, e neste caso, todos os elementos que estão dentro da circunferência são retornados. Um exemplo prático deste tipo de busca seria *"Retorne todas as imagens que difiram desta imagem de consulta (s_q) em até 10 unidades"*, sendo que a unidade de distância depende da função empregada.

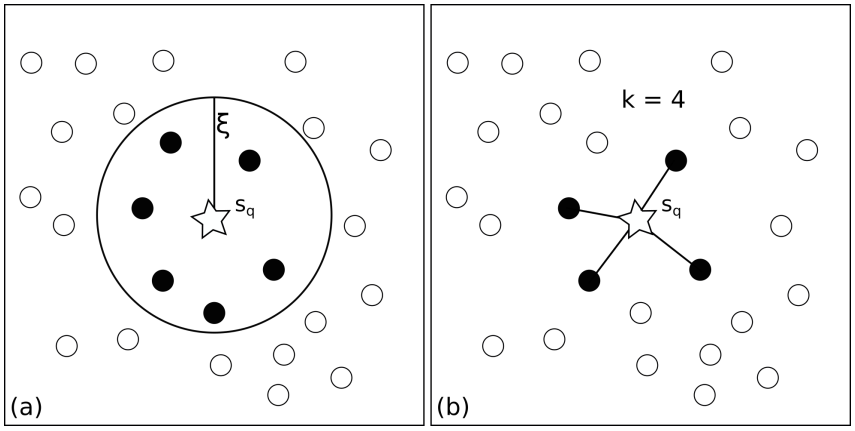
O elemento central de consulta pode ou não fazer parte da base de dados ($s_q \in S$), e cada caso é uma variante da busca por abrangência. Para verificar se o elemento s_q faz parte ou não da base de dados, uma busca por abrangência com $\xi = 0$ pode ser executada. Tipicamente, essa busca recebe o nome de busca pontual (*pontual query*) e pode ser utilizada para outros fins, como coletar estatísticas sobre a base de dados. Um outro tipo de busca derivada da busca por abrangência é a abrangência reversa. Neste caso, são retornados todos os elementos cuja a distância em relação a s_q seja maior que ξ: $R = \{\forall s_j \in S : \delta(s_q, s_j) > \xi\}$ (BÊDO, 2017; SANTOS, 2017).

2.4.2 Busca aos k-Vizinhos mais Próximos

Uma busca aos k-vizinhos mais próximos (k-NNq) recupera os k elementos da base de dados que estão mais próximos do elemento de central de consulta (s_q). A definição desta busca é apresentada a seguir (SANTOS, 2017):

Definição 5. Busca aos k-Vizinhos mais Próximos - k-$NNq(s_q$,k). Dado um espaço métrico $\mathbb{M} =< \mathbb{S}, \delta >$, um conjunto de elementos $S \subseteq \mathbb{S}$, um elemento central de consulta $s_q \in \mathbb{S}$ e o

Figura 2 – Exemplos de buscas por similaridade em um espaço euclidiano bi-dimensional. (a) Busca por abrangência. (b) Buscas aos 4-vizinhos mais próximos.



Fonte: Elaborada pelo autor.

número de elementos a serem retornados $k \in \mathbb{N}$, uma busca aos k-vizinhos mais próximos é dada pela equação:

$$R = \{s_j \in S, \forall s_i \in S - R : \delta(s_q, s_j) \leq \delta(s_q, s_i), |R| = k\} \tag{2.4}$$

A Figura 2(b) representa um busca aos k-vizinhos mais próximos, para $k = 4$, em um espaço Euclidiano bi-dimensional. De maneira intuitiva, essa busca tende a retornar os k elementos mais próximos ao elemento central de consulta. Um exemplo prático deste tipo de busca seria: *"Retorne as 5 imagens mais próximas desta imagem de consulta"*.

Assim como a busca por abrangência, existem variantes da busca aos k-vizinhos mais próximos, por exemplo, a busca aos k-vizinhos mais distantes, que retorna os k elementos mais distantes ao elemento central de consulta, e a busca aos k-vizinhos reversa, a qual retorna, todos os elementos que têm o elemento s_q como um dos seus k-vizinhos mais próximos (SANTOS, 2017; BÊDO, 2017).

Um ponto interessante sobre a busca k-NNq é que ela poder ser vista como uma variante da Rq. Neste caso, pode-se definir um raio ξ_k, tal que, o resultado de uma busca $Rq(s_q, \xi_k)$ seja equivalente ao de uma k-NNq (BÊDO, 2017; DIAS; BUENO; RIBEIRO, 2013; VIEIRA *et al.*, 2007).

2.5 Método de Acesso Métrico

Os resultados das buscas discutidas até aqui podem ser obtidos por um processo de busca sequencial sobre todos os elementos da base de dados, ou seja, comparando-se todos os elementos da base de dados com o elemento central de consulta. Contudo, em alguns casos, principalmente em *big data*, realizar todos os cálculos de distância necessários pode tornar o processo de busca

bastante lento, devido ao alto custo computacional envolvido. Além disso, pode não ser possível manter/carregar todos os elementos da base de dados em memória principal, sendo necessário salvar os dados em disco e carregar os elementos a serem comparados em memória, o que acarreta em realizar muitos acessos a disco (SANTOS, 2017; TRAINA *et al.*, 2007; JR. *et al.*, 2000). Nos SGBDs tradicionais, existem estruturas (chamadas de índices) baseadas nas relações de identidade e de ordem que permitem armazenar e recuperar os elementos de forma eficiente. De forma análoga, foram propostas estruturas que permitem armazenar e recuperar os elementos baseando-se na similaridade entre eles. Algumas delas indexam os elementos utilizando um modelo espacial definido pelos vetores de características (Métodos de Acesso Espaciais - MAE) enquanto outras utilizam as relações de distância entre os vetores características e as propriedades dos espaços métricos (Métodos de Acesso Métrico - MAM). Como as relações de distância podem ser utilizadas para expressar a similaridade entre os elementos, as técnicas baseadas em distância se mostram mais adequadas para realizar as buscas por similaridade (SOUZA, 2018; TRAINA *et al.*, 2007).

Os MAM são estruturas que permitem indexar os elementos, desde que eles estejam representados em algum espaço métrico. Essas estruturas são projetadas para utilizar as propriedades dos espaços métricos, como a desigualdade triangular e a simetria, para tornar o acesso aos dados mais eficiente, bem como reduzir o número de comparações. Na literatura são encontradas várias implementações de MAM, tais como: *M-tree* (CIACCIA; PATELLA; ZEZULA, 1997), *Omni-Technique* (TRAINA *et al.*, 2007), *Slim-tree* (JR. *et al.*, 2000) e VP-*tree* (YIANILOS, 1993). Uma revisão dos principais métodos de indexação para buscas por similaridade pode ser encontrada em (ZEZULA *et al.*, 2006). As abordagens desenvolvidas neste trabalho são baseadas na *Omni-Technique* e suas propriedades e, por conta disso, ela é mais discutida a seguir.

2.5.1 Omni-Technique

A *Omni-Technique* permite criar MAMs baseando-se em pivôs globais. A abordagem utilizada seleciona um conjunto de elementos da base de dados para serem pivôs, e então, pré-computar a distância de todos os elementos da base de dados para os pivôs. A abordagem *Omni* se refere ao conjunto de distâncias entre um elemento s_i e os pivôs, como as *omni*-coordenadas de s_i. Depois de geradas, as *omni*-coordenadas são armazenadas utilizando alguma estrutura secundária. Neste caso, a técnica pode ser dividida em dois níveis: um que gera as *omni*-coordenadas, e outro que armazena os elementos e suas respectivas coordenadas numa estrutura de dados secundária.

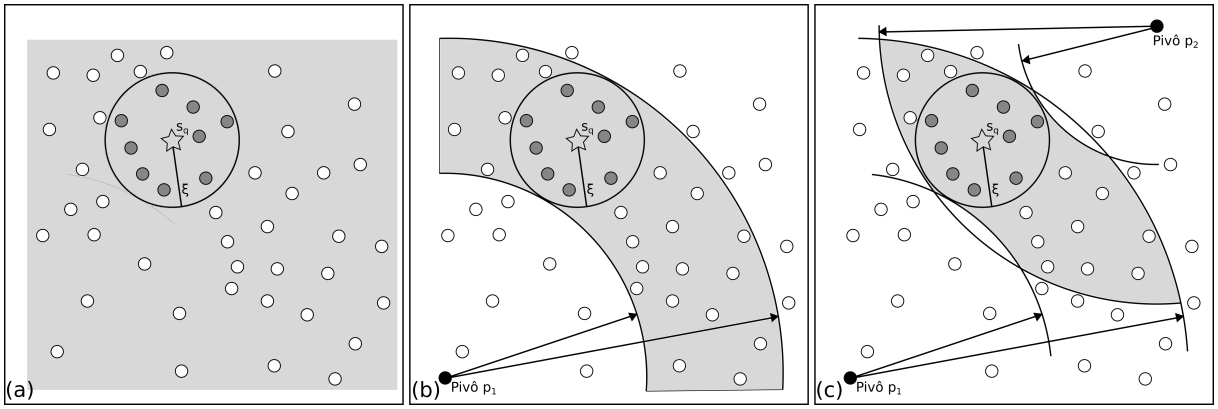
As diferentes estruturas secundárias utilizadas criam diferentes variações da *Omni-Technique*, que constituem a família *Omni* de estruturas. Por exemplo, pode-se utilizar uma *B-tree*, *R-tree*, *Slim-tree* ou um *Random Access File* (RAF) e cada uma delas gera, respectivamente, os membros *Omni - B-forest*, *Omni - R-tree*, *Omni - Slim-tree* e *Omni - Sequential*.

Tipicamente, o processo de busca utilizando a *Omni-Technique* pode ser dividido em duas etapas: filtragem e refinamento. Durante a etapa de filtragem são geradas as *omni*-coordenadas do

elemento central de consulta e a partir delas, utilizando a propriedade de desigualdade triangular, é definida uma região de interesse que contém os elementos que devem ser retornados pela busca que contém todos os elementos da resposta e também, alguns falso-positivos. Em seguida na etapa de refinamento a busca é executada na estrutura secundaria, que remove os falsos-positivos e gera a resposta final.

A Figura 3 ilustra uma busca por abrangência, em um espaço euclidiano bi-dimensional com a métrica L_2 utilizando a técnica *Omni-Sequential*. Na Figura 3(a), a busca é realizada sem utilizar nenhum pivô. Por conta disso, todos elementos precisam ser comparados com s_q. Na Figura 3(b), é utilizado um pivô. Neste caso, pela propriedade de desigualdade triangular, somente os elementos que estão dentro do hiper-anel definido pelo pivô p_1 são comparados a s_q. Já a Figura 3(c), ilustra uma poda ainda mais eficiente ao utilizar mais de um pivô, e somente os elementos dentro dos hiper-anéis definidos pelos dois pivôs é que são comparados a s_q. A região definida pelo hiper-anéis é chamada de região de fronteira mínima (mbOr do inglês - *minimal bounding Omni region*).

Figura 3 – Uma busca por abrangência com centro s_q e raio ξ usando a métrica L_2. A área em cinza ilustra a região com os elementos que foram comparados a s_q.(a) Sem a utilização de nenhum pivô, todos os elementos precisam ser comparados. (b) Com um pivô, somente os elementos do hiper-anel são os comparados. (c) Com dois pivôs, somente os elementos que estão contidos nos dois hiper-anéis são comparados.



Fonte: Adaptada de Traina *et al.* (2007).

2.5.2 Estrategias de Busca Utilizando MAM

O processo de busca por abrangência utilizando um MAM pode ser realizado de forma direta. Para isso basta verificar quais elementos estão dentro da região definida por $< s_q, \xi >$ especificado. Já em uma busca aos k-vizinhos mais próximos, não se sabe a priori qual é o raio que contém os k elementos solicitados, e por isso, algumas estratégias devem ser empregadas. Por exemplo, pode-se utilizar uma estratégia do tipo *branch-and-bound*. O raio da busca é inicialmente definido como infinito ($\xi = \infty$) e ele vai sendo dinamicamente reduzido até que os k elementos mais próximos a s_q sejam encontrados. Essa redução dinâmica é feita utilizando

uma lista que mantém os k elementos candidatos ordenados pela distância a s_q. Sempre que um elemento com distância menor que o k-ésimo for encontrado, a lista é atualizada.

Uma segunda estratégia para realizar uma busca k-NN é fazer uma estimativa do raio (ξ_k) que contém os k elementos desejados. Isso pode ser feito utilizando alguns conceitos da teoria dos fractais (BÊDO, 2017; DIAS; BUENO; RIBEIRO, 2013; VIEIRA *et al.*, 2007; FALOUTSOS *et al.*, 2000).

Alguns trabalhos mostram que a distribuição de distâncias entre elementos na maioria dos conjuntos de dados reais segue uma distribuição fractal (ou auto-similaridade). Para esses conjuntos de dados, a distribuição de distâncias entre os elementos segue alguma lei de potência, por exemplo: Dado um conjunto de N elementos e uma função de distância δ, o número (k') médio de elementos em uma determinada distância ξ, é proporcional a $\xi^{\mathfrak{D}}$, onde $\mathfrak{D}$ é a dimensão fractal do conjunto de dados. A partir disso, a quantidade de pares de elementos com distância até ξ pode ser definida pela seguinte Equação (VIEIRA *et al.*, 2007; FALOUTSOS *et al.*, 2000):

$$PC(\xi) = K_p \cdot \xi^{\mathfrak{D}} \tag{2.5}$$

Onde K_p é uma constante de proporcionalidade.

Para dados que atendam a essa propriedade, a partir da Equação 2.5 é possível estimar um raio ξ_k que contém k elementos utilizando as Equações 2.6 e 2.7 (VIEIRA *et al.*, 2007). A Equação 2.6 permite estimar um raio ξ' que contém aproximadamente os k elementos solicitados. Entretanto, como a densidade de dados no espaço varia dependendo da região do espaço considerada, essa estimativa pode ser subestimada, e a quantidade de elementos retornados pode ser menor que k. Para melhorar essa estimativa, pode-se utilizar novamente a Equação 2.6, mas agora o raio ξ_k é estimado utilizando os resultados obtidos pela primeira estimativa. Mais especificamente, utilizando o raio ξ' e a quantidade de elementos (k') retornados pela estimativa inicial, pode-se ajustar a estimativa do raio à distribuição dos elementos no entorno do elemento central de consulta (Equação 2.7) (BÊDO, 2017; VIEIRA *et al.*, 2007).

$$\xi' = R \cdot exp\left(\frac{log(k-1) - log(N-1)}{\mathfrak{D}}\right) \tag{2.6}$$

Aqui R é a maior distância entre os elementos (ou o diâmetro) do conjunto de dados.

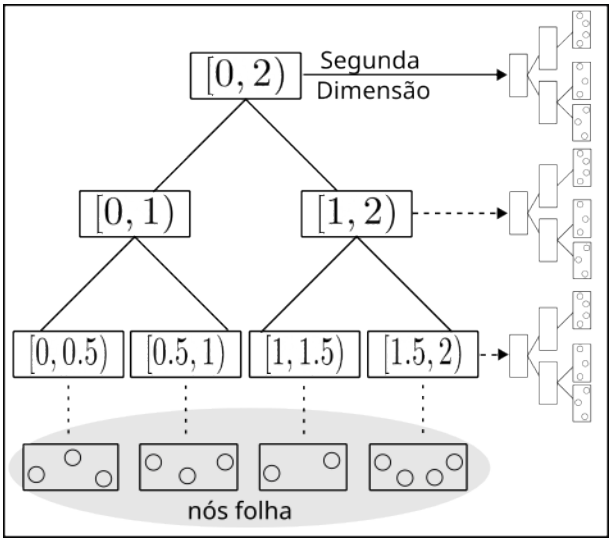
$$\xi_k = \xi' \cdot exp\left(\frac{log(k-1) - log(k'-1)}{\mathfrak{D}}\right) \tag{2.7}$$

2.6 Range Tree

A Range Tree (RT) (WANG; MELIOU; MIKLAU, 2018; BERG *et al.*, 1997) é uma estrutura de dados que tem como objetivo encontrar de forma rápida todos os elementos que estão

contidos dentro de um determinado intervalo de busca. Essa estrutura organiza os elementos da base dados, particionando os elementos considerando os seus vetores de características. A Figura 4 ilustra como a RT organiza um conjunto bi-dimensional onde o intervalo da primeira dimensão dos vetores de características vai de 0 até 2. Os limites (inferior e superior) do nó raiz são iguais ao intervalo da primeira dimensão, logo, o nó raiz cobre todos os elementos da base de dados. A partir disso, todos os nós abaixo têm como limites um subintervalo do intervalo do nó raiz. Neste caso, os filhos do nó raiz geram uma partição dos elementos da base de dados, considerando a mediana das características dos elementos contidos no nó raiz. Esse processo de particionamento dos elementos é repetido, recursivamente, para os filhos do nó raiz até que os nós contenham um único elemento, o que define os nós folhas da árvore. Para armazenar elementos com mais de uma dimensão, cada nó possui um filho('next') que aponta para uma RT que organiza os elementos do nó considerando a próxima dimensão.

Figura 4 – Exemplo de uma Range Tree bi-dimensional.



Fonte: Elaborada pelo autor.

.

Para realizar uma busca é preciso definir um intervalo de busca (Rd_r), que descreve o intervalo de busca de cada uma das d dimensões dos elementos. Um exemplo de intervalo de busca para um espaço bi-dimensional pode ser dado da seguinte forma: $Rd_r = \{x_{inicial}, x_{final}, y_{inicial}, y_{final}\}$. Vale a pena ressaltar que esse intervalo de busca é uma sequência de valores para cada dimensão, o que é completamente diferente do limiar de distância (ξ) utilizado pela busca por abrangência. A partir do intervalo de busca, a busca na RT é executada para encontrar os nós da primeira dimensão que estão dentro de Rd_r. Quando um nó na primeira dimensão que está dentro de Rd_r é encontrado, o seu filho 'next', se existir, é utilizado para continuar a busca na próxima dimensão. Esse processo se repete até que todas as dimensões tenham sido investigadas e os nós contidos dentro de Rd_r sejam encontrados.

A complexidade de tempo de busca da RT é $O(\log^d(n))$ e a complexidade de espaço é $O(n\log^d(n))$, onde d é dimensionalidade da base de dados e n a sua cardinalidade. Como ambas

complexidades dependem da dimensionalidade da base de dados, a RT, apresenta problemas ao ser utilizada em base de dados com alta dimensionalidade (WANG; MELIOU; MIKLAU, 2018). Além disso, ela não foi projetada para responder a buscas por similaridade, pois ela só consegue expressar buscas por intervalo, o que não é equivalente às buscas *Rq* e *k-NNq*. Contudo, neste trabalho a RT será adaptada e utilizada como base para as abordagens desenvolvidos nessa dissertação.

2.7 Considerações Finais

Este capítulo abordou os conceitos básicos relacionados às buscas por similaridade e métodos de acesso métricos. As buscas por similaridade tem se mostrado uma das melhores formas de se recuperar dados complexos, contudo, ao serem aplicadas em grandes conjuntos de dados elas tem apresentados problemas relacionados a perda de expressividade o que reduz a qualidade dos resultados gerados pelas buscas.

No próximo capítulo são apresentadas as buscas por similaridade com diversidade, um conceito que permite contornar os problemas de perda de expressividade das buscas por similaridade convencionais.

CAPÍTULO

3

MODELOS DE DIVERSIDADE

3.1 Considerações Iniciais

As buscas por similaridade têm se tornado uma das principais formas de se recuperar dados complexos. Contudo, ao serem aplicadas em conjuntos de dados muito densos e/ou volumosos, a qualidade das respostas tende a reduzir. Conforme a densidade e o volume dos dados crescem, a quantidade de elementos que são muito similares entre si também aumenta. Assim, os operadores fundamentais de seleção por similaridade (k-NNq e Rq) têm sua expressividade reduzida. Neste cenário, ao realizarmos uma busca por similaridade, é muito provável que os elementos retornados serão similares ao elemento central de consulta, mas também serão muito similares entre si. Em alguns casos, podem ser até mais similares entre si do que em relação ao elemento central de consulta. Logo, boa parte dos elementos podem apresentar as mesmas informações, o que pode não agregar valor para o usuário, reduzindo a qualidade dos resultados (NOVAES; et. al, 2019; ZHENG *et al.*, 2017; SANTOS; et. al, 2018; DROSOU *et al.*, 2017a; SKOPAL *et al.*, 2009). Para solucionar esse problema, várias pesquisas têm considerado a inclusão do conceito de diversidade às buscas por similaridade (SANTOS, 2017; ZHENG *et al.*, 2017; VIEIRA *et al.*, 2011; GOLLAPUDI; SHARMA, 2009; SKOPAL *et al.*, 2009; HARITSA, 2009). Neste sentido, pode se dizer que as buscas por similaridade convencionais foram projetadas para recuperar os elementos considerando apenas a relação de similaridade entre os elementos da base de dados e o elemento central de consulta s_q. Enquanto que uma busca por similaridade com diversidade também considera a relação de similaridade entre os elementos da base de dados e o elemento s_q, mas também considera a relação de similaridade entre os elementos retornados pela busca, com o objetivo de evitar que elementos muito similares entre si sejam retornados.

Este capítulo apresenta o conceito de busca por similaridade com diversidade e as principais abordagens de diversificação de buscas por similaridade que podem ser utilizadas em espaços métricos. Os principais conceitos das buscas por similaridade com diversidade são apresentadas

na Seção 3.2. O foco principal deste trabalho está nas abordagens de diversificação baseadas em novidade (ou otimização), que são apresentadas na Seção 3.3. Contudo, algumas abordagens desenvolvidas nessa dissertação, bem como algumas abordagens da literatura, utilizam algoritmos de diversificação baseada em cobertura. Por conta disso, na Seção 3.4 são apresentados os principais algoritmos baseados em cobertura. Por fim, na Seção 3.5 são apresentadas as conclusões deste capítulo.

3.2 Busca por Similaridade com Diversidade

O conceito de diversificação de resultados teve sua origem na área de recuperação de informação. Intuitivamente, considera-se que as respostas geradas pelas buscas não devem possuir apenas alta similaridade ao elemento central de consulta, mas também um certo grau de diversidade entre os elementos da resposta, evitando que os elementos da resposta sejam muito similares entre si (ZHENG *et al.*, 2017; DROSOU *et al.*, 2017a; GOLLAPUDI; SHARMA, 2009).

Na literatura podem ser encontradas diferentes estratégias para a diversificação de resultados, por exemplo, as baseadas em atributos e as baseadas em conteúdo. A estratégia baseada em atributos consiste em comparar o quanto cada atributo da descrição dos dados difere dos atributos dos elementos já inseridos no resultado (ZHENG *et al.*, 2017; AGRAWAL *et al.*, 2009). Já a diversificação baseada em conteúdo, consiste em comparar o quão similar é a descrição dos dados (vetor características): em outras palavras, o quão próximo dois elementos são (ZHENG *et al.*, 2017; SANTOS *et al.*, 2013; VIEIRA *et al.*, 2011; SKOPAL *et al.*, 2009). Neste trabalho iremos abordar a estratégia de diversificação baseada em conteúdo, principalmente, por ser a mais indicada para se trabalhar com dados complexos (SANTOS, 2017; ZHENG *et al.*, 2017; VIEIRA *et al.*, 2007).

Foram propostas diferentes abordagens para incluir diversidade às buscas por similaridade. Tipicamente, elas são classificadas em baseadas em cobertura ou novidade, também chamadas de baseadas em similaridade/distância (NOVAES *et al.*, 2021; LOPES *et al.*, 2021; ZHENG *et al.*, 2017). A primeira retorna os elementos considerando que deve existir um limiar de dissimilaridade entre os elementos retornados. Nessa abordagem, o objetivo é encontrar um conjunto de elementos que cubram diferentes informações. O segundo tipo de abordagem, busca encontrar elementos que maximizem uma função objetivo de critério duplo, onde similaridade e diversidade são balanceadas conforme a preferência do usuário. O objetivo dessa abordagem é tentar encontrar elementos que maximizem a 'novidade' de informações dentro do conjunto resposta (DROSOU *et al.*, 2017b; VIEIRA *et al.*, 2011; ZHENG *et al.*, 2017). Na literatura não há uma definição de qual dessas abordagens é a melhor, sendo que essa decisão depende do domínio dos dados e dos objetivos da busca (NOVAES *et al.*, 2021; LOPES *et al.*, 2021). Neste trabalho o foco estará nas abordagens baseadas em novidade. Contudo ambas abordagens são

discutidas nas próximas seções.

3.3 Diversificação de Resultados baseada em Novidade

Tipicamente, os algoritmos de busca por similaridade com diversidade baseados em novidade retornam os k elementos mais próximos ao elemento central de consulta (s_q), utilizando uma medida de similaridade e outra de diversidade: $Sim(s_q,R)$ calcula o quão similares os elementos são em relação ao elemento central de consulta (Definição 6) e $Div(R)$ calcula o quão diverso são os elementos do conjunto resposta (Definição 7). De modo mais formal, o problema da diversificação se resume em encontrar um conjunto resposta $R : R \subseteq S$, de cardinalidade k, em que cada elemento de R é tanto similar ao elemento central de busca, como diverso em relação aos outros elementos em R. A partir disso, o problema de diversificação pode ser modelado como um problema de otimização de critério duplo, onde deve-se encontrar um conjunto R que maximize similaridade e diversidade (SANTOS, 2017; DROSOU *et al.*, 2017a; VIEIRA *et al.*, 2011; GOLLAPUDI; SHARMA, 2009; CARBONELL; GOLDSTEIN, 1998).

Definição 6. ($Sim(s_q,R)$). Seja $\mathbb{S}$ um domínio de dados, $S \subseteq \mathbb{S}$ o conjunto de elementos da base de dados, δ_{sim} uma função que avalia a similaridade entre os elementos, s_q um elemento central de consulta e R um conjunto reposta de cardinalidade k. A similaridade do conjunto R é dada pela equação:

$$Sim(s_q,R) = \sum_{i=0}^{k} \delta_{sim}(s_q,s_i) \tag{3.1}$$

Definição 7. ($Div(R)$) Seja $\mathbb{S}$ um domínio de dados, $S \subseteq \mathbb{S}$ o conjunto conjunto de elementos da base de dados, δ_{div} uma função de distância que avalia a diversidade entre dois elementos e R um conjunto reposta de cardinalidade k. A diversidade do conjunto R é dada pela equação:

$$Div(R) = \sum_{i=0}^{k-1} \sum_{j=i+1}^{k} \delta_{div}(s_i,s_j) \tag{3.2}$$

A equação 3.3 é um exemplo de função de critério duplo, que permite maximizar a similaridade e a diversidade.

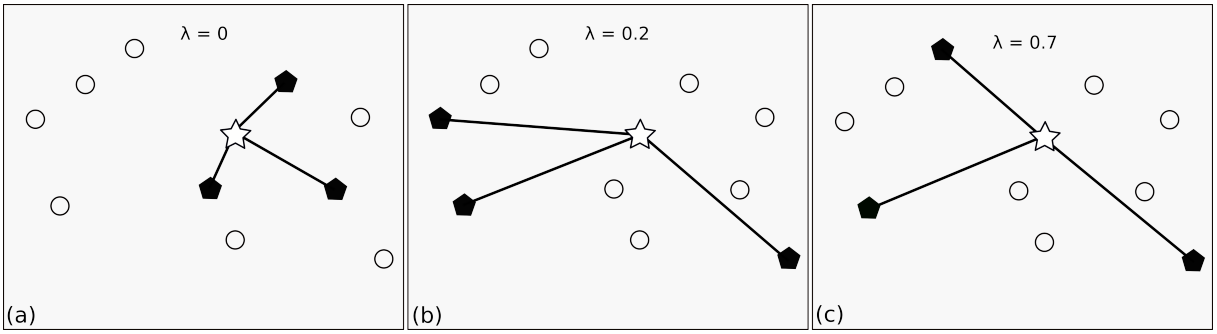
$$\mathscr{F}(s_q,R) = (k-1)(1-\lambda)Sim(s_q,R) + 2\lambda Div(R). \tag{3.3}$$

O parâmetro $\lambda[0,1]$ representa a preferência de diversidade do usuário. Se $\lambda = 0$, o problema se resume a uma busca k-NNq convencional. Já para $\lambda > 0$, o problema se torna NP-Difícil, pois se trata de um problema de análise combinatória conhecido como problema

da diversidade máxima (*Maximum Diversity Problem* — MDP) (WANG; MELIOU; MIKLAU, 2018; DROSOU *et al.*, 2017a; ZHENG *et al.*, 2017). Se o espaço de busca S tem cardinalidade n ($|S| = n$), a quantidade de subconjuntos de tamanho k é $C(n,k) = \frac{n!}{(n-k)!}$, o que resulta em complexidade de tempo $O(n^k)$, tornando pouco provável gerar respostas exatas em tempo hábil (DROSOU *et al.*, 2017a; SANTOS *et al.*, 2015; VIEIRA *et al.*, 2011). Além disso, avaliar cada possível solução para o problema requer $O(k^2)$ cálculos de distância, logo, o algoritmo exato tem custo de tempo $O(n^k * k^2)$ (VIEIRA *et al.*, 2011).

A Figura 5 ilustra como o parâmetro λ modifica o processo de diversificação modelado como um problema de otimização.

Figura 5 – Resultado de uma busca por similaridade com diversidade baseada em novidade, para $k = 3$. (a) Preferência de diversidade $\lambda = 0.0$, o resultado da busca é equivalente a uma k-NNq. (b) Preferência de diversidade $\lambda = 0.2$, os elementos retornados começam a se afastar de s_q e um dos outros. (c) Preferência de diversidade $\lambda = 0.7$, neste caso, os elementos retornados estão o mais distante possível um dos outros (máxima diversidade), mas a distância em relação a s_q também aumenta bastante.

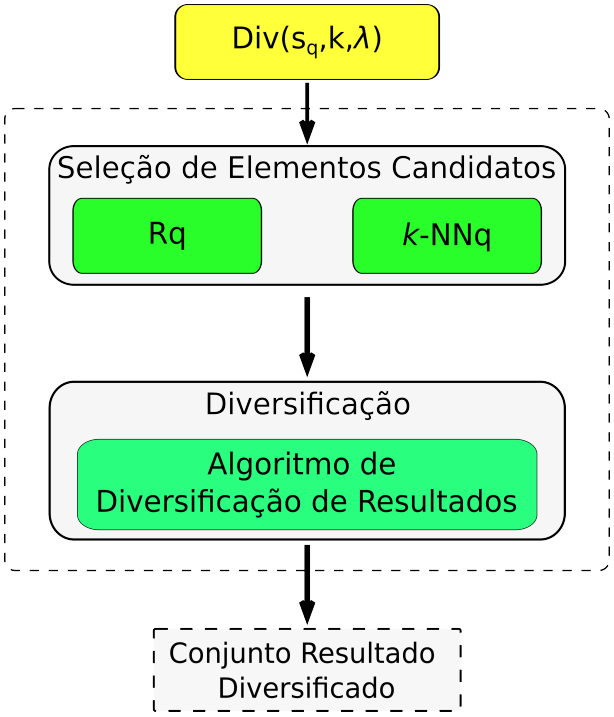


Fonte: Elaborada pelo autor.

Uma estratégia bastante utilizada para reduzir os custos desse tipo de busca, é a de reduzir o espaço de busca. Nessa estratégia elementos que são muito dissimilares(distantes) ao elemento central de consulta (s_q) são desconsiderados. O objetivo dessa estratégia é selecionar um conjunto que contém os $m(m > k)$ elementos mais próximos do elemento s_q. Neste caso, a estratégia não resolve o problema de complexidade, só o reduz, pois o número de possíveis combinações é reduzido de $C(|S|,k)$ para $C(m,k)$, o que é verdade para qualquer $m < |S|$. Por conta disso, muitos trabalhos combinam essa estratégia com algoritmos(gulosos) que tendem a gerar respostas determinísticas e aproximadas, com custo de tempo bem menor que o algoritmo ótimo (SANTOS, 2017; DROSOU *et al.*, 2017a; DIAS; BUENO; RIBEIRO, 2013; VIEIRA *et al.*, 2011). Tipicamente, por conta dessas duas estratégias, pode-se dividir o processo de diversificação em duas etapas (Figura 6): Seleção de Candidatos e Diversificação.

A etapa de seleção de candidatos é, geralmente, executada por uma busca por similaridade que retorna os m elementos mais próximos(similares) a s_q. Já na etapa de diversificação, um algoritmo de diversidade é aplicado sobre o conjunto candidato a fim de extrair os k elementos que

Figura 6 – Processo de diversificação de resultados, baseada em novidade/otimização.



maximizam a função objetivo (WANG; MELIOU; MIKLAU, 2018; SANTOS, 2017; ABBAR *et al.*, 2013; VIEIRA *et al.*, 2011). O número de candidatos m precisa ser definido de forma correta, pois a quantidade de elementos candidatos não impacta somente no tempo de execução dos algoritmos. Se m for muito pequeno, os elementos candidatos serão todos bastante similares a s_q e, provavelmente, muito similares entre si. Caso m seja muito grande, o tempo de execução dos algoritmos de diversificação pode se tornar impraticável.

Os algoritmos gulosos utilizados, durante a etapa de diversificação, podem ser classificados pela sua estratégia de construção do resultado final, podendo ser: incremental, baseada em meta-heurísticas ou baseada em troca. A estratégia incremental começa com o conjunto vazio, e iterativamente escolhe no conjunto candidato um elemento que maximiza a função objetivo. A estratégia baseada em meta-heurística utiliza duas fases, sendo a primeira responsável por definir um conjunto inicial e por escolher os elementos que trazem melhor contribuição para a resposta final segundo uma função objetivo; a segunda fase utiliza uma busca local, e troca os elementos para verificar se é possível otimizar o resultado já encontrado. Já a estratégia baseada em troca começa com um conjunto, inicialmente preenchido considerando somente a similaridade ao elemento central de busca. A partir disso, cada elemento restante no conjunto candidato é avaliado para uma troca por outro elemento do conjunto inicial; caso essa troca maximize a função objetivo, o elemento é inserido permanentemente no conjunto (SANTOS, 2017; ZHENG *et al.*, 2017; VIEIRA *et al.*, 2011).

Carbonell e Goldstein (1998) definiram o algoritmo de relevância marginal máxima (*Maximal Marginal Relevance* - MMR) (Algoritmo 1), baseado no conceito de importância

marginal (*marginal relevance*). Um elemento tem maior importância marginal se ele é tão similar ao elemento central de busca quanto dissimilar aos elementos já inseridos no conjunto resposta. O MMR segue uma estratégia gulosa incremental. Inicialmente, o elemento mais similar à s_q é inserido em R, e em seguida, de forma incremental, são escolhidos os $k-1$ elementos que, quando inseridos em R, maximizam a função o objetivo (3.4). No entanto, a qualidade dos resultados gerados pelo MMR depende diretamente da escolha do primeiro elemento. Como ele é sempre o mais similar a s_q, a qualidade dos resultados gerados pelo MMR tende a ser reduzida conforme λ cresce, pois quanto maior λ for, maior vai ser a preferência por diversidade, como isso não acontece na escolha do primeiro elemento o MMR acaba perdendo qualidade (SANTOS, 2017; VIEIRA *et al.*, 2011).

$$F_{MMR}(s_i) = (1-\lambda)\delta_{sim}(s_q, s_i) + 2\lambda \sum_{s_j \in R} \delta_{div}(s_i, s_j) \tag{3.4}$$

Algoritmo 1 – MMR

Entrada: Conjunto de elementos S e a cardinalidade k do conjunto resultado R.
Saída: Conjunto resultado $R \subseteq S$, $|R| = k$.

1: $R \leftarrow \emptyset$
2: $R \leftarrow \{$o elemento mais similar a $s_q\}$
3: **enquanto** $|R| < k$ **faça**
4: $\quad s_t \leftarrow argmax_{s_i \in S}(MMR(s_i))$
5: $\quad R \leftarrow R \cup s_t$
6: $\quad S \leftarrow S - s_t$
7: **fim enquanto**

Gollapudi e Sharma (2009) apresentam um conjunto de axiomas que os sistemas de diversificação de resultados devem atender e estabelecem que nenhuma função objetivo pode satisfazer, simultaneamente, todos os axiomas. Além disso, eles apresentam três novos algoritmos de diversificação, sendo o MSD (*Max-Sum Dispersion*) (Algoritmo 2) o mais utilizado. O algoritmo MSD constrói incrementalmente o conjunto R, selecionando o par de elementos que maximize a função objetivo (3.5). Basicamente, a cada iteração são escolhidos os dois elementos s_i e $s_j \in S$ que são tanto similares a s_q quanto diversos entre si. Para os casos em que k é ímpar, o MSD escolhe aleatoriamente o último elemento a ser inserido. Um problema com o MSD é que ele avalia a diversidade entre os pares de elementos, sendo que no conjunto R já podem ter sido inseridos elementos muito similares à s_i e/ou s_j.

$$F_{MSD}(s_i, s_j) = (1-\lambda)(\delta_{sim}(s_q, s_i), \delta_{sim}(s_q, s_j)) + 2\lambda\,\delta_{div}(s_i, s_j) \tag{3.5}$$

Outra abordagem para gerar o conjunto R diversificado é utilizar algum algoritmo de agrupamento, por exemplo o CLT (*Clustering-Based Method*) (Algoritmo 3) proposto por Leuken *et al.* (2009). Inicialmente, os elementos candidatos são divididos em k grupos utilizando o

Algoritmo 2 – MSD

Entrada: Conjunto de elementos S e a cardinalidade k do conjunto resultado R.
Saída: Conjunto resultado $R \subseteq S$, $|R| = k$.

1: $R \leftarrow \emptyset$
2: **enquanto** $|R| < \lfloor k/2 \rfloor$ **faça**
3: $\{s_i, s_j\} \leftarrow argmax_{s_i,s_j \in S}(F_{MSD}(s_i, s_j))$
4: $R \leftarrow R \cup \{s_i, s_j\}$
5: $S \leftarrow S - \{s_i, s_j\}$
6: **fim enquanto**
7: **se** $k \ mod \ 2 \neq 0$ **então**
8: Seleciona aleatoriamente um elemento $s_i \in S$
9: $R \leftarrow R \cup s_i$

algoritmo *k-medoid*, com a função δ_{div}. Em seguida, um elemento de cada grupo é selecionado e inserido em R. Um dos problemas dessa abordagem é que o agrupamento não leva em conta o parâmetro λ, ou seja, os grupos são gerados levando em consideração somente a diversidade entre os elementos, o que pode ser um problema, principalmente, para valores de λ menores que 0.5 (VIEIRA *et al.*, 2011).

Algoritmo 3 – CLT

Entrada: Conjunto de elementos S e a cardinalidade k do conjunto resultado R.
Saída: Conjunto resultado $R \subseteq S$, $|R| = k$.

1: $R \leftarrow \emptyset$
2: $C = c_1, c_2, c_3, \cdots, c_k$ ▷ Resultado do algoritmo *k-medoid*
3: **para** $\forall c_i \in C$ **faça**
4: selecionar um elemento s_i do grupo c_i.
5: $R \leftarrow R \cup s_i$
6: **fim para**=0

Vieira *et al.* (2011) definiram os algoritmos GMC (*Greedy Marginal Contribution*) e GNE (*Greedy Randomized with Neighborhood Expansion*). O algoritmo GMC segue basicamente os mesmos passos do algoritmo MMR, porém o GMC (Algoritmo 4) utiliza uma função de avaliação diferente da utilizada pelo MMR, chamada de máxima contribuição marginal (*maximum marginal contribution* - MMC) (Equação 3.6). A função MMC avalia a contribuição do elemento $s_i \in S$, considerando a similaridade entre s_i e s_q, a diversidade entre s_i e os elementos já inseridos em R e a diversidade entre s_i e os elementos do conjunto candidato que ainda não foram inseridos na resposta.

$$F_{MMC}(s_i) = (1-\lambda)\delta_{sim}(s_q, s_i) + \frac{\lambda}{k-1} \sum_{s_j \in R_{t-1}} \delta_{div}(s_i, s_j) + \frac{\lambda}{k-1} \sum_{l=1, s_j \in S - s_i}^{l \leq k-t} \delta_{div}^{l}(s_i, s_j) \quad (3.6)$$

A principal diferença entre o GNE e o GMC é a estratégia de construção do conjunto R. O GNE (Algoritmo 5) é baseado na meta-heurística GRASP (*Greedy Randomized Adaptive*

Algoritmo 4 – GMC

Entrada: Conjunto de elementos S e a cardinalidade k do conjunto resultado R.
Saída: Conjunto resultado $R \subseteq S$, $|R| = k$.

1: $R \leftarrow \emptyset$
2: **para** $p \leftarrow 1$ até $p = k$ **faça**
3: $\quad s_t \leftarrow argmax_{s_i \in S}(MMC(s_i))$
4: $\quad R_p \leftarrow R_{p-1} \cup s_t$
5: $\quad S \leftarrow S - s_t$
6: **fim para**
7: $R \leftarrow R_p$

Search Procedure). De maneira similar ao GRASP, é possível dividir o GNE em duas etapas: a etapa de construção e a etapa de busca local. Na etapa de construção, o algoritmo tenta, de forma iterativa, gerar um conjunto parcial (candidato). Como a quantidade de candidatos pode ser muito grande, o algoritmo usa uma lista restrita de candidatos (*Restricted Candidate List* - RCL) de tamanho fixo p ($p \geq k$). Deste modo, são escolhidos de forma aleatória p elementos que maximizem a equação (3.6), para serem inseridos na RCL. A partir da RCL, são escolhidos aleatoriamente os k elementos que farão parte da solução inicial. A etapa de busca local tenta melhorar a solução inicial obtida pela etapa de construção, buscando na vizinhança da solução inicial, uma solução de maior qualidade. Caso não exista uma solução melhor, a solução inicial é considerada um ótimo local (VIEIRA *et al.*, 2011).

Algoritmo 5 – GNE

Entrada: Conjunto de elementos S e a cardinalidade k do conjunto resultado R.
Saída: Conjunto resultado $R \subseteq S$, $|R| = k$.

1: $R \leftarrow \emptyset$
2: **para** $i \leftarrow 0$ *at* $i \leftarrow i_{max}, i \leftarrow i + 1$ **faça**
3: $\quad R' \leftarrow EtapadeConstrução()$
4: $\quad R' \leftarrow EtapadeBuscaLocal(R')$
5: $\quad$ **se** $F_{MMC}(s_q, R') > F_{MMC}(s_q, R)$ **então**
6: $\quad\quad R \leftarrow R'$
7: $\quad$ **fim se**
8: **fim para**

Yu, Lakshmanan e Amer-Yahia (2009) definem o algoritmo Swap (Algoritmo 6), um dos algoritmos mais simples para se construir o conjunto R. O algoritmo é dividido em duas fases. Na primeira fase, os k elementos mais próximos de s_q são inseridos em R. Na segunda fase, todos os elementos candidatos que não foram inseridos em R são ordenados de forma decrescente em relação à distância para o elemento s_q. Em seguida, cada elemento é comparado com os elementos inseridos em R para se verificar se a troca desses elementos aumenta o valor de alguma função objetivo, por exemplo a da equação (3.3). Um problema do Swap é que os elementos são sempre selecionados conforme a sua similaridade a s_q. Logo, não é possível garantir que R maximize a função objetivo (VIEIRA *et al.*, 2011).

Algoritmo 6 – SWAP

Entrada: Conjunto de elementos S e a cardinalidade k do conjunto resultado R.
Saída: Conjunto resultado $R \subseteq S$, $|R| = k$.

$R \leftarrow \emptyset$
enquanto $|R| < k$ **faça**
  $s_t \leftarrow argmax_{s_i \in S}(\delta_{sim}(s_q, s_i))$
  $R \leftarrow R \cup s_t$
  $S \leftarrow S - s_t$
fim enquanto
enquanto $|S| > 0$ **faça**
  $s_t \leftarrow argmax_{s_i \in S}(\delta_{sim}(s_q, s_i))$
  $S \leftarrow S - s_t$
  $R' \leftarrow R$
  para $s_j \in R$ **faça**
    se $F(s_q, \{R - s_t\} \cup s_t) > F(s_q, R')$ **então**
      $R' \leftarrow \{R - s_j\} \cup s_t$
    fim se
    se $F(s_q, R') > F(s_q, R)$ **então**
      $R \leftarrow R'$
    fim se
  fim para
fim enquanto

Todos os algoritmos discutidos neste seção tentam maximizar a similaridade e a diversidade do conjunto-resposta (R), segundo a preferência de diversidade do usuário. Dentre os algoritmos apresentados, os que mais se destacam são o GMC, GNE e MSD, pois conforme foi mostrado em trabalhos anteriores, a qualidade dos resultados gerados por estes algoritmos é maior que a dos demais (LOPES *et al.*, 2021; NOVAES; et. al, 2019; VIEIRA *et al.*, 2011). Contudo, o tempo de execução deles tendem a ser maior do que outros, como por exemplo o MMR.

Mesmo considerando a redução no espaço de busca, em geral, o tempo de execução destes algoritmos tende a ser alto, pois a maioria deles depende de um número quadrático ($O(m^2)$) de cálculos de distância, o que torna o processo de analisar muitos elementos ainda mais lento. No próximo Capítulo serão apresentadas outras abordagens de seleção de candidatos, que tentam otimizar ainda mais o tempo de execução dos algoritmos de diversidade.

3.4 Diversificação baseada em Cobertura

Uma busca por similaridade com diversidade baseada em cobertura retorna os k elementos mais similares a s_q considerando que os elementos do conjunto-resposta (R) devem ter uma distância mínima de separação (ϕ) entre eles. Geralmente, ϕ é definido pelo usuário e representa o mínimo de distância que um elemento deve ter em relação aos demais elementos em R, para ser considerado diverso (SANTOS *et al.*, 2013; HARITSA, 2009; SKOPAL *et al.*, 2009).

O algoritmo Motley (HARITSA, 2009) é um dos primeiros algoritmos conhecidos que utiliza essa abordagem para encontrar um conjunto k diversificado. Haritsa (2009) propõe que o algoritmo Motley utilize uma função de avaliação de diversidade (*divdist*) baseada no coeficiente de Gower. Entretanto, essa função pode ser substituída por outra função de avaliação, por exemplo, uma função de distância. Neste caso, uma função de distância δ_{div} é utilizada para avaliar a diversidade entre dois elementos.

O Motley (Algoritmo 7) considera que os elementos da base de dados estão ordenados pela similaridade ao elemento s_q. A partir disso, ele constrói o resultado iterativamente, selecionando a cada iteração o elemento mais próximo ao elemento central de busca (s_q) e verificando se a diversidade δ_{div} entre o elemento selecionado e os elementos já inseridos na resposta é maior que a distância Mínima de separação (ϕ) definida pelo usuário. A Figura 7(a) ilustra o resultado de uma busca utilizando o Motley.

Algoritmo 7 – Motley

Entrada: conjunto de dados S ordenado pela similaridade ao elemento s_q, cardinalidade k do conjunto resultado R.

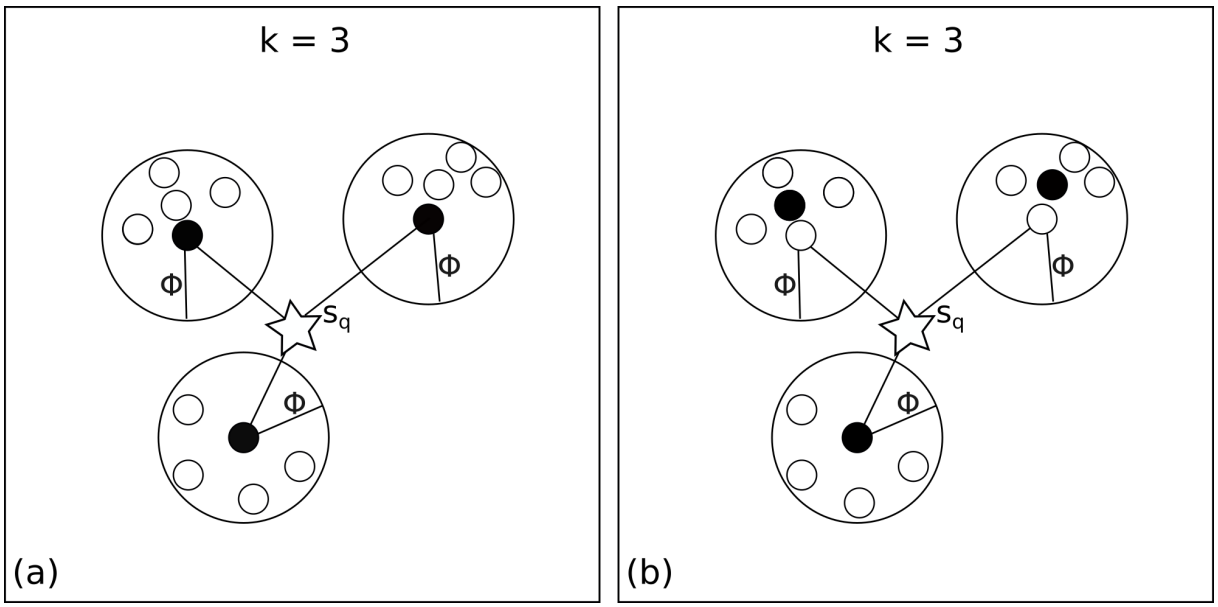
Saída: Conjunto resultado $R \subseteq S$, $|R| = k$.

1: $R \leftarrow \emptyset$
2: $s_t \leftarrow argmax_{s_i \in S}(\delta_{sim}(s_q, s_i))$
3: $S \leftarrow S - s_t$
4: $R \leftarrow R \cup s_t$
5: **enquanto** $|R| < k$ **faça**
6: $\quad s_t \leftarrow \text{argmax}_{s_i} \in S(\delta_{sim}(s_i, s_q))$.
7: $\quad$ **se** $\delta_{div}(s_t, r_i) \geq \phi, \forall r_i \in R$ **então**
8: $\quad\quad R \leftarrow R \cup s_t$.
9: $\quad$ **fim se**
10: **fim enquanto**

Skopal *et al.* (2009) apresenta dois algoritmos que utilizam esta mesma abordagem, o First-Match e o Centroid-Match. Se for considerada a mudança na função de avaliação de diversidade, o processo de construção do resultado utilizado pela abordagem First-Match é bastante similar a do Motley. Já o Centroid-Match processa o conjunto do mesmo modo, porém os elementos retornados são os centróides do conjunto de elementos considerados similares entre si. A Figura 7(b) ilustra o resultado de uma busca utilizando o Centroid-Match.

Todas as abordagens de diversificação de resultados discutidas até aqui solicitam algum parâmetro adicional (λ, m, ϕ) em relação as buscas por similaride tradicionais. Contudo, definir estes parâmetros com precisão pode ser uma tarefa difícil, pois é necessário ter conhecimento prévio sobre a distribuição e densidade dos dados na região de busca. Caso um parâmetro seja mal especificado, o usuário pode ser obrigado a re-executar a busca, variando os parâmetros, até que o mesmo esteja satisfeito com os resultados. Visando superar tais desvantagens, Santos *et al.* (2013) propôs um modelo de diversificação de resultados que não exige nenhum parâmetro adicional, chamado de Diversificação de Resultados baseada em Influência.

Figura 7 – Busca por similaridade com diversidade baseada em cobertura, para $k = 3$. Os pontos em preto são os elementos retornados pela busca. (a) Resultado de uma busca utilizando os algoritmos Motley ou First-Match, os elementos retornados são os elementos mais próximos de s_q que estão a pelo menos ϕ de distância dos outros elementos retornados. (b) Resultado de uma busca utilizando o Centroid-Match, os elementos retornados são os centroides das regiões definidas com o parâmetro ϕ.



Fonte: Elaborada pelo autor.

3.4.1 Diversificação de Resultados baseada em Influência

A Diversificação de resultados baseada em influência (RDI do inglês - *Result Diversification based on Influence*) também utiliza o conceito de distância Mínima de separação (ϕ) para encontrar um conjunto resposta (R) diversificado. No entanto, de maneira distinta da estratégia discutida na seção anterior, RDI considera que a distância Mínima de separação deve ser definida automaticamente e ajustada incrementalmente, utilizando a distribuição dos dados ao redor do elemento central de consulta (s_q), ou seja, a distância de separação entre r_i e $r_j \in R$ deve ser estimada usando a relação de distância entre s_q e os elementos r_i e r_j. Para isso, RDI considera que existe uma força atrativa entre os elementos do espaço de busca $S \subseteq \mathbb{S}$, chamada de influência (Definição 8), em que, quanto menor for a distância entre dois elementos, maior será a influência entre eles. Logo, quanto maior a influência entre dois elementos, maior é a probabilidade deles terem as mesmas características (SANTOS, 2017; SANTOS *et al.*, 2013).

Definição 8. (Influência $I(s_i,s_j)$). Dados dois elementos s_i e $s_j \in S$ e uma função de distância δ, a influência entre os elementos é definida por:

$$I(s_i, s_j) = \frac{1}{\delta(s_i, s_j)}$$

Durante uma busca, o conceito de influência pode ser utilizado para selecionar um conjunto R onde todos os elementos são mais influenciados pelo elemento s_q do que por qualquer

outro elemento em R. Por exemplo, um elemento s_j é mais influenciado por s_i do que por s_q se: $I(s_i, s_j) \geq I(s_q, s_j)$. A partir disso, podemos utilizar $I(s_i, s_q)$ para encontrar todos elementos que são mais influenciados por s_i do que por s_q. Neste caso, será encontrado o conjunto de influência forte de s_i (Definição 9).

Definição 9. (Conjunto de Influência Forte ($\bar{\bar{\text{S}}}$)). Dado o elemento central de consulta s_q e um elemento s_i, o conjunto de influência forte de s_i são todos os elementos s_j que respeitam à seguinte propriedade:

$$\bar{\bar{\text{S}}}(s_i, s_q) = \{s_j \in S \mid (I(s_i, s_q) \geq I(s_j, s_q)) \wedge (I(s_j, s_i) \geq I(s_j, s_q))\} \tag{3.7}$$

O conjunto de influência forte permite definir buscas por similaridade com diversidade. Formalmente, uma busca por similaridade com diversidade utilizando RDI (Definição 10) deve garantir que nenhum elemento na resposta (R) faça parte do conjunto de influência forte de nenhum outro elemento em R (SANTOS *et al.*, 2013).

Definição 10. (Diversificação de Resultados baseada em Influência) - RDI).

$$R = \{r_i \in S | \forall r_j \in R, r_i \neq r_j \rightarrow r_i \notin \bar{\bar{\text{S}}}(r_j, s_q)\}$$

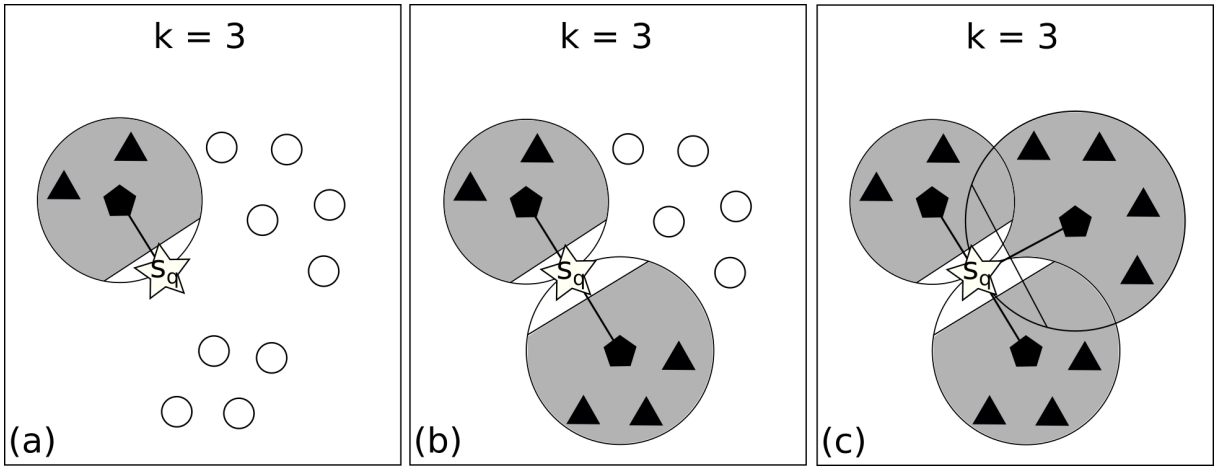
A partir deste conceito, Santos *et al.* (2013) definiram a técnica Resultados Melhores com Diversificação baseada em Influência (BRID - *Better Results with Influence Diversification*), que é uma abordagem incremental para a construção do conjunto resposta diversificado R, de uma busca por similaridade, seja ela uma Rq ou k-NNq.

O BRID utiliza uma abordagem incremental em que, a cada iteração, o elemento mais próximo ao elemento s_q, que respeite as restrições do RDI, é inserido no conjunto R. Desta forma, a cada escolha o BRID garante que a influência de s_q sobre o elemento escolhido é máxima, ou seja, o elemento escolhido a cada iteração é o mais similar a s_q e que não é influenciado por nenhum outro elemento já escolhido. Por exemplo, dados dois elementos s_i, $s_j \in S$, em que $s_i \notin \bar{\bar{\text{S}}}(s_j, s_q)$ e $s_j \notin \bar{\bar{\text{S}}}(s_i, s_q)$, caso $I(s_i, s_q) > I(s_j, s_q)$, pode-se dizer que s_i é o elemento não-influenciado mais próximo a s_q que pode ser inserido em R e a intensidade de influência $I(s_i, s_q)$, pode ser usada para podar todos os elementos mais similares a s_i do que à s_q, e neste caso, $\phi_i = \delta(s_i, s_q)$. Um detalhe interessante desta abordagem, é que a distância mínima de separação aumenta conforme a distância entre o elemento selecionado e s_q. Isso, permite que a distância mínima de separação se ajuste a distribuição dos dados em torno do elemento s_q.

A Figura 8 exemplifica o processo de construção de uma busca aos 3-vizinhos mais próximos utilizando a técnica BRID. Os pentágonos representam os elementos não influenciados, os triângulos representam os elementos influenciados e os círculos representam os elementos

que ainda não foram analisados. Na Figura 8(a), inicialmente, o elemento s_i mais próximo a s_q é inserido em R. Em seguida, todos os elementos que estão no conjunto de influência forte de s_i são ignorados. Na Figura 8(b e c), um novo elemento próximo a s_q, que não faz parte do conjunto de influência forte de algum elemento $r_i \in R$ é encontrado e sua influência em relação à s_q é utilizada para excluir outros elementos.

Figura 8 – Construção de um conjunto R diversificado usando o BRID. Os pentágonos são elementos adicionados ao conjunto R, os triângulos são elementos presentes no conjunto de influência forte de algum $r_i \in R$ e os círculos representam os elementos que ainda não foram analisados.



Fonte: Elaborada pelo autor.

Os algoritmos baseados em distância de separação são mais eficientes do que aqueles que tentam resolver o problema de otimização, pois o processo de diversificação é mais direto: basta verificar se o elemento é similar à s_q e diverso a todos os elementos já inseridos na resposta para se saber se ele deve ou não ser inserido na resposta. Entretanto, o processo mais comum para executar os algoritmos dessa abordagem é ordenar os elementos da base de dados em relação à sua distância ao elemento s_q, o que pode tornar o processo lento, pois é preciso comparar o elemento s_q com todos os elementos da base de dados, o que exige muitas comparações distância.

3.5 Considerações Finais

Neste capítulo foram apresentados os principais conceitos e algoritmos de diversificação de buscas por similaridade. O conceito de diversidade explorado em todas as abordagens descritas, permitem evitar que elementos muito similares entre si sejam retornados. Contudo, para incluir diversidade nas buscas é preciso realizar muito mais operações do que em uma busca por similaridade convencional, principalmente, para as buscas baseadas em otimização, que demandam muito mais tempo e realizam muitas comparações de distância.

No próximo capítulo, são apresentadas algumas abordagens que tentam reduzir os custos envolvidos no processo de diversificação de resultados baseada em novidade/otimização.

CAPÍTULO

4

TRABALHOS RELACIONADOS

4.1 Considerações Iniciais

As buscas por similaridade com diversidade permitem aumentar a qualidade das repostas geradas pelas buscas por similaridade convencionais. Contudo, incluir diversidade às buscas por similaridade aumenta os custos da busca. Por exemplo, considerando uma abordagem ingênua, em uma busca por similaridade é necessário comparar a distância entre os elementos da base de dados e o elemento central de consulta (s_q). Já em uma busca por similaridade com diversidade é necessário comparar os elementos da base de dados com o elemento s_q e, pelo menos, todos os outros elementos já recuperados. Além disso, para recuperar k elementos diversos, é necessário avaliar uma quantidade de elementos maior do que seria usada em buscas por similaridade. Por conta desses problemas, foram propostas abordagens de diversificação de resultados que não só aumentam a qualidade dos resultados gerados, mas que podem ser computadas de forma mais eficaz e eficiente.

Este capítulo apresenta algumas abordagens encontradas na literatura que tentam reduzir os custos dos modelos de diversidade baseados em novidade. Na Seção 4.2 são apresentadas as abordagens que tentam reduzir o custo das buscas por similaridade com diversidade, todas elas tendo como objetivo selecionar um pequeno conjunto de elementos candidatos que possuam um certo grau de diversidade. Na Seção 4.3 são apresentadas as conclusões deste capítulo.

4.2 Seleção de Candidatos

Conforme discutido na Seção 3.3, as buscas por similaridade com diversidade baseadas em novidade são modeladas como um problema de otimização, onde diversidade e similaridade concorrem conforme a preferência de diversidade do usuário(λ). Contudo, por se tratar de um problema de análise combinatória, essa abordagem tem um custo muito elevado. A abordagem mais utilizada para tentar contornar este problema é selecionar um conjunto candidato. Neste

caso, o objetivo é selecionar um pequeno conjunto de elementos que sejam similares a s_q, com isso, a complexidade de tempo dos algoritmos que antes eram $O(n^2)$ é reduzida para $O(m^2)$, onde n é o número de elementos da base de dados e m o número de elementos candidatos, onde $m << n$.

Tipicamente, essa abordagem usa uma busca por similaridade para recuperar os m elementos candidatos (Rq ou k-NNq). Por conta disso, apesar de ser uma estratégia bastante promissora, alguns problemas podem ser encontrados. Um deles é que o número de elementos candidatos selecionados impacta bastante no passo seguinte de diversificação, isso acontece, principalmente, porque os elementos candidatos são selecionados por uma busca por similaridade, que tende a retornar elementos similares a s_q, mas que também retornar muitos elementos que são similares entre si. Logo, se o conjunto candidato for muito pequeno, a diversidade entre os elementos vai ser muito baixa, o que impacta negativamente a qualidade do resultado. Contudo, aumentar muito o número de elementos candidatos pode impactar positivamente a qualidade dos resultados, mas, consequentemente, aumenta o tempo de execução dos algoritmos de diversificação e o tempo para executar a busca. A partir disso, temos que, uma boa abordagem de seleção de candidatos deve ser capaz de retornar o menor número possível de elementos candidatos, que possuam similaridade a s_q e que tenham um certo grau diversidade entre si. Na literatura podem ser encontradas diferentes abordagens que permitem selecionar elementos candidatos que atendam a, pelo menos, um desses pontos.

Ravi, Rosenkrantz e Tayi (1994) desenvolveram dois algoritmos aproximados para o problema de diversidade (dispersão) máxima, o GMM e o GMA. O GMM (Algoritmo 8) resolve o problema considerando que a diversidade no conjunto resposta é definida pela menor distância entre os elementos retornados. Já o GMA considera que a diversidade é dada pela média das distâncias entre os elementos. Ambos os algoritmos podem ser executados em tempo $O(kn)$, onde n é a cardinalidade do conjunto de dados. Contudo, como tanto o GMM quanto o GMA tendem a selecionar os elementos mais distantes entre si, eles não levam em consideração a similaridade em relação a um elemento central de consulta e nem a preferência de diversidade do usuário, ou seja, eles resolvem o problema de se encontrar um conjunto de k elementos que estejam o mais distante possível. Os dois trabalhos apresentados a seguir utilizam o GMM ou o GMA como uma rotina interna para encontrar k elementos diversos.

A abordagem definida por Drosou e Pitoura (2014) foi inicialmente projetada para gerar respostas diversificadas em *data streams*, mas pode ser modificada para responder a buscas por similaridade com diversidade em dados estáticos. Essa abordagem é baseada no algoritmo GMM e utiliza uma Cover Tree (BEYGELZIMER; KAKADE; LANGFORD, 2006; KOLLAR, 2006) para organizar os dados e permitir que as buscas sejam executadas de forma eficiente. Uma Cover tree(CT) é uma árvore que organiza os dados baseando-se na distância entre eles. Mais especificamente, uma CT impõe que a distância entre os elementos de um mesmo nível C_i têm de ser maior que b^i, ou seja, s_j,$s_k \in C_i : \delta_{div}(s_j, s_k) > b^i, b > 1$. Com isso, é possível garantir que

Algoritmo 8 – GMM

Entrada: Conjunto de elementos S e a cardinalidade k do conjunto resultado R.
Saída: Conjunto resultado $R \subseteq S$, $|R| = k$.

1: $R \leftarrow \emptyset$
2: $s_t \leftarrow$ qualquer ponto de S.
3: $R \leftarrow R \cup s_t$
4: $S \leftarrow S - s_t$
5: **enquanto** $|R| < k$ **faça**
6: $\quad s_t \leftarrow argmax_{s_i \in S}(min_{r_i \in R}(\delta_{div}(s_i, r_i))$
7: $\quad R \leftarrow R \cup s_t$
8: $\quad S \leftarrow S - s_t$
9: **fim enquanto**
10: **retorna** R

existe uma medida de diversidade (b^i) entre os elementos que estão em um mesmo nível da árvore. Na abordagem proposta pelos autores, a CT é adaptada para que cada um dos seus níveis contenha uma possível solução do algoritmo GMM e, a partir disso, foram propostos vários algoritmos que usufruem desta estrutura para gerar as respostas diversificadas. O Algoritmo 9 mostra o funcionamento de um dos algoritmos propostos. Esse algoritmo, basicamente, busca encontrar o maior nível da CT que tenha pelo menos k elementos e, então, ele seleciona aleatoriamente os k elementos do nível. A complexidade desse algoritmo é $O(n+k)$, onde n é o número de elementos na Cover Tree e altura máxima da árvore[1]. Entretanto, a complexidade do algoritmo de construção da Cover-tree é quadrático no número de elementos, que é equivalente a complexidade dos algoritmos apresentados na Seção 3.3, o que inviabiliza construir a estrutura para todos os elementos da base de dados. Por conta disso, uma das formas de adaptar essa abordagem para as buscas por similaridade é utilizar a abordagem como filtro. Neste caso, são recuperados os m mais próximos a s_q e então esses elementos são utilizados para construir a CT (WANG; MELIOU; MIKLAU, 2018).

Algoritmo 9 – Level-Basic

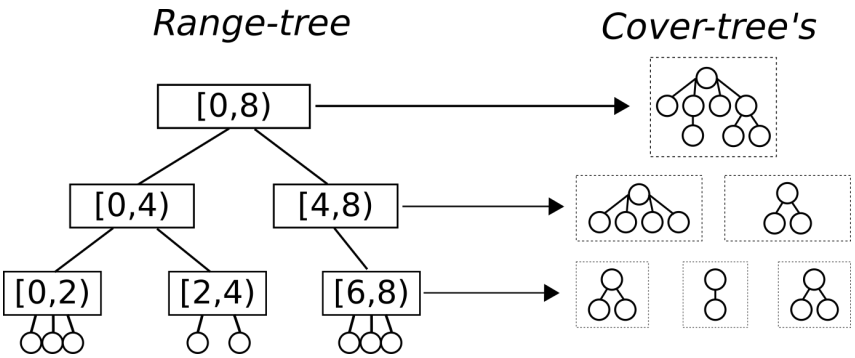
Entrada: Cover Tree CT e um inteiro k
Saída: Conjunto de elementos R: $|R| = k$.

1: $l \leftarrow CT.l_{max}$
2: **enquanto** $|CT[l]| < k$ **faça**
3: $\quad l \leftarrow l - 1$
4: **fim enquanto**
5: $R \leftarrow$ qualquer subconjunto de k elementos de $CT[l]$.
6: **retorna** R

Em (WANG; MELIOU; MIKLAU, 2018) é proposto o RC-Index, uma estrutura de indexação baseada nas estruturas Cover Tree e Range Tree. A abordagem proposta pode ser dividida em três partes: construção do RC-Index, extração de candidatos e diversificação. Na

[1] A altura máxima de uma Cover Tree é $O(n)$.

Figura 9 – Exemplo de um RC-Index, considerando que a Range-tree é unidimensional. Cada nó da Range-tree é mapeado em uma Cover-tree.



Fonte: Adaptada de Wang, Meliou e Miklau (2018).

etapa de construção do RC-Index, a Range-tree é construída e, para cada um dos seus nós é construída uma Cover-tree. A Figura 9 ilustra este processo. Na etapa de extração de candidatos, dado o intervalo de busca e a quantidade (k) de elementos a serem retornados, os nós da Range Tree contidos no intervalo de busca são acessados, e as Cover Tree's correspondentes são utilizadas para extrair os elementos candidatos (X^C). O Algoritmo 10 mostra como os candidatos são extraídos de cada uma das Cover Tree's. Inicialmente, é selecionado o maior nível da Cover Tree, que tem mais que k elementos. Em seguida, são selecionados os elementos que estão em algum nível abaixo daquele encontrado. Os autores do trabalho, utilizam por padrão os elementos que estão três níveis abaixo. Já na etapa de diversificação, é utilizado um algoritmo similar ao GMM para extrair os k elementos diversos do conjunto candidato.

A complexidade de tempo de construção dessa abordagem é menor do que a abordagem anterior e consegue gerar melhores resultados. Contudo, a qualidade das respostas tende a ser menor do que as geradas pelo GMM. Além disso, essa abordagem pode ser ineficiente para espaços de alta dimensionalidade, pois sua complexidade de construção é $O(\gamma^6 nlog^{d+1}(n))$ e sua complexidade de espaço é $O(nlog^d(n))$.

Santos *et al.* (2015) apresenta várias abordagens de seleção de elementos candidatos, que já consideram a diversidade entre os elementos selecionados. Dentre as abordagens propostas duas se destacam: a seleção aleatória (RnD) e a por Influência (RDI). A abordagem RnD consiste em selecionar aleatoriamente m elementos que estão no máximo a ξ_{max} de distância do elemento s_q. Já abordagem de RDI utiliza o *BRID* (Seção 3.4.1) para selecionar os m elementos candidatos. Essas duas abordagens de seleção conseguiram gerar conjuntos candidatos menores, que possuem diversidade equivalente a aquela encontrada pela abordagem tradicional de busca por similaridade. Como consequência, o tempo de execução dos algoritmos de diversificação foi bastante reduzido e qualidade das resposta geradas foi mantida. Entretanto, as abordagens propostas podem apresentar alguns problemas quanto a eficiência e escalabilidade. A abordagem RnD seleciona os elementos sem realizar nenhum tipo de comparação, logo, não é possível garantir que existe similaridade a s_q ou diversidade entre os elementos selecionados. Já abordagem RDI é uma

Algoritmo 10 – Extração de Candidatos

Entrada: Um inteiro $k > 0$, Conjunto de Cover-tree's $T = \{CT_1, \cdots, CT_n\}$ e um inteiro $\gamma > 0$.
Saída: Conjunto Candidato X^C

```
1: X^C ← ∅
2: para CT ∈ T faça
3:     X^C ← X^C ∪ ExtractTree(CT, k, γ)
4: fim para
5: retorna X^C
6: procedimento ExtractTree((CT, k, γ))
7:     se |CT| ≤ k então
8:         retorna Todos Elementos em CT.
9:     fim se
10:    l_k ← argmax_l(|CT| ≥ k)
11:    l ← max{l_k − γ, l_min}          ▷ Tenta encontrar um nível com ainda mais elementos
12:    retorna CT_l                      ▷ Retorna todos os elementos do nível l
13: fim procedimento
```

abordagem de diversificação baseada em cobertura, que tende a ser muito mais exploratória e que não restringe o seu espaço de busca. Por conta disso, não é incomum essa abordagem analisar todos os elementos da base de dados, o que pode tornar o processo de seleção mais custoso do que outras abordagens.

A Tabela 1 apresenta uma comparação dos algoritmos listados nesta seção, em relação aos aspectos esperados por uma abordagem de seleção de candidatos para as buscas por similaridade com diversidade. É esperado que a complexidade de tempo para gerar o conjunto candidato seja menor que a dos algoritmos listados na Seção 3.3. Além disso, o algoritmo deve ser capaz de retornar elementos que sejam similares ao centro de consulta s_q e que possuam diversidade entre os elementos selecionados.

4.3 Considerações Finais

Neste capítulo foram apresentados algumas abordagens que podem otimizar o tempo de execução das buscas por similaridade com diversidade baseadas em novidade. Todas as abordagens apresentadas seguem a estratégia de seleção de candidatos. Essa estratégia tenta reduzir o tempo de execução das buscas, reduzindo a quantidade de elementos analisados pelos algoritmos de diversificação. Tipicamente essas abordagens selecionam um subconjunto dos elementos da base de dados para serem utilizados durante a busca. Cada abordagem tenta selecionar os elementos de forma diferente. Contudo, muitos dos métodos utilizados apresentam pontos que podem ser melhorados, sejam eles em relação ao tempo de execução da abordagem de seleção, a qualidade dos elementos selecionados e/ou a escalabilidade dos métodos.

No próximo capítulo são apresentadas as abordagens desenvolvidas neste trabalho. Elas

Tabela 1 – Aspectos das abordagens que visam selecionar elementos candidatos, para as buscas por similaridade com diversidade baseada em novidade.

Aspectos / Abordagem	Complexidade de tempo	Complexidade de espaço	Considera a similaridade ao elemento s_q	Considera a diversidade entre os elementos
Algoritmos da Seção 3.3	$O(m^2)$	$O(n)$	Sim	Sim
Busca por Similaridade	$O(n)$	$O(n)$	Sim	Não
Drosou e Pitoura (2014)	$O(m^2)$	$O(m)$	Sim	Não
Wang, Meliou e Miklau (2018)	$O(\beta log^d(n)) : \beta < n$	$O(nlog^d(n))$	Não	Sim
Santos *et al.* (2015)	$O(n)$	$O(n)$	Sim	Sim
Abordagens desenvolvidas	$O(log^{\mathscr{D}}(n))$	$O(nlog^{\mathscr{D}}(n))$	Sim	Sim

utilizam a Range Tree (RT), assim como o RC-Index, para particionar os elementos da base dados, mas de forma diferente, a RT foi acoplada à Omni-Technique, dando origem a uma nova estrutura de dados que tem uma complexidade de espaço, de busca e construção menor e que pode ser utilizada para executar buscas por similaridade. Além disso, todas as abordagens desenvolvidas utilizam as partições geradas pela RT para selecionar elementos candidatos que maximizam tanto a similaridade a s_q, quanto a diversidade entre os elementos selecionados.

CAPÍTULO

5

PARTICIONAMENTO E SELEÇÃO DE CANDIDATOS

5.1 Considerações Iniciais

No Capítulo 2 foram apresentados os conceitos básicos relacionados as buscas por similaridade. No Capítulo 3 foram apresentados os principais conceitos e algoritmos de busca por similaridade com diversidade. No Capítulo 4 foram apresentadas algumas abordagens relacionadas aos objetivos deste trabalho. Neste capítulo são apresentadas as abordagens desenvolvidas neste trabalho.

O objetivo deste trabalho de mestrado foi desenvolver métodos que aumentem a eficiência das buscas por similaridade com diversidade, mas especificamente, daquelas baseadas em novidade(otimização). As principais contribuições, que serão apresentadas neste capítulo, visam reduzir o tempo de execução dos algoritmos de diversidade utilizando a abordagem de seleção de candidatos.

5.1.1 Busca por similaridade com diversidade baseada em novidade

Como foi discutido nas Seções 3.3 e 4.2, uma busca por similaridade com diversidade baseada em novidade é modelada como um problema de otimização, e pode ser dividida em duas etapas: seleção de candidatos e diversificação. Tipicamente, na etapa de seleção de candidatos, os m elementos mais próximos a s_q são escolhidos como candidatos e, na etapa de diversificação, os elementos candidatos que maximizam uma função objetivo de diversidade são selecionados, utilizando algum algoritmo de diversificação. Por ser um problema de otimização, várias iterações e cálculos de distância são necessários para gerar as respostas diversificadas, sendo que, quanto menor for o conjunto candidato, mais rapidamente as respostas serão geradas. Entretanto, considerando que os candidatos são selecionados por meio de uma busca por similaridade, se o

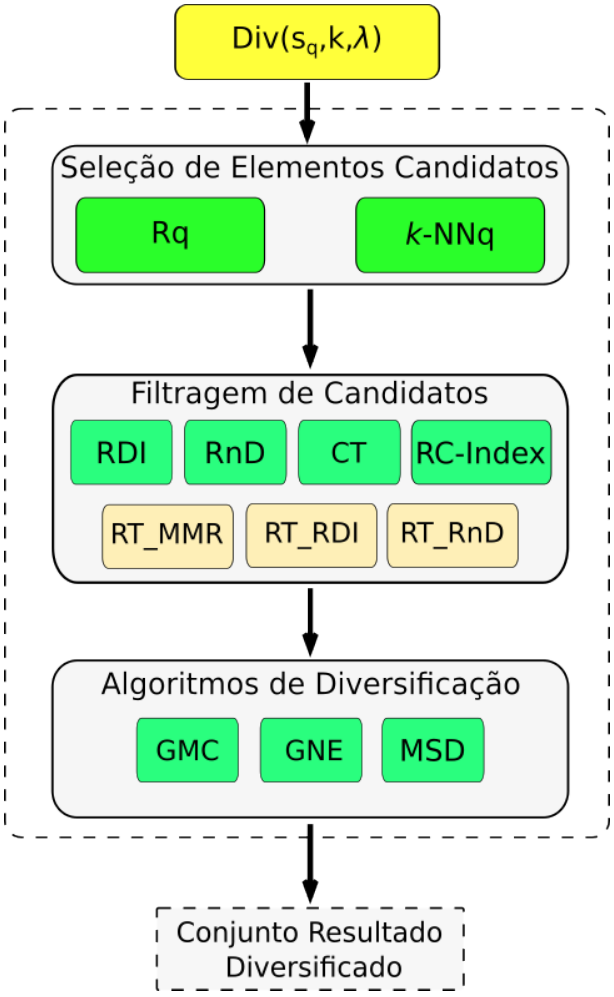
conjunto candidato for muito pequeno, a diversidade entre os elementos pode ser muito baixa. Logo, gerar conjuntos candidatos de baixa cardinalidade com alta similaridade ao elemento s_q e diversidade entre os elementos é uma opção para agilizar as buscas. Por conta disso, um dos principais objetivos deste trabalho é adicionar uma terceira etapa ao processo de diversificação (Figura 10): a filtragem de candidatos. Essa etapa tem o objetivo de remover elementos que não agregam valor ao resultado final, tais como, elementos que são muito similares entre si ou muito dissimilares ao elemento central de consulta(s_q). Neste sentido, a ideia principal do processo de filtragem de candidatos é reduzir o número de elementos retornados por uma busca por similaridade, tornando assim, a execução do algoritmo de diversificação mais rápida. Contudo, as abordagens utilizadas para filtrar os elementos candidatos não podem ser muito custosas, a ponto de tornarem o processo de seleção, filtragem e diversificação mais lento do que aplicar os algoritmos de diversificação sobre todos os elementos selecionados. A fim de garantir que esses resultados sejam alcançados, também foram desenvolvidas abordagens de seleção/filtragem de elementos candidatos que, de maneira diferente das abordagens discutidas na seção 4.2, têm como foco gerar conjuntos candidatos que possuem elementos similares a s_q e que sejam diversos entre si. Além disso, as abordagens desenvolvidas têm como premissa, garantir que o processo de filtragem não seja mais custoso do que aplicar diretamente os algoritmos de diversificação.

As abordagens desenvolvidas neste trabalho são baseadas em dois conceitos: Particionamento espacial e Seleção de candidatos. O primeiro tem como objetivo particionar o conjunto de dados em pequenos subconjuntos, tal que, o custo de encontrar elementos que sejam similares a s_q e o custo de selecionar candidatos diversos, em cada subconjunto, seja o menor possível. O objetivo do segundo conceito é selecionar rapidamente, de cada subconjunto, elementos que maximizem tanto a similaridade quanto a diversidade. Para particionar o conjunto de dados, foi desenvolvida a *Omni-Range Tree*, uma nova estrutura de dados que combina a Range Tree com a Omni-Technique.

5.2 ORTree

Uma das principais etapas das abordagens desse trabalho é a de particionar o conjunto de dados. Na literatura, diferentes abordagens de partição podem ser encontradas (Quad-Tree, R-Tree, VP-Tree, dentre outras). Cada uma delas possui alguma característica que as diferencia. Neste sentido a Range Tree - RT (Seção 2.6) é uma estrutura que permite particionar o conjunto de dados considerando os vetores de características dos elementos da base de dados. A abordagem descrita neste trabalho é baseada na RT, contudo, outras estruturas podem ser utilizadas. A RT foi escolhida neste caso por conta da sua baixa complexidade de busca e, pela facilidade de implementação. Contudo, conforme foi discutido anteriormente, a RT não foi planejada para executar buscas por similaridade e, por conta disso, é preciso fazer algumas modificações na RT para que essas buscas sejam executadas dentro dessa estrutura. Para solucionar esse problema, foi desenvolvida a *Omni-Range Tree* (**ORTree**), uma nova estrutura de dados que utiliza as

Figura 10 – Proposta de processo de diversificação de resultados, baseada em função objetivo.



Fonte: Elaborada pelo autor.

omni-coordenadas definidas pela Omni-Technique, ao invés dos vetores de características, para construir a RT e, consequentemente, particionar os dados. Neste caso, como as *omni*-coordenadas são geradas considerando as distâncias entre os elementos, a ORTree organiza os dados considerando o espaço das distâncias ao invés do espaço das características, o que permite que as buscas por similaridade sejam executadas. Mais especificamente, a RT realiza as buscas considerando os intervalos entre as características dos elementos, já ORTree realiza as buscas considerando os intervalos de distância entre os elementos. Por conta disso, a ORTree permite que as buscas por similaridade (tanto R_q e k-NN$_q$), que também são baseadas nas relações de distância, sejam executadas utilizando os nós da estrutura.

Além disso, a dimensionalidade das *omni*-coordenadas é definida pela dimensão fractal do conjunto de dados, o que geralmente é menor que a dimensionalidade dos dados, ou seja, o número de *omni*-coordenadas é menor que o número de características dos elementos. Consequentemente, as complexidades de espaço, busca e construção da ORTree são menores que os da RT. Outro ponto importante, é que as *omni*-coordenadas podem ser geradas para qualquer

domínio de dados que respeite as propriedades dos espaços métricos. Desta forma, a ORTree pode ser utilizada para organizar dados como cadeias de caracteres e textos, domínios aos quais a RT não suporta. A Tabela 2, apresenta as principais diferenças entre RT e ORTree.

Tabela 2 – Aspectos das estruturas RT e ORTree. Onde n é a cardinalidade da base de dados, d a dimensionalidade dos vetores de características e $\mathscr{D}$ a dimensão fractal do conjunto de dados ou a dimensionalidade das *omni*-coordenadas.

	Complexidade de Espaço	Complexidade de Busca	Complexidade de Construção	Domínio de Dados	Buscas Suportadas
Range Tree	$O(nlog^d(n))$	$O(log^d(n))$	$O(nlog^d(n))$	Numérico	por intervalo
Omni-Range Tree	$O(nlog^{\mathscr{D}}(n))$	$O(log^{\mathscr{D}}(n))$	$O(nlog^{\mathscr{D}}(n))$	Espaços Métricos	Rq e k-NNq

5.2.1 ORTree - Construção

O processo de construção da ORTree é bastante similar ao da RT, só que utilizando as omni-coordinadas ao invés dos atributos originais. O Algoritmo 11 mostra o processo de construção da ORtree. Depois de escolher os pivôs, as *omni*-coordenadas de cada elemento são calculadas (linhas 1-9) e geram um conjunto de dados $\mathscr{D}$-dimensional. A partir disso, a ORTree é construída da seguinte forma. Os elementos são ordenados considerando a primeira dimensão e inseridos no nó raiz (linhas 10-11). Neste ponto, os elementos são particionados em dois subconjuntos considerando a mediana da primeira dimensão do vetor de *omni*-coordenadas. Em seguida são construídos para a dimensão atual os filhos à esquerda e à direita (linhas 17-18) e, caso exista uma próxima dimensão, é criado o filho 'next' que constrói o nó considerando a próxima dimensão (linhas 19-21). Esse processo se repete recursivamente até que não existam mais elementos ou dimensões.

Gerar as *omni*-coordenadas tem complexidade de tempo de $O(n)$. O restante do processo de construção segue os mesmos passos da RT, logo, a complexidade de tempo do algoritmo de construção é $O(nlog^{\mathbb{D}}(n))$, onde $\mathscr{D}$ é a dimensão fractal do conjunto de dados (ou o número de pivôs).

Um detalhe importante sobre a implementação da ORTree é que o algoritmo foi projetado para construir a árvore até que não existam mais elementos a serem inseridos. Contudo, raramente, se deseja realizar uma busca que retorna um único elemento, logo, o algoritmo pode ser adaptado para encerrar a construção quando um número mínimo de elementos no nó for alcançado, o que reduz as complexidades de tempo e espaço da estrutura.

5.2.2 ORTree - Busca

Assim como a Range Tree, a ORTree responde as buscas considerando um determinado intervalo de busca. No caso da RT esse intervalo usa o espaço das características, já na ORTree o intervalo utiliza o espaço das distâncias (ou *omni*-coordenadas). Neste sentido, para realizar algum tipo de busca é preciso definir qual o intervalo de busca. Na RT o usuário precisa definir

Algoritmo 11 – Construção ORTree

Entrada: Conjunto de elementos pivô $\mathscr{P}$, δ Função de distância métrica e o Conjunto de dados S.

Saida: ORTree.

1: $\mathscr{O}_S \leftarrow \emptyset$ // conjunto de *omni*-coordenadas.
2: **para** $\forall\ s_i \in S$ **faça**
3: $\quad\mathscr{O}_{si} \leftarrow \emptyset$
4: $\quad$**para** $\forall\ p \in \mathscr{P}$ **faça**
5: $\quad\quad coord \leftarrow \delta(s_i, p)$ //coordenada correspondente a p.
6: $\quad\quad\mathscr{O}_{si} \leftarrow \mathscr{O}_{si} \cup coord$
7: $\quad$**fim para**
8: $\quad\mathscr{O}_S \leftarrow \mathscr{O}_S \cup \mathscr{O}_{si}$
9: **fim para**
10: Sort($\mathscr{O}_S$, 0) // Ordenando os elementos considerando a primeira dimensão.
11: ORTree.root = $\leftarrow$ Construct($\mathscr{O}_S$, 0)
12: **retorna** ORTree
13: **função** CONSTRUCT(dataset, dim)
14: $\quad$**se** dataset.size == 0 **então**
15: $\quad\quad$**retorna** NULL
16: $\quad$**fim se**
17: $\quad$node.left $\leftarrow$ CONSTRUCT(left_half(dataset), dim)
18: $\quad$node.right $\leftarrow$ CONSTRUCT(right_half(dataset), dim)
19: $\quad$**se** dim $< |\mathscr{P}| - 1$ **então**
20: $\quad\quad$Sort(dataset, dim + 1). // Ordenando os elementos considerando a próxima dimension.
21: $\quad\quad$node.next $\leftarrow$ CONSTRUCT(dataset, dim + 1)
22: $\quad$**fim se**
23: $\quad$**retorna** node
24: **fim função**

quais são as faixas de valores de cada característica e, assim, construir o intervalo de busca. Na ORTree este mesmo processo pode ser realizado. Entretanto, o objetivo é responder as buscas por similaridade, logo, é mais interessante definir um processo que seja capaz de criar o intervalo de busca, a partir dos parâmetros da busca por similaridade (s_q, k ou ξ).

Conforme foi discutido na Seção 2.4, uma busca por abrangência (R_q) deve retornar todos os elementos que possuem uma distância, em relação a s_q, menor ou igual que ξ, ou seja, todos os elementos s_i da base de dados, tal que: $\delta(s_q, s_i) \leq \xi$. Quando consideramos as *omni*-coordenadas dos elementos, da base de dados ($\mathscr{O}_{si}$) e do elemento s_q ($\mathscr{O}_{sq}$), e as propriedades dos espaços métricos, temos que os elementos a serem retornados pela busca são aqueles cuja a diferença entre suas *omni*-coordenadas e as coordenadas de s_q são menores que ξ, ou seja, $|\mathscr{O}_{si} - \mathscr{O}_{sq}| \leq \xi$. Neste caso, todos os elementos cuja as *omni*-coordenadas estão dentro do intervalo: $[\mathscr{O}_{sq} - \xi, \mathscr{O}_{sq} + \xi]$, devem ser considerados pela busca. Esse intervalo pode ser utilizado pela ORTree para recuperar os elementos solicitados pela busca. Lembrando que esse processo pode retornar alguns elementos falso-positivos que não estão dentro do limiar de busca $\delta(s_q, s_i) \leq \xi$ e por isso é necessário excluir esse elementos ao final.

O Algoritmo 12 mostra o processo de criação do intervalo de busca para mais de um pivô, considerando os parâmetros de uma busca por abrangência (s_q e ξ). Inicialmente (linhas 3-6), são geradas as *omni*-coordenadas do elemento central de consulta (s_q). A partir disso (linhas 7-9), o intervalo de busca (RD_ξ) de cada uma das omni-dimensões é definido pela soma/diferença entre o limiar de dissimilaridade (ξ) e as *omni*-coordenadas de s_q: $\{[O_{sq_i} - \xi, O_{sq_i} + \xi]\}$, onde i representa a i-ésima dimensão e/ou o i-ésimo pivô p_i. Neste caso, RD_ξ é um vetor que contém os intervalos de busca para cada uma das $\mathscr{D}$-dimensões de *omni*-coordenadas.

Algoritmo 12 – Criação do Intervalo de Busca

Entrada: Conjunto de elementos pivôs $\mathscr{P}$, uma função de distância metrica δ, o elemento central de consulta s_q e o limiar de dissimilaridade ξ.

Saída: Conjunto de intervalos de cada dimensão das omni-coordinadas RD_ξ.

1: $RD_r \leftarrow \emptyset$
2: $\mathscr{O}_{sq} \leftarrow \emptyset$
3: **para** $\forall\ p \in \mathscr{P}$ **faça**
4: $\quad coord \leftarrow \delta(s_q, p)$ //coordenada correspondente a p.
5: $\quad \mathscr{O}_{sq} \leftarrow \mathscr{O}_{sq} \cup coord$
6: **fim para**
7: **para** $\forall\ coord \in \mathscr{O}_{sq}$ **faça**
8: $\quad RD_\xi \leftarrow \{coord - \xi, coord + \xi\}$
9: **fim para**
retorna RD_ξ

Dado o intervalo de busca (RD_ξ), a ORTree pode então ser utilizada para retornar os nós que estão contidos dentro do intervalo. O Algoritmo 13 mostra esse processo de busca. Inicialmente, é verificado se existe alguma interseção entre RD_ξ e os limites (inferior e superior) do nó raiz, caso não exista a busca é encerrada, pois neste caso, não existem nós na ORTree que estão contidos dentro do intervalo (linhas 5-7). Em seguida, é verificado se os limites do nó raiz estão totalmente contidos dentro do intervalo RD_ξ, caso esteja, um nó com elementos dentro do intervalo de busca foi encontrado e, neste momento, duas decisões podem ser tomadas: (*i*) retornar o nó encontrado ou (*ii*) investigar, se existir, à ORTree criada pelo filho 'next', que organiza os elementos considerando a próxima dimensão (linhas 8-13). Investigar os nós da próxima dimensão é equivalente, adicionar mais um pivô ao processo de busca e, conforme discutido na Seção 2.5.1, quanto mais pivôs são utilizados durante a busca, menos falsos-positivos serão retornados pela busca e menos comparações de distância precisaram ser feitas. Por isso, neste trabalho, optou-se por utilizar sempre a segunda opção. Por fim, caso nenhuma das checagens anteriores tenham sido satisfeitas, é preciso investigar os filhos (direita e esquerda) do nó raiz. Neste caso, o processo de busca se repete recursivamente, utilizando os nós filhos (linhas 15-17). Vale a pena ressaltar que, caso não existam nós que estejam totalmente contidos pelo intervalo de busca, o processo de busca também retorna um conjunto vazio. Neste caso, não existe elementos que estão dentro do intervalo de busca, pois, os nós folha da ORTree, contém um único elemento do conjunto de dados, logo, se nenhum nó está totalmente contido no intervalo, é por que

nenhum elemento está. Como o processo de busca pode retornar alguns falsos positivos, é preciso comparar todos os elementos retornados, com o elemento s_q e, somente aqueles cuja distância seja menor ou igual a ξ, podem ser de fato retornados pela busca. Contudo, neste trabalho, o foco está nas partições geradas pela ORTree e, por isso, o resultado esperado de uma busca executada sobre a ORTree é um conjunto de nó da árvore, que representam as partições.

Algoritmo 13 – Busca Por Abrangência na ORTree

Entrada: Intervalo de busca (RD_ξ), o nó raiz da ORTree.
Saída: Conjunto de nós da ORTree (R).

```
1: R ← ∅
2: R ← Query(RD_ξ, ORTree.root, 0)
3: retorna R
4: função QUERY(Intervalo, node, dim)
5:     se RD_ξ[dim] ∩ node.limits = ∅ então
6:         retorna ∅
7:     fim se
8:     se node.limits ⊆ RD_ξ[dim] então
9:         se node.next <> NULL então
10:            retorna Query(RD_ξ, node.next, dim + 1)
11:        senão
12:            retorna node
13:        fim se
14:    fim se
15:    R ← Query(RD_ξ, node.right, dim)
16:    R ← R ∪ Query(RD_ξ, node.left, dim)
17:    retorna R
18: fim função
```

Para responder uma busca aos k-vizinhos mais próximos utilizando a ORTree, é necessário utilizar alguma estratégia para estimar o limiar de dissimilaridade (ξ_k), como por exemplo as discutidas na seção 2.5.2. A partir disso, a busca pode ser executada da mesma forma que uma Rq.

Quanto à complexidade de tempo do algoritmo de busca, temos o custo de construção do intervalo de busca e o custo de percorrer os nós da árvore. O Custo do algoritmo 12 é $O(\mathscr{D})$, pois é preciso iterar sobre as $\mathscr{D}$-dimensões das *omni*-coordenadas do elementos s_q. Já o custo de percorrer a ORTree é o mesmo da RT, neste caso, $O(log^{\mathscr{D}}(n))$.

5.2.3 Inserção e Remoção

O processo de inserção de elementos na ORTree segue o mesmo definido pela RT, para os casos em que o elemento a ser inserido está dentro dos intervalos cobertos pela árvore. Caso o elemento esteja fora desses intervalos, muito provavelmente, ele interfere em algumas das regras de seleção de elementos pivôs da Omni-Techinique, o que pode degenerar a ORTree. Neste caso, o procedimento mais correto é desconstruir e construir a ORTree considerando que esse novo

elemento já faz parte do conjunto de dados. Já para o caso de remoção de elementos, o maior problema que pode ocorrer é árvore ficar desbalanceada, sendo que alguns nós podem conter mais elementos que outros. Considerando que os intervalos dos nós da árvore são definidos durante a construção, remover um elemento que define o início ou o fim de um nó não implica diretamente em destruir o nó.

Neste trabalho, o foco está em criar a ORTree e nas buscas que podem ser realizadas por ela. Por conta disso, foi considerado que os conjuntos de dados que irão utilizar a estrutura sofrem poucas ou nenhuma alteração.

5.2.4 ORtree - Busca por Similaridade com Diversidade

A ORTree foi desenvolvida e utilizada neste trabalho por dois principais motivos: (*i*) ela permite particionar os dados previamente, ou seja, todo o custo de particionar os dados está na construção da estrutura; (*ii*) além disso, é possível realizar buscas por similaridade dentro da estrutura e, a partir disso, considerar somente as partições geradas que possuem elementos retornados pela busca. Deste modo, é possível selecionar os elementos candidatos de forma particionada, o que evita um possível custo adicional de ter de particionar os dados depois de selecioná-los.

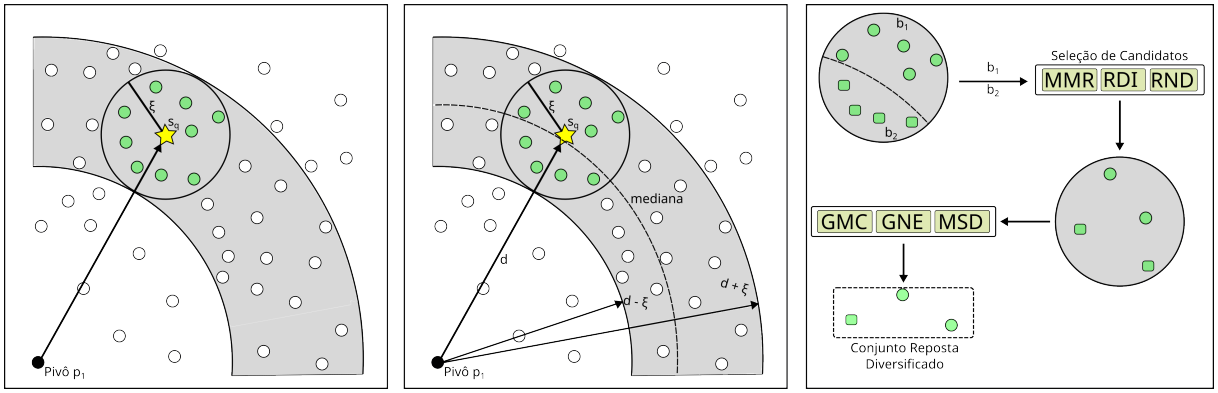
Considerando, como os dados são organizados pela ORTree, os elementos dentro de um nó da árvore tendem a serem mais similares entre si, do que em relação aos elementos que estão em outro nó[1]. Deste modo, também pode-se dizer que a diversidade entre nós (ou entre os elementos de cada nó) é maior do que a diversidade que seria encontrada em único nó. Assim sendo, uma boa abordagem de filtragem de candidatos pode usufruir dessas características dos nós da ORTree e, extrair de cada nó os seus elementos mais diversos e, então combinar os elementos selecionados em um único conjunto candidato. Neste sentido, é possível garantir que vai existir diversidade entre os elementos candidatos. Contudo, o objetivo deste trabalho é executar buscas por similaridade com diversidade, logo, é necessário garantir a similaridade entre os elementos candidatos e o elemento central de consulta (s_q). Por conta disso, as abordagens desenvolvidas partem sempre do resultado de uma busca por similaridade ($R_q(s_q, \xi)$) executada sobre a ORTree, o que já garante que os nós analisados possuem os elementos 'mais' similares a s_q. Outro ponto importante é que as abordagens utilizadas sempre levam em consideração a similaridade a s_q e a diversidade entre os elementos de cada nó.

A Figura 11 ilustra o processo de filtragem de candidatos utilizando a ORTree. Na Figura 11 (a) temos o intervalo de busca(RD_ξ) definido por uma busca por abrangência ($R_q(s_q, \xi)$). Já na Figura 11 (b), o intervalo de busca é particionado, considerando os nós da ORTree. Neste caso, ao executar a busca na ORTree, os elementos cobertos pelo intervalo de busca estão em nós diferentes, o que gera uma partição no intervalo de busca e nos elementos retornados. Dada

[1] essa afirmação pode não ser verdade para os elementos que estão próximos dos limites que definem o nó, neste caso, esses elementos podem ser muito similares a elementos que estão em outro nó.

as partições geradas, a Figura 11 (c) ilustra a filtragem de candidatos. Inicialmente, em cada partição, é aplicado um algoritmo de seleção de candidatos, que extrai m elementos candidatos de cada partição. A partir dai, os elementos selecionados de cada partição, são unidos em um único conjunto de candidatos que é enviado para um algoritmo de diversificação, que então gera o conjunto-resposta diversificado. Neste trabalho foram desenvolvidas três abordagens de seleção de candidatos: RT_MMR, RT_RDI e RT_RnD, que seguem o processo descrito anteriormente e serão melhor apresentadas abaixo.

Figura 11 – Processo de seleção de candidatos utilizando as partições criadas pela ORTree, considerando um único pivô em um espaço Euclidiano Bidimensional. (a) Uma busca por abrangência definida por $R_q(s_q, \xi)$. (b) O intervalo de busca particionado (linha pontilhada) pela ORTree, gerando dois subintervalos. (c) Considerando as partições geradas (b_1 e b_2), uma abordagem de seleção de candidatos é aplicada em cada partição. A partir disso, os elementos selecionados de cada partição são unidos e enviados ao algoritmo de diversificação, que gera o conjunto resposta diversificado.



Fonte: Elaborada pelo autor.

A abordagem *Range Tree MMR* (**RT_MMR**) utiliza o algoritmo MMR para selecionar os elementos candidatos. O MMR é mais rápido do que os algoritmos GMC, GNE e MSD, mas tende a gerar conjuntos resultados de qualidade inferior. Contudo, o MMR segue a mesma estrategia de diversificação desses algoritmos, selecionando os elementos considerando a preferencia de diversidade (λ). Considerando que o MMR é utilizado para selecionar elementos de alguns nós da ORTree, espera-se que a qualidade dos elementos selecionados (de cada nó) seja melhor, do que aquela que seria encontrada se o MMR fosse aplicado sobre todos os elementos. Essa expectativa parte da ideia de que quanto menos elementos o algoritmo (heurística) precisa analisar, menores são as chances de que o algoritmo encontre um máximo local ao invés de um máximo global, o que, neste caso, pode impactar positivamente a qualidade dos elementos candidatos selecionados.

Já a abordagem *Range Tree RnD* (**RT_RnD**) combina a seleção de nós da ORTree com a seleção aleatória de elementos. Essa abordagem toma vantagem da velocidade da seleção aleatória e da limitação de espaço gerada pelos nós da ORTree. A seleção aleatória não depende

de nenhum tipo de comparação entre elementos, logo, ela tende a ser mais rápida do que as demais abordagens discutidas. Contudo, por conta da falta de comparação entre elementos, não é possível garantir que os elementos selecionados são similares a s_q e/ou possuem diversidade entre si. Podemos utilizar a ORTree para contornar esses possíveis problemas. Considerando que os elementos a serem selecionados estão em diferentes nós da ORTree, as chances da seleção aleatória selecionar elementos que possuem uma maior diversidade entre si aumentam, pois a distância entre elementos de nós diferentes tende a ser maior. Além disso, considerando que o número de elementos em um nó é menor que o número de elementos em todos os nós a serem retornados, as chances de elementos similares a s_q serem selecionados aleatoriamente também aumentam. Dito isso, essa abordagem possui mais chances, em relação a abordagem original, de gerar bons conjuntos candidatos.

Na abordagem *Range Tree RDI* (**RT_RDI**), os elementos candidatos são selecionados considerando o conceito de diversificação de resultados baseada em influência (*RDI*). Por se tratar de uma abordagem de diversificação baseada em cobertura, *RDI* é mais rápida que o MMR e um pouco mais lenta que RnD, mas seleciona elementos que são similares a s_q e que são diversos entre si. Contudo, essa abordagem tende a analisar muito mais elementos do que as duas abordagens anteriores, o que pode torna-la mais lenta. Para contornar esse problema, foi decidido utilizar *RDI* nos nós retornados pela ORTree. Dessa forma, é possível reduzir o número de comparações entre elementos, o que torna a seleção mais rápida. Além disso, como essa abordagem é aplicada sobre os nós retornados por uma busca na ORTree, o espaço de busca da abordagem é definido por $R_q(s_q, \xi)$, ou seja, o número de elementos analisados pela abordagem é $|R_q(s_q, \xi)|$, o que, geralmente, tende a ser bem menor do que o total de elementos da base de dados.

Todas as abordagens desenvolvidas tentam selecionar m elementos candidatos de cada nó retornado por uma busca executada na ORTree. Caso o nó tenha menos que m elementos, todos os elementos do nó são selecionados como candidatos. Além disso, para todas as abordagens é esperado que o número de elementos candidatos selecionados seja menor que o número definido pelo espaço de busca original($R_q(s_q, \xi)$).

5.3 Considerações Finais

Neste capítulo foram apresentadas as abordagens de filtragem de candidatos, baseadas em particionamento, que reduzem a quantidade de elementos analisados pelas buscas aos k-vizinhos diversos mais próximos, reduzindo assim o tempo de execução das buscas por diversidade. A abordagem desenvolvida utiliza a ORTree, uma estrutura de dados baseada na Range Tree e na Omni-Technique, que utiliza *omni*-coordenadas para particionar e organizar os dados. A partir da ORTree são definidos três métodos de seleção de candidatos que utilizam as partições geradas pela ORTree, para gerar conjuntos candidatos com o menor número possível de elementos, que

maximizem a função objetivo utilizada pelas buscas.

Nó próximo Capítulo são apresentados os resultados dos experimentos realizados, em relação a tempo de busca, quantidade de elementos selecionados e qualidade.

CAPÍTULO

6

EXPERIMENTOS

6.1 Materiais e Métodos

A avaliação das abordagens desenvolvidas neste trabalho utilizou diferentes conjuntos de dados, tanto reais quanto sintéticos. Os principais objetivos na seleção destes conjuntos de dados foi avaliar o comportamento das abordagens desenvolvidas e os principais concorrentes da literatura, em relação a diferentes valores de dimensionalidade, dimensão fractal (número de pivôs) e cardinalidade. Para isso foram escolhidos quatro conjuntos de dados: US_Cities, Nasa (FIGUEROA; NAVARRO; CHÁVEZ, 2007), Corel[1] e Synthetic[2]. A Tabela 3 apresenta as informações sobre cada dos conjunto de dados e os parâmetros de consulta utilizados em cada um deles. Para cada conjunto é indicado: o nome, a cardinalidade ($|S|$), a dimensionalidade (D), o número de pivôs utilizados ($|\mathscr{P}|$), a função de distância de similaridade e de diversidade (δ), o limiar de dissimilaridade utilizado (ξ), a média de elementos retornados pelas buscas por similaridade ($|R|$) e uma breve descrição do conjunto de dados. De cada um desses conjuntos de dados, foram extraídos 50 elementos de forma aleatória, para serem centros de consulta (s_q).

Todos os experimentos realizados partem, de alguma forma, do resultado de uma busca por abrangência ($Rq(s_q, \xi)$). Deste modo, o parâmetro ξ foi definido de forma que o número de elementos retornados em cada conjunto de dados seja aproximadamente o mesmo.

As configurações do computador utilizado para realizar os experimentos foram: 1 processador Intel Corei5 10400F com 16GB de memória RAM e disco rígido de 500GB. O sistema operacional utilizado foi o Ubuntu 18.04 LTS.

As abordagens desenvolvidas (RT_MMR, RT_RDI e RT_RnD), foram comparadas com as seguintes abordagens da literatura:

---

[1] Amostra extraída de <https://archive.ics.uci.edu/ml/datasets/corel+image+features>, acessado em: 24/08/2022.

[2] https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make$_blobs.html$

Tabela 3 – Características dos conjuntos de dados utilizados.

Nome	$\|S\|$	d	$\mathscr{D}$	$\|\mathscr{P}\|$	δ	ξ	$\|R\|$	Descrição
US_Cities	25,374	2	1.62	2	L_2	{2.0,**4.0**}	{560,1860}	Coordenadas Geográficas de cidades Americanas
Nasa	40,150	20	2.63	3	L_2	{0.6, **0.7**}	{716,1514}	Conjunto de vetores de características gerados a partir de imagens da NASA
Corel	20,000	9	4.8	5	L_2	{1.9, **2.3**}	{680, 1572}	Conjunto de vetores de características de imagens comuns.
Synthetic	50,000	5	3.75	4	L_1	{0.5, **0.6**}	{680, 1572}	Conjunto de dados sintético com elementos gerados aleatoriamente utilizando o sklearn.

- *Rq*: Utiliza todos os elementos retornados por uma busca por abrangência (*Rq*) definida por s_q e ξ.
- RC-Index: Para garantir que os elementos retornados pela abordagem estejam dentro do mesmo intervalo das demais abordagens, o RC-Index foi implementado utilizando a ORTree ao invés da Range Tree.
- CT: Foi ajustado para usar somente os elementos retornados por uma busca *Rq*, pois o custo de construir a estrutura para todo conjunto de dados é muito alto.
- RnD: Utiliza 30% dos elementos retornados por uma busca *Rq*, selecionados aleatoriamente.
- RDI: Utiliza um subconjunto dos elementos do conjunto de dados, retornados pela abordagem de diversificação baseada em influência (RDI). O número de elementos solicitados para abordagem é igual ao número de elementos retornados pela *Rq*, neste caso, $|R_q(s_q, r)|$.

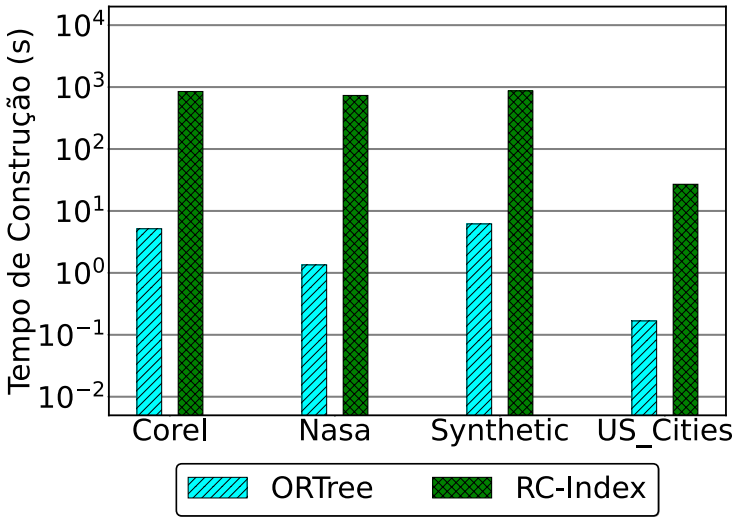
Para validar os elementos candidatos retornados por todas as abordagens de seleção, foram utilizados os algoritmos GMC, GNE e MSD. Para cada um dos algoritmos foram utilizados os seguintes valores de parâmetros (em negrito estão os valores padrão): $\lambda = \{0.3, \mathbf{0.5}, 0.7\}$ e $k = \{10, \mathbf{20}, 30, 40\}$. Para abordagens RT_MMR, RT_RDI e RT_RnD foi definido que o número de elementos que devem ser selecionados de cada partição (nó) da ORTree deve ser igual ao número de elementos a serem retornados pela busca (k), neste caso: $m = k$.

6.2 Tempo de Construção das Árvores

A Figura 12 mostra o tempo de construção da ORTree em comparação ao tempo de construção do RC-Index. Como pode ser visto na figura, para todas as bases de dados, o tempo de construção da ORTree é bem menor que o do RC-Index. Essa diferença de tempo se deve

a dois fatores: (*i*) a complexidade de construção da ORTree ($O(nlog^{\mathscr{D}}(n))$) é, pelo menos, um grau menor que a complexidade do RC-Index ($O(\gamma^6 nlog^{\mathscr{D}+1}(n))$), o que já torna o RC-Index muito mais lento. Além disso, (*ii*) a construção da ORTree não depende de cálculos de distância, enquanto que o RC-Index constrói uma Cover Tree, que precisa fazer vários cálculos de distância, para construir cada um dos seus nós, o que aumenta muito o seu tempo de construção.

Figura 12 – Tempo de construção da ORTree e do RC-Index em escala logarítmica.

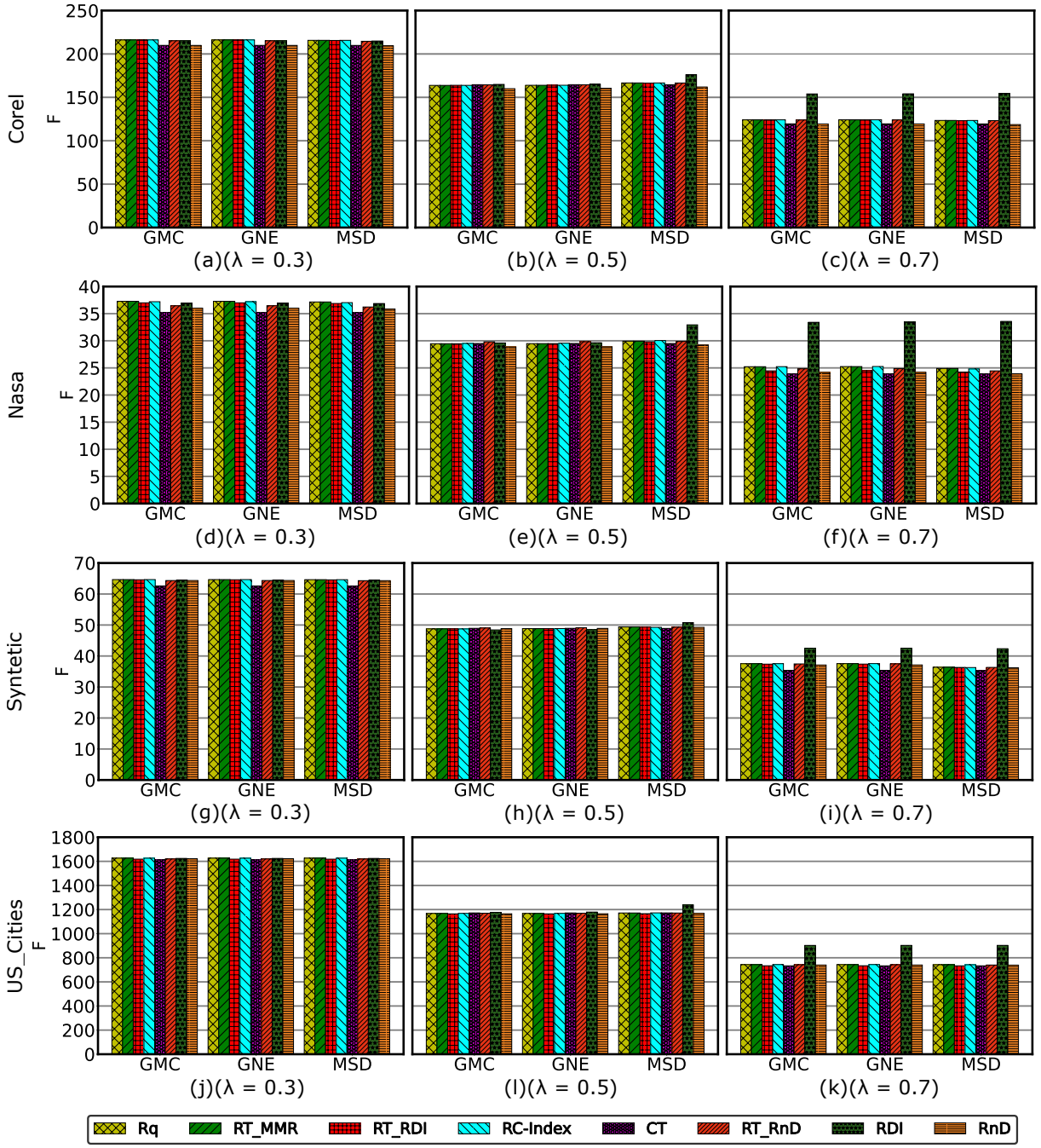


6.3 Experimentos de Qualidade

A Figura 13 mostra os valores retornados pela função objetivo de diversidade (Eq. 3.3) para cada uma das abordagens testadas utilizando os algoritmos GMC, GNE e MSD. As Figuras 13 (a, b, c) mostram os resultados para a base de dados Corel ($\xi = 2.3$). As Figuras 13 (d, e, f) mostram os resultados para a base de dados Nasa ($\xi = 0.7$). Já as Figuras 13 (g, h, i) mostram os resultados para a base dados Synthetic ($\xi = 0.6$) e, por fim, as Figuras 13 (j, l, k) mostram os resultados para a base dados US_Cities ($\xi = 4$).

Para a base de dados Corel muitas abordagens empataram em $\lambda = 0.3$. Contudo, as abordagens CT e RnD tiveram os piores resultados em comparação com as demais abordagens. Para $\lambda = 0.5$, as abordagens empataram novamente, entretanto, RnD novamente, teve os piores resultados e a abordagem RDI alcançou melhores resultados, melhores inclusive do que aqueles gerados pela abordagem tradicional *Rq*. Em $\lambda = 0.7$, o comportamento se mantém o mesmo, CT e RnD tiveram os piores resultados e RDI alcançou resultados melhores do que todas as outras abordagens, inclusive *Rq*. Para a base de dados Nasa, em $\lambda = 0.3$, muitas abordagens empataram. Contudo, as abordagens CT, RT_RnD e RnD alcançaram os piores resultados, sendo CT a com piores resultados. Para $\lambda = 0.5$, novamente, muitas abordagens empataram, contudo, as abordagens RT_RnD e RDI alcançaram os melhores resultados, RT_RnD gerou os melhores resultados para os algoritmos GMC e GNE, enquanto RDI gerou os melhores resultados com o algoritmo MSD. Em $\lambda = 0.7$, temos que, CT e RnD alcançaram os piores resultados e, RDI os

Figura 13 – Qualidade dos resultados de acordo com a função objetivo de diversidade(F), para as bases de dados Corel (a, b e c) para $\xi = 2.3$, Nasa (d, e e f) para $\xi = 0.7$, Synthetic (g, h e i) para $\xi = 0.6$ e US_Cities (j, l e k) para $\xi = 4$.

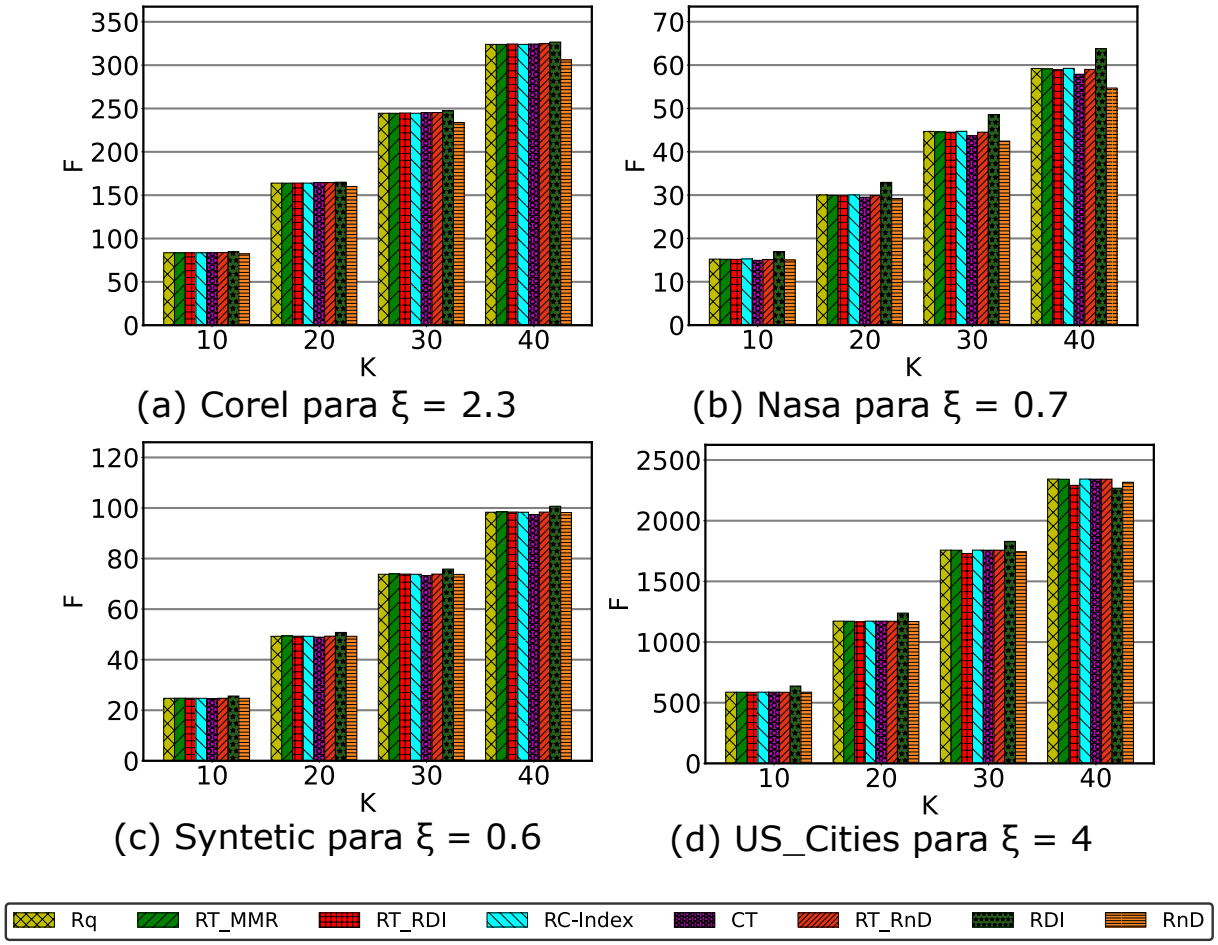


melhores resultados. Neste caso, a abordagem RT_RDI teve uma piora nos resultados, ficando um pouco abaixo das abordagens *Rq*, RT_MMR e RC-Index. Na base de dados Synthetic, para $\lambda = 0.3$, todas as abordagens empataram, exceto CT, que ficou abaixo das demais abordagens. Em $\lambda = 0.5$, houve um empate entre as abordagens, sendo que a abordagem RDI ficou um pouco abaixo para os algoritmos GMC e GNE, mas apresentou melhores resultados para o algoritmo MSD. Já abordagem CT, teve um resultado inverso, manteve a qualidade dos resultados para

os algoritmos GMC e GNE, mas apresentou uma piora nos resultados do algoritmo MSD. Já para $\lambda = 0.7$, todas as abordagens empataram, exceto CT e RDI, CT ficou abaixo das demais abordagens e RDI superou os resultados das outras abordagens. Na base de dados US_Cities, houve um empate entre todas as abordagens para $\lambda = 0.3$. Para $\lambda = 0.5$ e $\lambda = 0.7$, todas abordagens empataram, exceto abordagem RDI, que para $\lambda = 0.5$ gerou melhores resultados usando o algoritmo MSD e, em $\lambda = 0.7$, gerou melhores resultados para todos os algoritmos.

Para esse experimento, o esperado das abordagens é que elas fiquem próximas da abordagem *Rq*, pois ela é a melhor solução conhecida (*baseline*). Contudo, quanto maior o valor da função objetivo, melhor o resultado. Em todos os resultados apresentados, as abordagens RT_-MMR e RC-Index, praticamente empataram com a abordagem *Rq*. Já as abordagens CT, RT_RnD e RnD, oscilaram bastante na qualidade dos resultados, empataram com *Rq* em alguns casos, mas em outros tiveram piores resultados. Em alguns casos, a abordagem RT_RnD conseguiu superar os resultados de *Rq*. Já abordagem RDI conseguiu para na maioria dos experimentos, gerar melhores resultados que *Rq*, sendo que no pior dos casos, o resultado gerado empatou com o valor gerado por *Rq*.

Figura 14 – Qualidade dos resultados de acordo com a função objetivo de diversidade($\mathscr{F}$), variando o parâmetro k, para as bases de dados Corel (a) para $\xi = 2.3$, Nasa (b) para $\xi = 0.7$, Synthetic (c) para $\xi = 0.6$ e US_Cities (d) para $\xi = 4$.



O experimento descrito demonstra o comportamento das abordagens, considerando diferentes algoritmos e preferências de diversidade (λ). Na Figura 14 são mostrados os resultados de um outro experimento, similar ao anterior, que avalia os resultados da função objetivo de diversidade para cada uma das abordagens analisadas, utilizando o algoritmo GNE para diferentes valores de k. A Figura 14 (a) mostra os resultados para a base de dados Corel com $\xi = 2.3$. A Figura 14 (b) mostra o resultado para a base Nasa com $\xi = 0.7$. Já a Figura 14 (c) mostra os resultados para a base Synthetic com $\xi = 0.6$ e, a Figura 14 (d) mostra os resultados para a base US_Cities com $\xi = 4$.

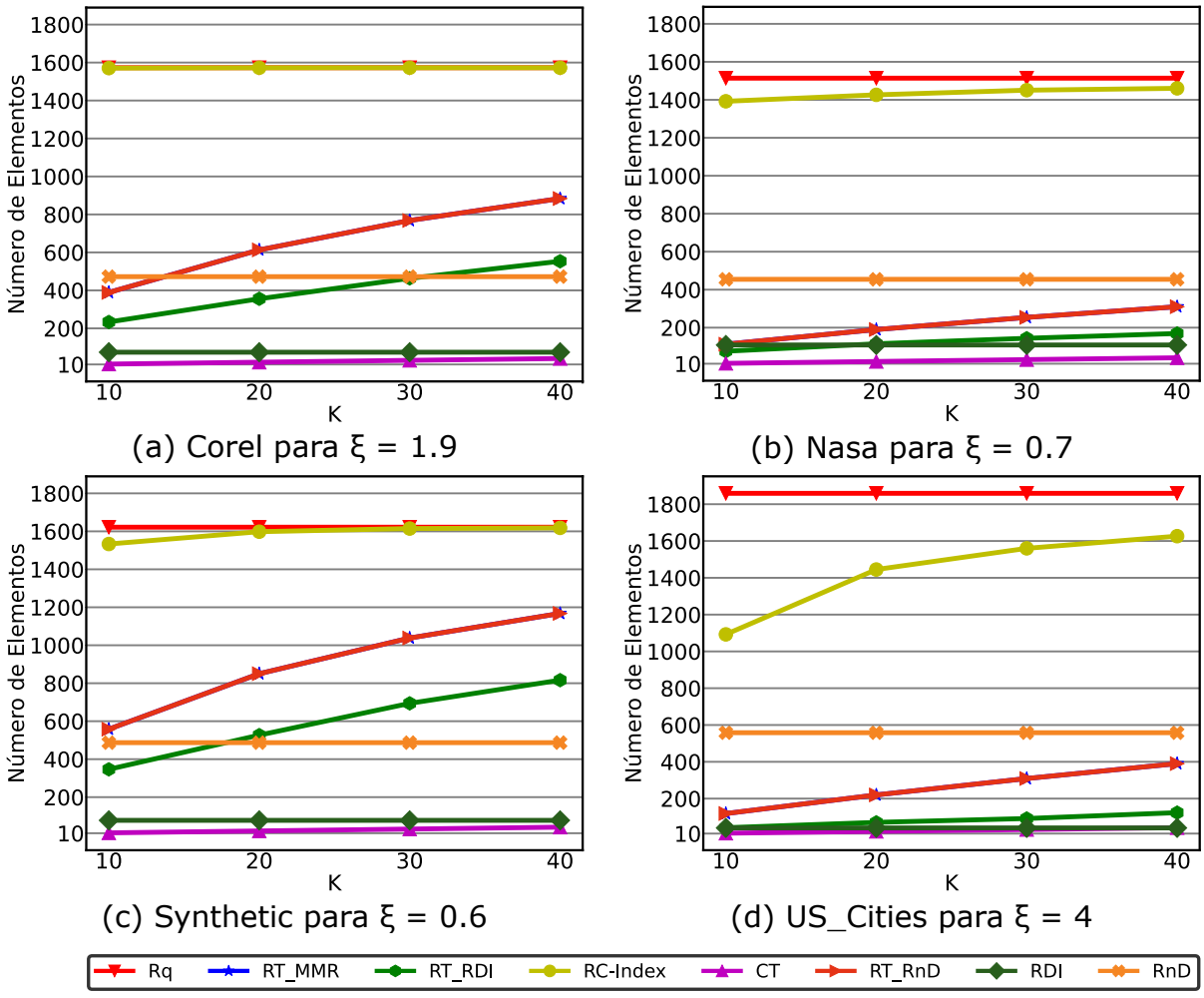
Na base de dados Corel, as abordagens seguiram, basicamente, o mesmo comportamento para todos os valores de k. RDI gerou os melhores resultados, enquanto RnD alcançou piores resultados em relação as outras abordagens. Conforme o k foi crescendo, a diferença entre RnD e as demais abordagens também foi aumentando. Para a base de dados Nasa, as abordagens *Rq*, RT_MMR, RT_RDI, RC-Index e RT_RnD empataram para todos os valores de k, enquanto que, abordagem RDI gerou os melhores resultados, e as abordagens CT e RnD geraram resultados abaixo das demais abordagens. Já na base de dados Synthetic várias abordagens empataram novamente, sendo que RDI alcançou os melhores resultados, e CT ficou um pouco abaixo. O comportamento das abordagens na base de dados US_Cities, muda um pouco em relação as demais bases de dados. Para $k = 10$ e $k = 20$ houve um empate entre todas abordagens, exceto RDI que alcançou melhores resultados. Para $k = 30$ o comportamento é quase o mesmo, entretanto, a qualidade da abordagem RT_RDI diminui um pouco em relação a *Rq*. Já para $k = 40$, muitas abordagens empataram, contudo, as abordagens RT_RDI e RDI tiveram os piores resultados, sendo RDI inferior à RT_RDI.

O fato de algumas das abordagens serem capazes de gerar resultados melhores do que a abordagem *Rq* levanta alguns pontos interessantes. No caso do RDI, a abordagem é mais exploratória e não segue a estratégia de restrição do espaço de busca. Sendo assim, ela analisa muito mais elementos do que as outras abordagens e, por conta disso, a diversidade entre os elementos retornados tende a ser maior, do que das abordagens que utilizam a restrição de espaço, tal como a *Rq*. Em relação a abordagem RT_RnD, isso se deve porque o processo de seleção/filtragem de candidatos, provavelmente, remove alguns elementos que fazem parte de soluções que são ótimos locais. Como os algoritmos de diversificação são heurísticas que nem sempre encontram a solução ótima, é possível que a solução retornada pelos algoritmos não seja um ótimo e, portanto, soluções melhores possam ser encontradas. À medida que o processo de filtragem de candidatos remove elementos do espaço de solução, provavelmente alguns ótimos locais também estão sendo removidos e, as chances de uma solução melhor ser encontrada aumenta. Portanto, o processo de filtragem de candidatos pode não apenas reduzir o tempo de execução dos algoritmos, mas também proporcionar maior qualidade.

6.4 Quantidade de Elementos Retornados

A Figura 15 mostra a média de elementos candidatos retornados por cada uma das abordagens. A Figura 15 (a) mostra os resultados para a base de dados Corel. A Figura 15 (b) os resultados para a base de dados Nasa. Já as Figuras 15 (c e d) mostram, respectivamente, os resultados para as bases de dados Synthetic e US_Cities.

Figura 15 – Número médio de elementos candidatos retornados por cada uma das abordagens analisadas. (a) Resultados para base de dados Corel. (b) Resultados para base de dados Nasa. (c) Resultados para base de dados Synthetic. (d) Resultados para base de dados US_Cities.



As abordagens *Rq* e RnD sempre selecionam um número fixo de elementos, pois em ambas abordagens o número de elementos é definido por um parâmetro fixo que não se ajusta a quantidade de elementos. Já a abordagem CT sempre retorna k elementos, o que fez essa abordagem ser a que retorna o menor número de elementos candidatos para todas as bases de dados. Para todas as bases de dados, o número de candidatos retornados pelas abordagens propostas (RT_MMR, RT_RnD e RT_RDI), assim como o RC-Index, cresce à medida que k aumenta. No entanto, em todos os casos, as abordagens recuperam menos elementos do que *Rq* e RC-Index. Para a base de dados Corel(Figura 15 (a)), RDI retornou o segundo menor número de elementos, sempre retornando menos de 5% da quantidade de elementos retornados por *Rq*. Já o

RC-Index, recuperou quase o mesmo número de elementos retornados por *Rq* (isso será discutido a seguir). As abordagens propostas, apresentaram o comportamento esperado, o número de elementos retornados cresce a medida que k aumenta. Contudo, ambas abordagens recuperam bem menos elementos que *Rq*, RC-Index e, em alguns casos, menos elementos que a abordagem RnD; nestes casos, recuperando menos de 30% do número de elementos de *Rq*. Para a base de dados Nasa (Figura 15 (b)), RDI continua sendo a abordagem que na média recupera menos elementos, entretanto, para $k = \{10, 20\}$ a abordagem RT_RDI consegue recuperar menos ou a mesma quantidade de elementos. O RC-Index, recupera menos elementos que *Rq*, mas ainda continua sendo a segunda abordagem que recupera mais elementos. Em geral, todas as abordagens propostas recuperam menos elementos que *Rq* e RnD, recuperando menos do que 30% dos elementos de *Rq*. Já para a base de dados Synthetic(Figura 15 (c)), RDI e CT são abordagens que menos recuperam elementos, seguidas da abordagem RnD. O RC-Index, inicia recuperando menos elementos que *Rq* mas acaba recuperando mais ou menos o mesmo número de elementos. As abordagens propostas, continuam recuperando menos elementos que abordagem *Rq*, mas, recuperam mais elementos do que RDI e RnD. Por fim, a Figura 15 (d) apresenta os resultados para a base de dados US_Cities. Para essa base de dados, os resultados são similares aos da base de dados Nasa. RDI e, RT_RDI são uma das abordagens que menos recuperam elementos, sendo que, a quantidade de elementos retornados pelas duas é bem próxima. O RC-Index, recupera menos elementos que *Rq*, mas continua sendo de longe a segunda abordagem que mais recupera elementos. As demais abordagens propostas recuperaram menos de 30% dos elementos de *Rq*, recuperando assim menos elementos que RC-Index e RnD.

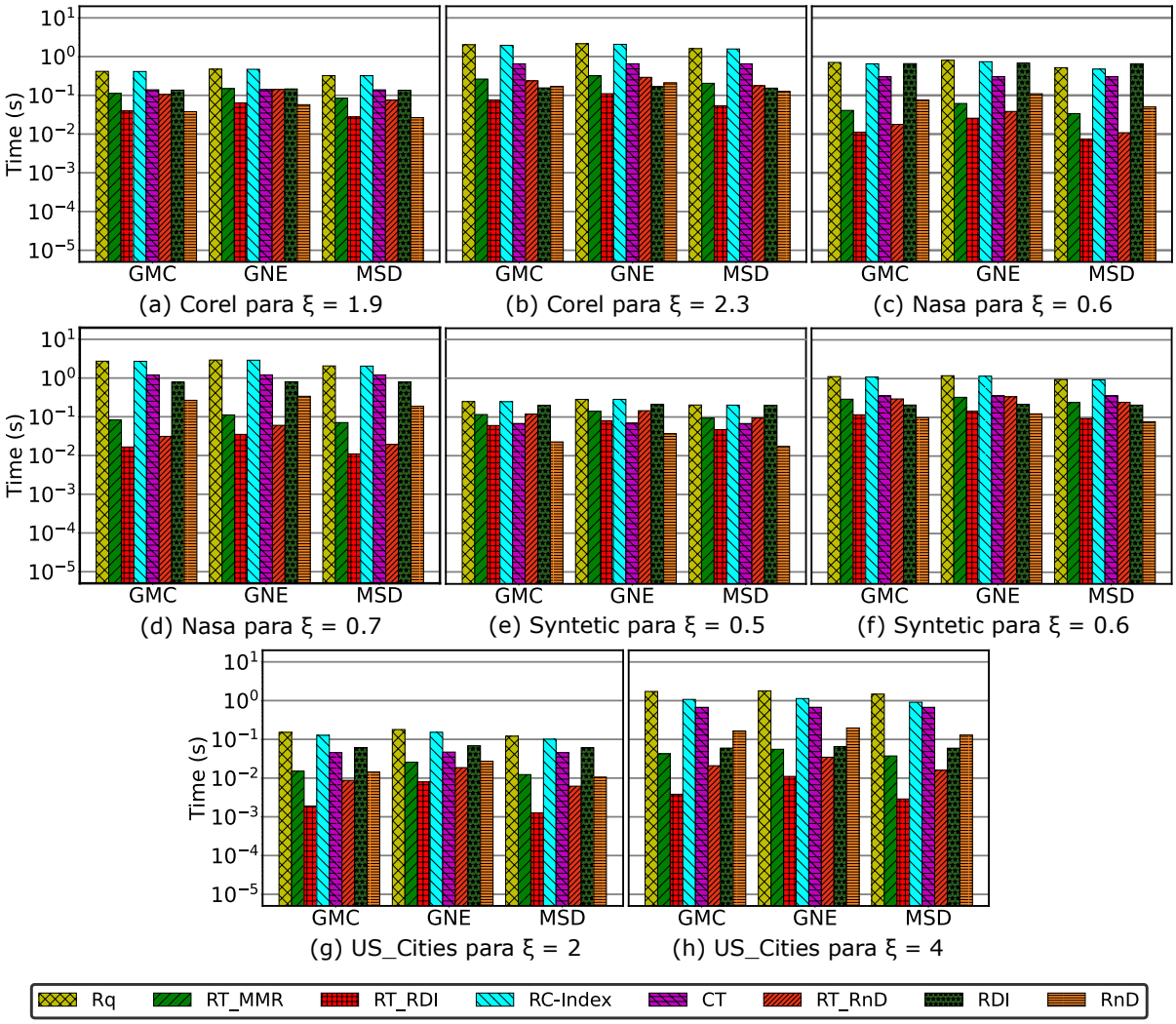
Em relação ao número de candidatos retornados pelo RC-Index, por conta da estratégia utilizada pela abordagem, de selecionar elementos que estão três níveis abaixo do primeiro nível da Cover Tree que possui k ou mais elementos, o número de elementos retornados é sempre maior que k. Além disso, dependendo da distribuição dos dados, descer três níveis pode ser suficiente para selecionar todos os elementos do nó pesquisado, o que explica por que a quantidade de elementos tende a ser próxima de *Rq*. Outro ponto que contribui para essa situação é que quanto mais pivôs, mais partições dos dados são geradas, o que implica em menos elementos nós por na ORTree (ou RT). No entanto essa mesma situação não acontece com as abordagens RT_MMR, RT_RnD e RT_RDI, pois elas sempre retornam k ou menos elementos de cada nó.

6.5 Experimentos de Tempo de Busca

A Figura 16 mostra o tempo de execução dos algoritmos de busca utilizando as abordagens analisadas, em escala logarítmica. As Figuras mostram, respectivamente, o tempo de execução para as bases de dados Corel (a e b), Nasa (c e d), Synthetic (e e d) e US_Cities(g e h). O tempo de execução usando *Rq* tende a ser maior que o tempo de execução de todas as outras abordagens, o que é esperado, devido ao grande número de elementos recuperados. Como o RC-Index recupera mais ou menos o mesmo número de elementos que *Rq*, o tempo de

execução das duas abordagens tende a ser próximo, sendo que em alguns casos, o RC-Index tem um custo um pouco maior, pois realiza mais operações. Em todos os gráficos, é possível ver que as abordagens RT_RDI e RnD oscilam entre as mais rápidas, seguidas pelas abordagens RT_RnD, RT_MMR.

Figura 16 – Tempo de execução dos algoritmos de diversificação, considerando cada uma das abordagens analisadas. (a e b) Base de dados Corel. (c e d) Base de dados Nasa. (e e f) Base de dados Synthetic. (g e h) Base de dados US_Cities.



As Figuras 16(a e b) mostram que para a base de dados Corel as abordagens RT_RDI e RnD são as mais rápidas, sendo que para $\xi = 2.3$ RT_RDI foi bem mais rápida que RnD. Em seguida, as abordagens RT_MMR, RT_RnD, RDI oscilaram sobre qual foi a terceira abordagem mais rápida. Como esperado *Rq* e o RC-Index empataram no tempo de execução e, CT foi mais rápida que as duas. Para a base de dados Nasa (Figuras 16(c e d)) as abordagens RT_RDI, RT_RnD e RT_MMR são, respectivamente, de longe as abordagens mais rápidas, sendo seguidas pelas abordagens RnD, CT e RDI. Para $\xi = 0.6$, RDI ficou com um custo muito próximo das abordagens *Rq* e RC-Index, neste caso, o custo de analisar todas as base de dados, não foi compensado pela baixa quantidade de elementos retornados. Para base de dados Synthetic (Figuras 16(e e f)), a abordagem RnD foi a mais rápida, seguida pela abordagem RT_RDI. As

abordagens RT_RnD, RT_MMR, RDI e CT, oscilam sobre qual é a mais rápida. Em $\xi = 0.5$, CT é a mais rápida, seguida por RT_RnD e RT_MMR que empatam no tempo, enquanto que RDI empata com a abordagem *Rq*. Já para $\xi = 0.6$, RDI é a mais rápida, RT_RnD e RT_MMR empatam novamente e, CT acabam sendo mais lenta do que elas. Já na base de dados US_-Cities(Figuras 16(g e h)), as abordagens propostas são as mais rápidas, sendo que RT_RDI é de longe abordagem mais rápida. Em seguida as abordagens RDI, RnD e CT são, respectivamente, as mais rápidas. Contudo o custo de CT é fica próximo do custo do RC-Index, que neste caso, é um pouco menor que o custo de *Rq*.

As abordagens CT e RDI, em alguns casos, ficam com um custo muito próximo da abordagem *Rq*. Isso acontece, principalmente, por conta das operações realizadas por cada uma dessas abordagens. CT retorna k elementos o que permite que os algoritmos de diversificação sejam executados de forma eficiente. Entretanto, essa abordagem constrói uma Cover Tree com custo $O(m^2)$ sobre os elementos retornados por *Rq*, e por conta disso, o seu tempo de execução tende a ser mais elevado do que as demais abordagens. Já RDI, analisa muitos elementos da base de dados. Para os testes realizados, a abordagem analisou todos os elementos de todas as bases de dados, e por conta disso, o custo de tempo dessa abordagem aumenta muito, ficando atrás de outras abordagens. Contudo houve casos, onde a baixa quantidade de elementos retornados, compensa o alto número de comparações e elementos analisados por essa abordagem. Em geral os resultados de tempo mostram que as abordagens propostas sempre ganham de algumas das abordagens da literatura, principalmente a abordagem RT_RDI, que na base de dados Nasa conseguiu ser 97% mais rápida que a abordagem tradicional *Rq*.

6.6 Experimentos de Escalabilidade

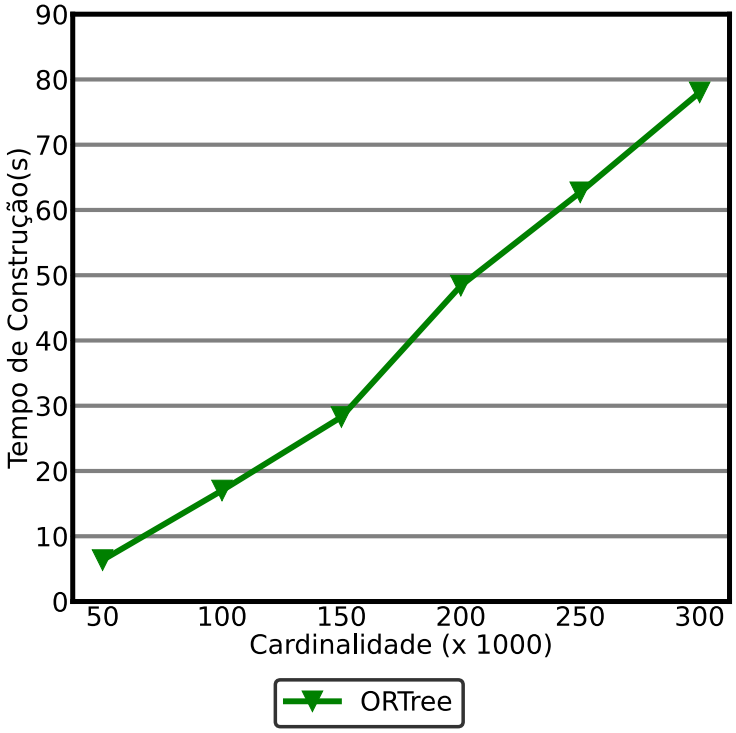
Para testar a escalabilidade das abordagens desenvolvidas, foram realizados testes utilizando a base de dados Synthetic, com um número maior de elementos (maior cardinalidade) e foram realizadas buscas com um ξ maior. Consequentemente, o tempo de construção e de busca das abordagens aumenta, bem como a quantidade de elementos retornados e analisados.

6.6.1 Escalabilidade - Tempo de Construção

Para testar a escalabilidade do tempo de construção da ORTree foram gerados seis bases de dados baseados no conjunto Synthetic, neste caso, aumentou-se a cardinalidade do conjunto de 50.000, para 300.000 elementos, aumentando quantidade em passos de 50.000 elementos.

A Figura 17 mostra os resultados desse experimento. Como esperado, o tempo de construção da ORTree segue um comportamento polilogarítmico, sendo que o tempo de construção ainda cresce em relação ao número de elementos da base de dados, mas com um custo bem menor que $O(n^2)$, mostrando assim, que a estrutura é escalável, quanto a cardinalidade da base de dados. Vale a pena ressaltar, que a complexidade de tempo e de espaço da ORTree é polilogarítmica em

Figura 17 – Tempo de Construção da ORTree, variando a cardinalidade da base de dados.



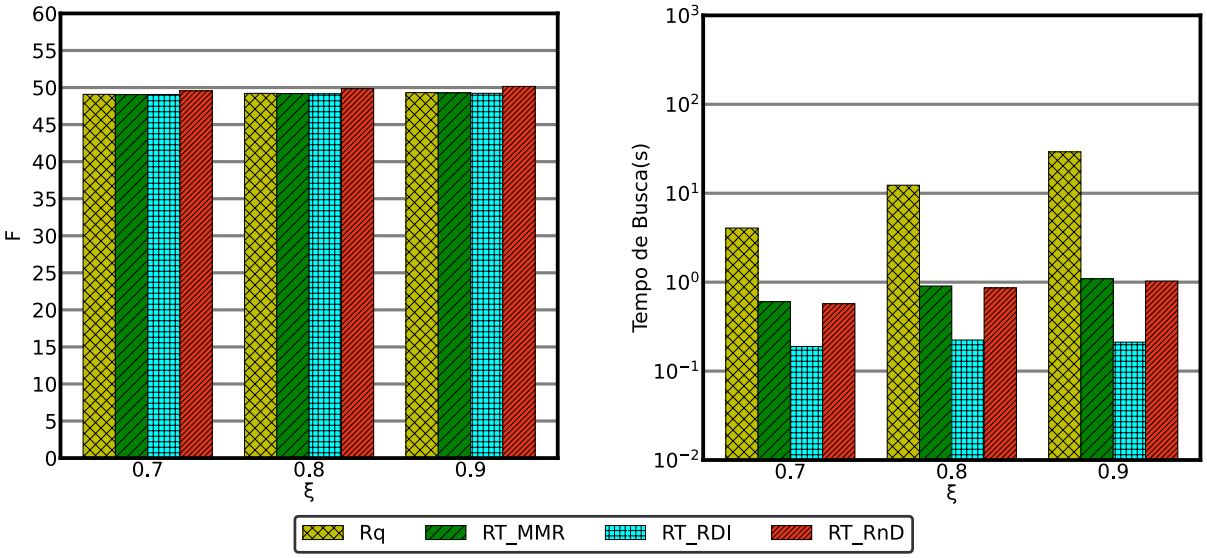
relação à dimensão fractal da base de dados. Sendo assim, a dimensão fractal do conjunto pode impactar bastante o tempo de construção.

6.6.2 Escalabilidade - Tempo de Busca e Seleção de Candidatos

Para testar o desempenho das abordagens propostas, quando a quantidade de elementos analisados aumenta, foram realizados testes utilizando o algoritmo GNE com os parâmetros: $k = 20$, $\lambda = 0.5$ e $\xi = \{0.7, 0.8, 0.9\}$. A Figura 18 mostra os resultados dos experimentos de qualidade e de tempo, utilizando esses parâmetros. Os resultados de qualidade (Figura 18(a)) mostram que as abordagens propostas são equivalentes à abordagem *Rq*, sendo que a abordagem RT_RnD gera melhores resultados que *Rq*. Já para os resultados tempo de busca (Figura 18(b)) mostram que as abordagens desenvolvidas reduzem bastante o tempo de execução, em comparação a *Rq*. Neste caso, é possível perceber que o tempo de busca da abordagem *Rq* cresce muito conforme ξ cresce, enquanto que o tempo das abordagens propostas, também aumenta mas de forma muito mais sutil e sempre a baixo de *Rq*.

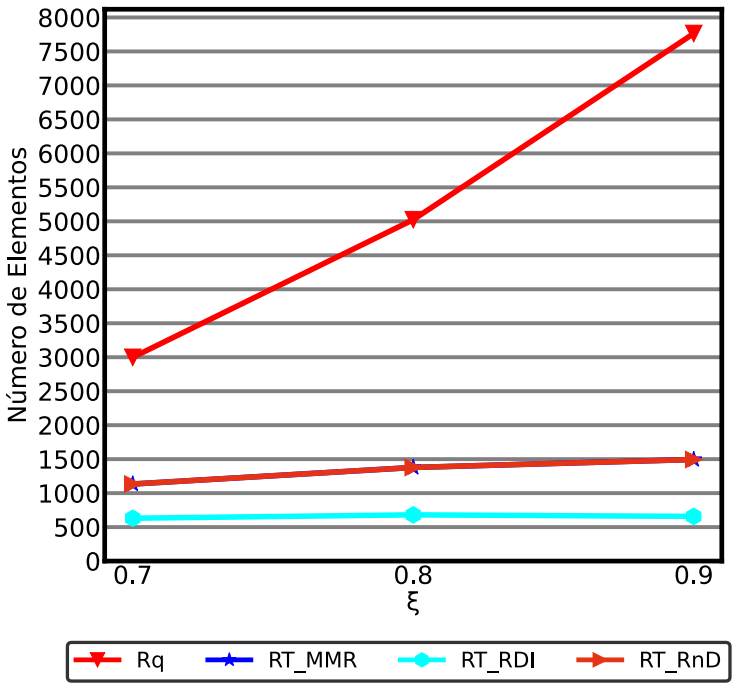
A Figura 19 mostra o número médio de elementos retornados por cada uma das abordagens propostas e da abordagem *Rq* utilizando a base de dados Synthetic com $\xi = \{0.7, 0.8, 0.9\}$. Como pode ser visto, a quantidade de elementos selecionados pelas abordagens propostas é bem menor que os selecionados pela abordagem *Rq*. Além disso, conforme ξ cresce a quantidade de elementos retornados por *Rq* cresce muito, enquanto que a quantidade de elementos retornados pelas abordagens propostas cresce muito pouco. Por exemplo, para $\xi = 0.8$ *Rq* retornou em torno de 5024 elementos, enquanto que, RT_RnD retornou 1377, uma redução de mais de 70%

Figura 18 – Qualidade dos resultados de acordo com a função objetivo de diversidade(F) e tempo de busca das abordagens desenvolvidas, para a base de dados Synthetic com $\xi = \{0.7, 0.8, 0.9\}$



no número de elementos, sendo que a qualidade dos resultados se manteve. Isso mostra que as abordagens propostas são escaláveis em relação a quantidade de elementos selecionados e, consequentemente, em relação ao tempo de busca. Além disso, a qualidade dos resultados é igual ou superior à dos elementos selecionados por *Rq*.

Figura 19 – Número de elementos candidatos retornados pelas abordagens propostas, para a base de dados Synthetic com $\xi = \{0.7, 0.8, 0.9\}$



6.7 Considerações Finais

Este capítulo apresentou o ambiente de teste utilizado nesta dissertação para avaliar as abordagens propostas, em comparação com as abordagens da literatura. Os testes realizados tinham como objetivo analisar o tempo de construção da ORTree em comparação ao RC-Index, a qualidade dos resultados gerados, o tempo de busca e quantidade de elementos retornados pelas abordagens de seleção de candidatos propostas.

Os resultados dos experimentos mostram que as abordagens propostas obtiveram bons resultados, retornando menos elementos candidatos, mantendo qualidade equivalente ou melhor do que a da abordagem tradicional. Consequentemente, o tempo de execução das buscas por similaridade com diversidade foi bastante reduzido. Além disso, os testes de escalabilidade mostram, que mesmo em situações onde a abordagem tradicional apresenta um alto custo de busca, as abordagens propostas conseguem recuperar, ainda menos elementos candidatos, com qualidade equivalente ou superior. Desde modo, pode-se concluir que a abordagem de particionamento dos dados (ORTree) e as abordagens de seleção de candidatos (RT_MMR, RT_RnD e RT_RDI) são eficientes e eficazes, em relação à seleção de elementos candidatos para o processo de diversificação de resultados.

No próximo capítulo são apresentadas as conclusões e trabalhos futuros dessa dissertação.

CAPÍTULO

7

CONCLUSÕES E TRABALHOS FUTUROS

Para evitar problemas como a redução de expressividade, vários trabalhos exploraram a inclusão do conceito de diversidade às buscas por similaridade tradicionais. Contudo, ao incluírem o processo de diversificação nas buscas por similaridade, a quantidade de comparações e elementos analisados pelas buscas aumentou drasticamente, principalmente, para abordagem de diversificação baseada em novidade, em que o processo de diversificação é baseado em um processo de otimização. Por conta disso, apesar de aumentar a qualidade dos resultados retornados ao usuário, as buscas por similaridade com diversidade podem apresentar problemas de desempenho. Para contornar esse problema, muitos trabalhos na literatura utilizam a estratégia de pré-seleção de candidatos. Neste caso, um subconjunto de elementos da base de dados é selecionado para serem utilizados pela busca. Como a quantidade de elementos é menor, o tempo de busca acaba sendo reduzido também. Contudo, apesar de ser bastante interessante, essa estratégia pode apresentar problemas quanto à quantidade e à qualidade do subconjunto selecionado. Quanto menor o conjunto, menor é a diversidade entre os elementos selecionados mas, quanto maior é o conjunto maior é o tempo da busca.

7.1 Principais Contribuições

Nessa dissertação de mestrado foi apresentada uma abordagem que permite otimizar o tempo das buscas por similaridade com diversidade, utilizando dois conceitos: particionamento de dados e seleção de candidatos. O primeiro conceito tem como objetivo particionar os elementos da base de dados em vários subconjuntos. Já o segundo conceito tenta extrair dos subconjuntos gerados, um conjunto de elementos que sejam similares ao elemento central de consulta e ao mesmo tempo diversos entre si. Deste modo, é possível selecionar elementos candidatos de forma rápida, e que possuem alta qualidade. As principais contribuições deste trabalho são:

- Desenvolvimento de uma nova estrutura de particionamento dos dados, baseada na distân-

cia entre os elementos.

- Três abordagens para selecionar rapidamente elementos candidatos, para os algoritmos de busca por similaridade com diversidade baseadas em novidade/otimização.

A abordagem de particionamento desenvolvida (ORTree) é baseada em métodos de acesso métrico e, além de particionar os dados, permite que buscas por similaridade sejam apoiadas por ela. Já as abordagens de seleção de candidatos partem do resultado de uma busca por similaridade executada sobre a ORTree, para então, selecionar em cada partição retornada pela busca um conjunto de elementos candidatos. Para garantir que os elementos selecionados possuem alto valor (qualidade), as abordagens utilizadas selecionam os elementos considerando sua similaridade em relação ao elemento central de consulta e a diversidade entre os elementos retornados.

Os experimentos realizados mostram que as abordagens desenvolvidas conseguem cumprir o seu objetivo principal, que é recuperar poucos elementos candidatos mas que possuem qualidade equivalente aos elementos recuperados pela abordagem tradicional. Os testes mostram que o tempo necessário para selecionar os candidatos e gerar as respostas diversificadas é sempre menor que a abordagem utilizada tradicionalmente. Além disso, os testes mostram que nossas abordagens são escaláveis tanto na cardinalidade da base de dados, quanto na quantidade de elementos analisados durante a busca, sendo essencialmente equivalentes na qualidade dos resultados gerados. De fato as abordagens desenvolvidas conseguem reduzir bastante o tempo de execução das buscas, sendo que em alguns casos, o custo de tempo foi reduzido em mais de 97%. Outro ponto importante, é que resultados de qualidade mostram que, em algumas situações, os elementos retornados pelas abordagens propostas conseguem melhorar as respostas geradas, maximizando ainda mais a função objetivo utilizada nas buscas.

7.2 Contribuições/Trabalhos Complementares

Além dos resultados dessa dissertação, outros trabalhos foram desenvolvidos e publicados durante o desenvolvimento da dissertação, e que são listados abaixo:

- J-EDA: A workbench for tuning similarity and diversity search parameters in content-based image retrieval, JIDM, vol. 12, no. 2, Sep. 2021.

- Analysis of ENEM's attendants between 2012 and 2017 using a clustering approach, JIDM, vol. 11, no. 2, Feb. 2021.

- ORTree: Tuning Diversified Similarity Queries by Means of Data Partitioning, ADBIS 2022. Lecture Notes in Computer Science, vol 13389. Springer, Cham.

7.3 Proposta de Trabalhos Futuros

Neste trabalho foram utilizados diferentes conceitos relacionados à buscas por similaridade com diversidade. Contudo existem alguns pontos que podem ser futuramente explorados e que podem expandir as contribuições deste trabalho.

7.3.1 Testes utilizando subconjuntos de elementos pivô

A estrutura desenvolvida para particionar os dados é baseada na estratégia de indexação baseada em pivôs. Na abordagem utilizada neste trabalho, o número de pivôs é definido com base na dimensão fractal. Contudo, podem ser utilizados subconjuntos dos pivôs que seriam originalmente selecionados pela própria abordagem. Neste caso, o objetivo seria analisar os possíveis impactos da escolha dos elementos pivôs nos resultados gerados pelas abordagens de seleção de candidatos. Especificamente propõe-se realizar os mesmos experimentos realizados neste trabalho, só que utilizando todos os possíveis subconjuntos de pivôs definidos pela Omni-Technique, explorando outras técnicas de escolha dos pivôs.

7.3.2 Outras estruturas de particionamento de dados

A Omni Range Tree (ORTree) foi desenvolvida tendo com base a *Range Tree*. A apesar dos bons resultados apresentados, existem várias outras estruturas (Por exemplo: R-tree e Quad-tree) que podem ser utilizadas como base para a estratégia de particionar os dados considerando a distância entre os elementos. Além disso, a RT tem uma complexidade de espaço de $O(nlog^d(n))$, o que pode ser restritivo, mesmo considerando a redução de dimensionalidade da Omni-Technique. Logo, explorar outras estruturas que permitam particionar os dados, mas que possuem complexidades (de tempo de construção, busca e espaço) diferentes, pode melhor a aplicabilidade das abordagens desenvolvidas. Além disso, dependendo das características da estrutura utilizada, a qualidade dos resultados pode ser melhor do que aqueles alcançados até este momento.

7.3.3 ORTree como estrutura de busca por similaridade

Neste trabalho, a ORTree foi utilizada somente como uma parte do processo de busca por similaridade com diversidade. Contudo, ela também pode ser utilizada para responder as buscas por similaridade convencionais. Neste caso, é preciso analisar os resultados da estrutura sobre esse tipo de busca e, se for necessário, realizar os ajustes necessários na estrutura para garantir que os custos das buscas por similaridade sejam de fato reduzidos. Lembrando que as operações mais custosas para uma busca por similaridade convencional são diferentes das de uma busca por similaridade com diversidade, sendo que o número de comparações e acessos a disco são um dos principais custos envolvidos na busca.

7.3.4 Novas métricas de avaliação de diversidade

Durante o desenvolvimento deste trabalho, foram percebidos alguns pontos importantes sobre as buscas por similaridade com diversidade baseadas em novidade. Os principais pontos encontrados e que pretende-se estudar futuramente são:

- Fator de aproximação das heurísticas: Boa parte das heurísticas utilizadas para resolver o problema de diversificação não possuem uma demonstração do quão próximas do valor ótimo elas estão. Logo, não é possível definir com precisão se os resultados atualmente gerados pelas heurísticas são de fato os melhores ou se existe espaço para melhora, tanto em relação à complexidade de tempo, quanto à qualidade das respostas.

- Avaliações baseadas no *feedback* do usuário: A maioria das métricas utilizadas para avaliar os resultados das buscas por similaridade com diversidade são de alguma forma baseados na distribuição dos dados e nas relações de distância dos elementos. Nenhuma das métricas utilizadas leva em conta o *feedback* direto do usuário sobre a qualidade dos resultados gerados. Neste caso, propõe-se analisar os resultados dos algoritmos de diversificação, considerando o *feedback* de especialistas do domínio dos dados em que as buscas estão sendo utilizadas.

- Uso de funções de distância diferentes para similaridade e diversidade: Apesar de ser matematicamente possível, poucos trabalhos exploram de fato, utilizar um descritor para medir a similaridade e outro para medir a diversidade dos elementos retornados. Considerando que cada descritor pode definir um espaço de busca completamente diferente do outro, pode ser que existam vantagens em utilizar descritores diferentes. Por exemplo, um descritor consegue representar melhor a relação de similaridade desejada pelo usuário, enquanto que, outro descritor consegue representar melhor a relação diversidade desejada. Contudo, desenvolver e validar esse tipo de abordagem provavelmente varia de acordo com o caso de uso e, provavelmente, do apoio e da experiência de especialistas de domínio.

REFERÊNCIAS

ABBAR, S.; AMER-YAHIA, S.; INDYK, P.; MAHABADI, S.; VARADARAJAN, K. R. Diverse near neighbor problem. In: **Proceedings of the Twenty-Ninth Annual Symposium on Computational Geometry**. New York, NY, USA: Association for Computing Machinery, 2013. (SoCG '13), p. 207–214. ISBN 9781450320313. Disponível em: <https://doi.org/10.1145/2462356.2462401>. Citado na página 45.

AGRAWAL, R.; GOLLAPUDI, S.; HALVERSON, A.; IEONG, S. Diversifying search results. In: **Proceedings of the Second ACM International Conference on Web Search and Data Mining**. New York, NY, USA: Association for Computing Machinery, 2009. (WSDM '09), p. 5–14. ISBN 9781605583907. Disponível em: <https://doi.org/10.1145/1498759.1498766>. Citado nas páginas 26 e 42.

AVALHAIS, L. P. S. **Transformação de espaços métricos otimizando a recuperação de imagens por conteúdo e avaliação por análise visual**. Dissertação Mestrado — Universidade de São Paulo, 2012. Citado nas páginas 25 e 31.

BÊDO, M. V. N. **Modelos de custo e estatísticas para consultas por similaridade**. Tese (Doutorado) — Universidade de São Paulo, 2017. Citado nas páginas 25, 32, 33, 34, 35 e 38.

BERG, M. D.; KREVELD, M. V.; OVERMARS, M.; SCHWARZKOPF, O. Computational geometry. In: **Computational geometry**. [S.l.]: Springer, 1997. p. 1–17. Citado na página 38.

BEYGELZIMER, A.; KAKADE, S.; LANGFORD, J. Cover trees for nearest neighbor. In: **Proceedings of the 23rd International Conference on Machine Learning**. New York, NY, USA: Association for Computing Machinery, 2006. (ICML '06), p. 97–104. ISBN 1595933832. Disponível em: <https://doi.org/10.1145/1143844.1143857>. Citado nas páginas 32 e 56.

CARBONELL, J.; GOLDSTEIN, J. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In: ACM. **Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval**. [S.l.], 1998. p. 335–336. Citado nas páginas 27, 43 e 45.

CAZZOLATO, M. T. **Conquering knowledge from images: improving image mining with region-based analysis and associated information**. Tese (Doutorado) — Universidade de São Paulo, 2019. Citado na página 33.

CIACCIA, P.; PATELLA, M.; ZEZULA, P. M-tree: An efficient access method for similarity search in metric spaces. In: JARKE, M.; CAREY, M. J.; DITTRICH, K. R.; LOCHOVSKY, F. H.; LOUCOPOULOS, P.; JEUSFELD, M. A. (Ed.). **VLDB'97, Proceedings of 23rd International Conference on Very Large Data Bases, August 25-29, 1997, Athens, Greece**. Morgan Kaufmann, 1997. p. 426–435. Disponível em: <http://www.vldb.org/conf/1997/P426.PDF>. Citado nas páginas 32 e 36.

DEZA, M. M.; DEZA, E. **Encyclopedia of distances**. [S.l.: s.n.], 2009. 1–590 p. ISBN 9783642002335. Citado na página 32.

DIAS, R. L.; BUENO, R.; RIBEIRO, M. X. Reducing the complexity of k-nearest diverse neighbor queries in medical image datasets through fractal analysis. In: **Proceedings of CBMS 2013 - 26th IEEE International Symposium on Computer-Based Medical Systems**. [S.l.: s.n.], 2013. p. 101–106. ISBN 9781479910533. Citado nas páginas 35, 38 e 44.

DROSOU, M.; JAGADISH, H.; PITOURA, E.; STOYANOVICH, J. c. **Big data**, Mary Ann Liebert, v. 5, n. 2, p. 73–84, 2017. Citado nas páginas 41, 42, 43 e 44.

____. Diversity in big data: A review. **Big data**, Mary Ann Liebert, Inc. 140 Huguenot Street, 3rd Floor New Rochelle, NY 10801 USA, v. 5, n. 2, p. 73–84, 2017. Citado nas páginas 31 e 42.

DROSOU, M.; PITOURA, E. Poikilo: A tool for evaluating the results of diversification models and algorithms. **Proc. VLDB Endow.**, VLDB Endowment, v. 6, n. 12, p. 1246–1249, aug 2013. ISSN 2150-8097. Disponível em: <https://doi.org/10.14778/2536274.2536287>. Citado na página 26.

____. Diverse set selection over dynamic data. **IEEE Transactions on Knowledge and Data Engineering**, IEEE, v. 26, n. 5, p. 1102–1116, 2014. ISSN 10414347. Citado nas páginas 26, 56 e 60.

FALOUTSOS, C.; SEEGER, B.; TRAINA, A.; TRAINA, C. Spatial join selectivity using power laws. **SIGMOD Rec.**, Association for Computing Machinery, New York, NY, USA, v. 29, n. 2, p. 177–188, maio 2000. ISSN 0163-5808. Disponível em: <https://doi.org/10.1145/335191.335412>. Citado na página 38.

FIGUEROA, K.; NAVARRO, G.; CHÁVEZ, E. **Metric Spaces Library**. 2007. Available at http://www.sisap.org/Metric_Space_Library.html. Citado na página 73.

GOLLAPUDI, S.; SHARMA, A. An axiomatic approach for result diversification. **WWW'09 - Proceedings of the 18th International World Wide Web Conference**, p. 381–390, 2009. Citado nas páginas 41, 42, 43 e 46.

HARITSA, J. R. **The KNDN Problem: A Quest for Unity in Diversity**. [S.l.], 2009. Citado nas páginas 41, 49 e 50.

JR., C. T.; TRAINA, A. J. M.; SEEGER, B.; FALOUTSOS, C. Slim-trees: High performance metric trees minimizing overlap between nodes. In: ZANIOLO, C.; LOCKEMANN, P. C.; SCHOLL, M. H.; GRUST, T. (Ed.). **Advances in Database Technology - EDBT 2000, 7th International Conference on Extending Database Technology, Konstanz, Germany, March 27-31, 2000, Proceedings**. Springer, 2000. (Lecture Notes in Computer Science, v. 1777), p. 51–65. Disponível em: <https://doi.org/10.1007/3-540-46439-5_4>. Citado nas páginas 32 e 36.

KOLLAR, T. Fast Nearest Neighbors. **Most**, p. 1–10, 2006. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.163.6892{&}rep=rep1{&}ty>. Citado na página 56.

KUNAVER, M.; POŽRL, T. Diversity in recommender systems–a survey. **Knowledge-based systems**, Elsevier, v. 123, p. 154–162, 2017. Citado na página 26.

LEUKEN, R. H. van; GARCIA, L.; OLIVARES, X.; ZWOL, R. van. Visual diversification of image search results. In: **Proceedings of the 18th International Conference on World Wide Web**. New York, NY, USA: Association for Computing Machinery, 2009. (WWW '09), p. 341–350. ISBN 9781605584874. Disponível em: <https://doi.org/10.1145/1526709.1526756>. Citado na página 46.

LOPES, C. R.; SANTOS, L. F. D.; JASBICK, D. L.; OLIVEIRA, D. de; BEDO, M. An empirical assessment of quality metrics for diversified similarity searching. **JIDM**, v. 12, n. 3, Oct. 2021. Citado nas páginas 42 e 49.

MARTINEZ, J. M.; KOENEN, R.; PEREIRA, F. Mpeg-7: the generic multimedia content description standard, part 1. **IEEE MultiMedia**, v. 9, n. 2, p. 78–87, Apr 2002. ISSN 1070-986X. Citado na página 33.

NOVAES, J. V. d. O. N.; et. al. J-EDA: A diversified similarity workbench for content-based image retrieval. In: SBC. **SBBD Demos**. [S.l.], 2019. Citado nas páginas 25, 26, 32, 33, 41 e 49.

NOVAES, J. V. de O.; BENEDITO, C. C. da S.; SANTOS, L. F. D. Avaliando diferentes implementações do descritor de cor dominante. **Revista Mundi Engenharia, Tecnologia e Gestão (ISSN: 2525-4782)**, v. 4, n. 5, 2019. Citado na página 33.

NOVAES, J. V. O.; SANTOS, L. F. D.; CARVALHO, L. O.; OLIVEIRA, D. de; BEDO, M. V. N.; TRAINA, A. J. M.; JR., C. T. J-eda: A workbench for tuning similarity and diversity search parameters in content-based image retrieval. **JIDM**, v. 12, Sep. 2021. Citado nas páginas 27, 28, 31 e 42.

RAVI, S. S.; ROSENKRANTZ, D. J.; TAYI, G. K. Heuristic and special case algorithms for dispersion problems. **Operations Research**, INFORMS, v. 42, n. 2, p. 299–310, 1994. Citado na página 56.

SANTOS, L.; et. al. Exploring Diversified Similarity with Kundaha. In: ACM. **CIKM**. [S.l.], 2018. p. 1903–1906. Citado nas páginas 25, 26 e 41.

SANTOS, L. F.; OLIVEIRA, W. D.; FERREIRA, M. R.; TRAINA, A. J.; TRAINA, C. Parameter-free and domain-independent similarity search with diversity. **ACM International Conference Proceeding Series**, p. 1–12, 2013. Citado nas páginas 27, 42, 49, 50, 51 e 52.

SANTOS, L. F. D. **Similaridade em big data**. Tese (Doutorado) — Universidade de São Paulo, 2017. Citado nas páginas 25, 26, 31, 32, 34, 35, 36, 41, 42, 43, 44, 45, 46 e 51.

SANTOS, L. F. D.; OLIVEIRA, W. D.; CARVALHO, L. O.; FERREIRA, M. R. P.; TRAINA, A. J. M.; TRAINA, C. Combine-and-conquer: Improving the diversity in similarity search through influence sampling. **Proceedings of the ACM Symposium on Applied Computing**, v. 13-17-Apri, p. 994–999, 2015. Citado nas páginas 27, 44, 58 e 60.

SKOPAL, T.; DOHNAL, V.; BATKO, M.; ZEZULA, P. Distinct nearest neighbors queries for similarity search in very large multimedia databases. **International Conference on Information and Knowledge Management, Proceedings**, p. 11–14, 2009. Citado nas páginas 26, 27, 41, 42, 49 e 50.

SOUZA, J. A. d. **Agrupamento de dados complexos para apoiar consultas por similaridade com tratamento de restrições**. Tese (Doutorado) — Universidade de São Paulo, 2018. Citado nas páginas 25, 31, 32, 33 e 36.

TRAINA, C.; FILHO, R. F.; TRAINA, A. J.; VIEIRA, M. R.; FALOUTSOS, C. The Omni-family of all-purpose access methods: A simple and effective way to make similarity search more efficient. **VLDB Journal**, v. 16, n. 4, p. 483–505, 2007. ISSN 10668888. Citado nas páginas 32, 36 e 37.

VIEIRA, M. R.; RAZENTE, H. L.; BARIONI, M. C. N.; HADJIELEFTHERIOU, M.; SRIVASTAVA, D.; JR., C. T.; TSOTRAS, V. J. On query result diversification. In: ABITEBOUL, S.; BÖHM, K.; KOCH, C.; TAN, K. (Ed.). **Proceedings of the 27th International Conference on Data Engineering, ICDE 2011, April 11-16, 2011, Hannover, Germany**. IEEE Computer Society, 2011. p. 1163–1174. Disponível em: <https://doi.org/10.1109/ICDE.2011.5767846>. Citado nas páginas 26, 27, 28, 41, 42, 43, 44, 45, 46, 47, 48 e 49.

VIEIRA, M. R.; TRAINA, C.; TRAINA, A. J.; ARANTES, A.; FALOUTSOS, C. Boosting k-nearest neighbor queries estimating suitable query radii. **Proceedings of the International Conference on Scientific and Statistical Database Management, SSDBM**, n. August, 2007. ISSN 10993371. Citado nas páginas 35, 38 e 42.

WANG, Y.; MELIOU, A.; MIKLAU, G. RCIndex: Diversifying answers to range queries. **Proceedings of the VLDB Endowment**, v. 11, n. 7, p. 773–786, 2018. ISSN 21508097. Citado nas páginas 27, 38, 40, 44, 45, 57, 58 e 60.

YIANILOS, P. N. Data structures and algorithms for nearest neighbor search in general metric spaces. In: **Soda**. [S.l.: s.n.], 1993. v. 93, n. 194, p. 311–21. Citado na página 36.

YU, C.; LAKSHMANAN, L.; AMER-YAHIA, S. It takes variety to make a world: Diversification in recommender systems. In: . [S.l.: s.n.], 2009. p. 368–378. Citado na página 48.

ZEZULA, P.; AMATO, G.; DOHNAL, V.; BATKO, M. **Similarity search: the metric space approach**. [S.l.]: Springer Science & Business Media, 2006. v. 32. Citado na página 36.

____. **Similarity Search: The Metric Space Approach**. [S.l.]: Springer, 2010. v. 2. Citado nas páginas 31 e 32.

ZHENG, K.; WANG, H.; QI, Z.; LI, J.; GAO, H. A survey of query result diversification. **Knowledge and Information Systems**, Springer London, v. 51, n. 1, 2017. ISSN 02193116. Citado nas páginas 26, 28, 32, 41, 42, 44 e 45.


ICMC
USP
SÃO CARLOS