UNIVERSIDADE FEDERAL FLUMINENSE

DANIEL LEONARDO JASBICK

Buscas por Similaridade Diversificada com o Apoio de Índices Métricos

NITERÓI

2021

UNIVERSIDADE FEDERAL FLUMINENSE

DANIEL LEONARDO JASBICK

Buscas por Similaridade Diversificada com o Apoio de Índices Métricos

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Mestre em Computação. Área de concentração: Ciência da Computação.

Orientador:
Prof. D.Sc. DANIEL CARDOSO MORAES DE OLIVEIRA

Co-orientador:
Prof. D.Sc. MARCOS VINÍCIUS NAVES BÊDO

NITERÓI

2021

Ficha catalográfica automática - SDC/BEE
Gerada com informações fornecidas pelo autor

J39b Jasbick, Daniel Leonardo
Buscas por Similaridade Diversificada com o Apoio de Índices Métricos / Daniel Leonardo Jasbick ; Daniel Cardoso Moraes de Oliveira, orientador ; Marcos Vinícius Naves Bêdo, coorientador. Niterói, 2021.
94 p. : il.

Dissertação (mestrado)-Universidade Federal Fluminense, Niterói, 2021.

DOI: http://dx.doi.org/10.22409/PGC.2021.m.15461838766

1. Buscas por similaridade. 2. Diversificação de resultados. 3. Espaços Métricos. 4. Dimensionalidade Intrínseca Local. 5. Produção intelectual. I. Oliveira, Daniel Cardoso Moraes de, orientador. II. Bêdo, Marcos Vinícius Naves, coorientador. III. Universidade Federal Fluminense. Instituto de Computação. IV. Título.

CDD -

Bibliotecário responsável: Debora do Nascimento - CRB7/6368

DANIEL LEONARDO JASBICK

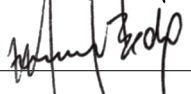
Buscas por Similaridade Diversificada com o Apoio de Índices Métricos

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Mestre em Computação. Área de concentração: Ciência da Computação.

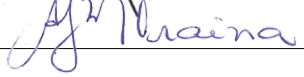
Aprovada em Maio de 2021.

BANCA EXAMINADORA

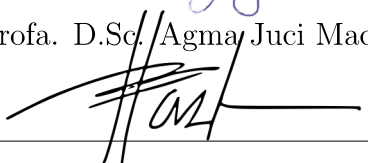
Prof. D.Sc. Daniel Cardoso Moraes de Oliveira - Orientador, IC/UFF



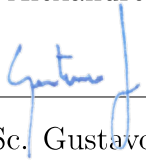
Prof. D.Sc. Marcos Vinícius Naves Bêdo - Co-Orientador, INFES/UFF



Profa. D.Sc. Agma Juci Machado Traina, ICMC/USP



Prof. D.Sc. Alexandre Plastino de Carvalho, IC/UFF



Prof. D.Sc. Gustavo Silva Semaan, INFES/UFF

Niterói

2021

Agradecimentos

Agradeço a Deus por direcionar minhas escolhas.

Ao Instituto de Computação - IC/UFF e ao Instituto do Noroeste Fluminense de Educação Superior - INFES/UFF, sua direção, administração, técnicos, terceirizados e corpo docente por oportunizar a conclusão deste trabalho.

Aos meus orientadores Marcos Vinícius Naves Bêdo e Daniel Cardoso Moraes de Oliveira pelo imenso apoio, incentivo e zelo pela qualidade desse trabalho. Um agradecimento também ao Prof. Lucio Fernandes Dutra Santos, pela troca de ideias com relação ao algoritmo BRID_k.

Aos meus familiares, em especial minha mãe, e aos meus amigos(as), em especial a Maria Luiza Falci, pelo apoio transmitido em todo o processo de realização deste trabalho.

Em geral, a todos que direta ou indiretamente fizeram parte da minha pós-graduação.

Resumo

Executar buscas em grandes quantidades de dados é um desafio, tanto do ponto de vista de tempo de execução quanto da qualidade semântica dos resultados, sendo necessário mais do que os operadores de identidade e ordem para uma recuperação eficaz e eficiente. Operadores por similaridade baseados em distância e modelados com a Teoria de Espaços Métricos permitem melhorar a capacidade semântica das consultas, mas são afetados pela cardinalidade, densidade e dimensionalidade dos dados, implicando em uma possível perda do discernimento das distâncias entre os objetos. A adição de um critério de diversidade permite que objetos de diferentes coleções sejam explorados e retornados, melhorando a percepção de proximidade e pluralidade a um custo adicional da aplicação do novo critério. Portanto, otimizações a serem consideradas pelos operadores de busca incluem acoplar diversidade à similaridade e o uso de estruturas de indexação. A Hipótese de Pesquisa desse trabalho é que *(i)* uma rotina de busca por similaridade diversificada construída sobre índices métricos pode melhorar expressivamente o desempenho desse tipo de consulta, e *(ii)* essa rotina pode melhorar a discernibilidade de objetos em espaços de alta dimensionalidade. A primeira contribuição dessa dissertação é o algoritmo *diversity browsing*, que estende a estratégia de busca incremental *distance browsing* para dar suporte ao critério de *Influência* com o apoio de um índice métrico. Assim, são propostos quatro novos lemmas que permitem criar regras para descartar partições do espaço de busca que não satisfaçam simultaneamente os critérios *(i)* de proximidade para o objeto consultado e *(ii)* Influência para os elementos no conjunto resposta. A primeira parte da Hipótese de Pesquisa é corroborada com uma avaliação numérica e experimental que mostra que o *diversity browsing* pode ser acoplado a índices métricos para aumentar o desempenho de buscas por similaridade diversificada. A segunda contribuição desse trabalho é a aplicação do conceito dimensionalidade intrínseca local (LID) junto com o *diversity browsing* para mostrar a melhor capacidade semântica de consultas por similaridade diversificada na exploração de espaços de alta dimensionalidade. Uma avaliação numérica e experimental indica que buscas com o *diversity browsing* se tornam, proporcionalmente, mais econômicas em espaços de consulta com maior LID. Os resultados também apontam que o *diversity browsing* é propenso a encontrar mais vizinhos diversificados em espaços de maior LID, devido ao aumento da cobertura de regiões Influenciadas em espaços de menor LID. A parametrização de índices métricos é relevante para a maior parte das buscas com o *diversity browsing*, sendo que os experimentos mostram que os pivôs escolhidos pelo critério que maximiza a variância de distâncias são os mais adequados. Por último, o *diversity browsing* pode ser usado para explorar visualmente a proximidade dos dados em espaços de alta dimensionalidade com diferentes valores de LID, corroborando a segunda parte da Hipótese de Pesquisa.

Palavras-chave: Busca por similaridade, diversificação de resultados, Espaços Métricos, dimensionalidade intrínseca local, concentração de distâncias

Abstract

Querying large databases is still a challenge from both computational efficiency and search quality viewpoints. Since identity and order operators are insufficient for solving those issues, distance-based similarity operators designed after the Metric Spaces Model are used to enhance the search semantics and performance. However, they are impacted by data cardinality, density, and dimensionality and may be unable to distinguish between close and far objects. The adding of a diversity criterion enables the exploration and retrieval of objects from different data collections, which enhances the meaning of vicinity and plurality at the cost of applying yet another search criterion. Therefore, search operator optimizations must include the injection of diversity into similarity and the adoption of metric indexes. This study hypothesis is *(i)* an indexed diversified similarity search routine can boost the performance of those types of searches, and *(ii)* such a routine can also improve the recognition of neighbor objects in high-dimensional spaces. The initial contribution of this dissertation is the indexed *diversity browsing* algorithm, which extends the incremental *distance browsing* approach to support the diversity-based Influence criterion. The algorithm relies on four new lemmas to discard search space partitions that do not fit the criteria of *(i)* proximity to the search object and *(ii)* Influence to the result set elements simultaneously. Numerical and experimental analyses indicate the algorithm validates the first part of the hypothesis as *diversity browsing* coupled to metric indexes substantially boosted the performance of diversified similarity searches. The second contribution of this study is combining *diversity browsing* and local intrinsic dimensionality (LID) to illustrate the advantages of exploring high-dimensional spaces with diversified similarity. A detailed experimental evaluation pinpoints *diversity browsing* searches are more economical than similarity-only searches for queries within high LID spaces. Results also show *diversity browsing* usually finds more diversified neighbors within high LID spaces due to the rapid increase in the coverage of Influenced regions within lower LID spaces. The experiments also provide shreds of evidence that indicate the fine-tuning of metric indexes is still a relevant issue, and distance-based maximal variance pivots are one of the most suitable choices. Finally, *diversity browsing* enables a visual high-dimensional data exploration by proximity according to distinct LID values, which confirms the second part of this study hypothesis.

Keywords:Similarity search, result diversification, Metric Spaces, local intrinsic dimensionality, distance concentration.

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

$\mathcal{A}$	Conjunto diversificado de uma consulta.
BRID_k	Algoritmo de busca por similaridade diversificada.
	Better Results with Influence Diversification
`CVX`	Métodos de escolha de pivôs com critério do fecho convexo.
$\mathcal{D}$	Dimensionalidade intrínseca global de um conjunto de dados.
$f_{div}(\mathcal{R}, o_q)$	Função de diversidade de $\mathcal{R}$ em relação ao objeto o_q.
GMC	Algoritmo de busca por similaridade diversificada
	Greedy Marginal Contribution.
GNE	Algoritmo de busca por similaridade diversificada
	Greedy Randomized with Neighborhood Expansion.
$\ddot{I}_{o_q,o_i}$	Conjunto de Influência Forte para os objetos $\langle o_q, o_i \rangle$.
$\mathcal{M} = \langle \mathbb{O}, \delta \rangle$	Espaço Métrico.
`MAX`	Métodos de escolha de pivôs com critério da variância máxima.
MMC	Algoritmo de busca por similaridade diversificada
	Maximum Marginal Contribution.
MMR	Algoritmo de busca por similaridade diversificada
	Maximal Marginal Relevance.
$\mathbb{N}$	Domínio dos números naturais.
$\mathbb{O}$	Domínio de dados.
$\emptyset$	Conjunto vazio.
$\mathcal{O}$	Conjunto $\mathcal{O}$.
$\mathcal{O}_{esq}, \mathcal{O}_{dir}$	Conjunto à esquerda e direita de uma partição VP-Tree.
$\mathcal{P}$	Conjunto de pivôs $\mathcal{P} \subseteq \mathbb{O}$.
$p \in \mathcal{O}$	Objeto pivô no conjunto $\mathcal{O}$.
PCA	Análise de componentes principais.
$\mathcal{R}$	Conjunto resposta de uma consulta.
$\mathcal{R}', \mathcal{R}''$	Conjuntos resposta parciais de uma consulta.
$\mathbb{R}_+$	Domínio dos números reais positivos.
$Rg(o_q, \xi)$	Consulta por abrangência.

`RND`	Métodos de escolha de pivôs aleatório.
r-DisC	Algoritmo de busca por similaridade diversificada r-DisC.
k	Número de vizinhos.
k-$NN(o_q, k)$	Consulta por vizinhança.
LID	Dimensionalidade Intrínseca Local.
L_ρ	Família de distâncias Minkowski.
$o_i, o_j \in \mathbb{O}$	Objetos no domínio $\mathbb{O}$.
$o_q \in \mathbb{O}$	Objeto de consulta no domínio $\mathbb{O}$.
VP-Tree	Índice métrico *Vantage-point Tree*.
VR	Variância Relativa de uma distribuição de distâncias.
α	Grau de aleatoriedade na seleção de objetos.
δ	Função de distância métrica.
δ_{div}	Função de diversidade como função de distância métrica.
δ_{max}	Distância máxima de uma partição a um objeto de consulta.
δ_{min}	Distância mínima de uma partição a um objeto de consulta.
δ_{sim}	Função de similaridade como função de distância métrica.
$\delta(o_q, o_k)$	Distância do objeto o_q ao seu k-ésimo vizinho o_k.
λ	Combinação linear de distância e diversidade.
μ	Mediana entre distâncias.
ξ	Raio de uma consulta por abrangência.
$\langle p, \xi_p \rangle$	Bola fechada no espaço métrico contido em p com raio ξ_p.

Sumário

Capítulo 1

Introdução

1.1 Motivação

A quantidade de dados textuais e multimídia produzidos e armazenados em formato digital cresce exponencialmente há décadas, resultando em uma gigantesca quantidade de informação "represada". Realizar consultas sobre esses dados é um crescente desafio tanto do ponto de vista da obtenção de resultados em tempo hábil, o que requer o desenvolvimento de estruturas de dados e algoritmos computacionalmente eficientes, quanto do ponto vista semântico, o que depende da modelagem adequada de operadores de consulta capazes de recuperar padrões e tendências imersos nos dados consultados [39, 59, 61].

Por exemplo, suponha que queira-se recuperar, dentre os dados represados, todos os registros associados ao cidadão brasileiro cujo Cadastro de Pessoa Física (`CPF`) é conhecido por `123`. Para essa busca, o operador de filtragem baseado em identidade "`CPF = 123`" apresenta uma alta capacidade semântica para capturar o padrão esperado. Esse operador também é eficientemente implementado por algoritmos que se beneficiam de estruturas de índice similares à B-Tree [9, 39]. Por outro lado, suponha que queira-se encontrar, pelo menos, 100 fotos de uma base de dados de criminosos condenados que contenham rostos similares a de um dado retrato falado feito às autoridades policiais. Nesse caso, não é possível aplicar o operador por identidade, pois espera-se recuperar dados que sejam próximos, mas não necessariamente idênticos.

Uma alternativa viável para esse caso é o uso de comparadores por similaridade, que são modelados pela Teoria de Espaços Métricos [18, 42, 55, 102], onde quanto mais *afastados* dois objetos estão, mais eles são considerados *dissimilares*. A proximidade entre dois objetos é medida por uma função de distância, que representa a semântica do padrão

Figura 1.1: Consultas por similaridade sobre um conjunto de fotografias de representações artísticas para uma vizinhança $k = 5$. (a) k-NN e (b) k-NN diversificado.

esperado para a consulta [26]. O operador de filtragem baseado em similaridade dos k-Vizinhos mais Próximos (k-NN) permite recuperar as $k = 100$ fotos mais próximas a um dado retrato falado de acordo com uma função de distância intuitiva à compreensão visual humana, *e.g.*, Euclideana [55]. Assim como no caso do operador de identidade, o operador de busca k-NN pode ser eficientemente implementado por algoritmos que se beneficiam de estruturas de índices métricos, como a VP-Tree e a Omni-Tree [20, 21, 43, 53, 88, 97].

A capacidade semântica de se obter respostas significativas com o uso de operadores por similaridade pode ser afetada pela cardinalidade dos dados represados, assim como suas densidade e dimensionalidade. A cardinalidade é a quantidade de entradas nos dados represados e, portanto, afeta os algoritmos de busca em função de suas complexidades computacionais [25, 97]. Já a densidade dos dados afeta a seletividade dos operadores de filtragem, que podem se tornar incapazes de reter entradas na execução da busca [8, 39]. Por último, a dimensionalidade refere-se a quantidade de atributos que são usados para descrever cada entrada nos dados represados antes da aplicação do operador de filtragem [1, 7]. Em particular, operadores por similaridade tendem a perder a capacidade semântica de discernir entre objetos "próximos" e "afastados" em conjuntos densos e de alta dimensionalidade devido ao fenômeno de concentração de distâncias [10, 36, 65], o que implica não apenas na perda da seletividade e no consequente desgaste da complexidade computacional da busca [64, 66], mas também que os conjuntos-resposta provavelmente conterão elementos muito próximos entre si [25, 28, 103, 104].

Para ilustrar essa limitação semântica, considere uma consulta k-NN a ser executada por um estudante de artes em um grande repositório de imagens[1], com o objetivo de en-

[1] `https://www.google.com/search?q=the+scream+munch`.

contrar fotografias e impressões artísticas similares ao famoso quadro "O Grito" de Edvard Munch[2]. Ao contrário das consultas anteriores, que foram eficientemente resolvidas por operadores de identidade e similaridade, essa busca tem semântica mais exploratória [40], de forma que não apenas deseja-se filtrar elementos similares a um dado objeto de consulta (o quadro), mas também recuperar aqueles que tenham algum grau de diferença entre suas impressões artísticas, uma qualidade que não necessariamente pode ser expressa em termos taxativos de uma ou múltiplas categorias previamente discretizadas. Dentro de um repositório massivo e de alta dimensionalidade de fotografias, uma busca k-NN tenderá a recuperar fotografias não só similares ao quadro da consulta, mas também similares entre si, frustrando a expectativa de análise exploratória da busca – Figura 1.1(a).

A adição de um critério de diversidade junto à similaridade é uma alternativa para garantir que diferentes coleções do conjunto de dados explorado estejam representadas por proximidade na resposta [76, 104]. Ainda para a consulta exemplo do quadro "O Grito", um operador de filtragem baseado em similaridade diversificada, que considera tanto a similaridade do objeto de consulta aos dados represados quanto a dissimilaridade entre os elementos do conjunto resposta, permite retornar não apenas a mesma pintura com diferentes esboços, mas também fotografias com estilos e impressões similares, como ilustrado na Figura 1.1(b). De uma perspectiva semântica hierárquica, um operador por similaridade diversificada deve "afrouxar" as condições de um operador por similaridade, assim como o operador por similaridade "relaxa" os requisitos estritos do operador de identidade. A definição do operador por similaridade diversificada ainda é um ponto em aberto na literatura [28, 104], podendo ser implementado de acordo com três estratégias:

1. Métodos baseados em distância [12, 67, 70],
2. Métodos baseados em novidade [15, 17, 56, 91],
3. Métodos baseados em cobertura [28, 47, 76, 79]

Estratégias das duas primeiras categorias são fortemente baseadas em conceitos de otimização em duas fases [28,103]. Tipicamente, um subconjunto de elementos candidatos é selecionado, por exemplo, tomando por base o resultado de uma busca k-NN tradicional com um número de vizinhos muito superior à cardinalidade k solicitada, sobre o qual o resultado final é construído de acordo com alguma função objetivo a ser maximizada. Por outro lado, estratégias baseadas em cobertura fazem a separação de elementos candidatos

[2] `https://github.com/divSearch`.

ao conjunto resposta em uma única passada dentro do processo de avaliação individual de cada entrada, o que permite que essa categoria de métodos possa ser diretamente embutida em rotinas de busca de baixo nível de forma muito parecida com a que já ocorre para os operadores de filtragem por identidade e similaridade [51, 55].

Essa dissertação explora essa característica de execução de consultas em uma única fase por meio dos métodos de cobertura, e visa acoplá-los diretamente na execução do operador de consulta por similaridade diversificada. Métodos de cobertura podem utilizar um limiar estático ou dinâmico de dissimilaridade para descartar elementos do conjunto resposta [33,47]. Em particular, limiares dinâmicos baseados no conceito de Influência [76, 79] apresentam como vantagem uma ponderação automática da dissimilaridade entre os elementos na resposta, independente de parâmetros externos do usuário e que aumenta proporcionalmente à similaridade dos elementos recuperados, seguindo a distribuição de distâncias entre os dados represados e o objeto consultado.

Na prática, limiares baseados em Influência usam uma razão do inverso da similaridade entre o objeto consultado e os elementos no conjunto resposta para definir uma região de exclusão no espaço de busca, garantindo a diversificação crescente do resultado em termos das distâncias entre os elementos no conjunto resposta. Uma revisão da literatura [28, 103, 104], mostra que esses métodos são facilmente acoplados em rotinas de baixo nível e servem como base para operadores de filtragem [76, 79] baseada em similaridade diversificada. Além do acoplamento em rotinas de baixo nível, uma otimização que poderia ser considerada é o uso intensivo de estruturas de indexação métricas, que podem melhorar o desempenho da consulta ao particionar o espaço de busca. Assim, a hipótese pesquisada nessa dissertação é a seguinte:

Hipótese de Pesquisa: Uma rotina de busca a ser acoplada sobre índices métricos pode aumentar expressivamente o desempenho de buscas por similaridade diversificada baseadas em Influência, ao se beneficiar do particionamento de um índice métrico para descartar regiões que não satisfaçam simultaneamente os critérios *(i)* de similaridade para o objeto consultado, e *(ii)* dissimilaridade para os elementos no conjunto resposta. Para além dos ganhos de desempenho, comparar experimentalmente o comportamento dessa nova rotina contra o de consultas por similaridade em espaços de alta dimensionalidade pode auxiliar a caracterizar os ganhos semânticos do uso da similaridade diversificada.

Nesse sentido, a primeira contribuição dessa dissertação é a proposta de um algoritmo de busca por similaridade diversificada incremental que pode ser acoplado à qualquer estrutura de indexação métrica e atuar como uma implementação mais eficiente do operador

de filtragem por similaridade diversificada em comparação aos existentes na literatura. O algoritmo é construído tomando por base a rotina de busca por similaridade estado-da-arte *distance-browsing* [44, 72, 102] e formalizado por meio da proposição e demonstração de quatro lemmas derivados de propriedades da Teoria de Espaços Métricos [42, 102].

A segunda contribuição dessa dissertação é a investigação do algoritmo proposto para a realização de consultas por similaridade diversificada em conjuntos imersos em espaços de alta dimensionalidade e sujeitos ao fenômeno de concentração de distâncias [7, 34, 66]. Em particular, a proposta é avaliada sob a perspectiva da *dimensionalidade intrínseca local* (LID) [4, 6, 87], que permite analisar detalhadamente o comportamento do novo algoritmo de acordo com o nível da concentração de distâncias obtido para cada objeto de consulta. De forma resumida, os resultados experimentais indicam:

1. A viabilidade do uso do algoritmo proposto para buscas por similaridade diversificada mesmo em espaços de alta dimensionalidade. A rotina proposta apresenta um desempenho comparável a buscas somente por similaridade à medida que cresce a concentração de distâncias; e

2. Os efeitos da concentração de distância são proporcionalmente suavizados pelo uso de critérios baseados em Influência, que são capazes de manter discerníveis grupos de elementos "próximos" e "afastados" em espaços de alta dimensionalidade.

A combinação das duas contribuições dessa dissertação permite levantar novas hipóteses de pesquisa a serem conduzidas no futuro, incluindo o uso de consultas por similaridade diversificada como técnica de pré-processamento e mineração visual de dados com as perspectivas do objeto de consulta e diversos intervalos de LID.

1.2 Objetivos

O objetivo dessa dissertação é investigar e mensurar como critérios baseados em Influência podem ser usados para implementar, de forma eficiente, o operador de filtragem por similaridade diversificada e enriquecer a semântica de consultas apenas por similaridade. As contribuições derivadas desse objetivo permitem avançar o estado-da-arte dos operadores disponíveis para consultas por similaridade, consolidando-os como parte de um paradigma que inclui ferramentas adequadas e compatíveis com soluções que estendem sistemas gerenciadores de bases de dados por meio de métodos construídos com base na teoria de Espaços Métricos [51, 55].

Em particular, esse trabalho divide o objetivo geral em quatro objetivos específicos relacionados às consultas por similaridade diversificada, com o propósito de obter algoritmos e medidas experimentais que corroborem a hipótese da pesquisa. São eles:

1. Revisar estratégias de diversificação de resultados, com foco em identificar critérios de diversificação de resultados que possam ser modelados de acordo com a Teoria de Espaços Métricos e que sejam compatíveis com os operadores de busca por similaridade já consolidados na literatura;

2. Propor um algoritmo de busca incremental para índices métricos, que implemente de forma eficiente os critérios de diversidade revisados de forma a estender os operadores por similaridade da literatura em operadores por similaridade diversificada;

3. Revisar o conceito de dimensionalidade intrínseca local, com foco em identificar os aspectos de concentração de distância que impactam rotinas de busca por similaridade indexada em espaços métricos; e

4. Avaliar o algoritmo proposto em espaços de alta dimensionalidade, com foco tanto no desempenho computacional quanto na capacidade de recuperação semântica de acordo com diferentes valores de dimensionalidade intrínseca local.

1.3 Estrutura do Documento

Além do presente, esse texto está organizado em cinco outros capítulos, a saber:

2. Consultas por Similaridade, que apresenta o arcabouço teórico de Espaços Métricos;

3. Consultas por Diversidade, que apresenta os trabalhos relacionados;

4. Diversity browsing, que apresenta e avalia o algoritmo proposto;

5. Diversity browsing e dimensionalidade intrínseca local, que inclui uma avaliação experimental do algoritmo proposto em espaços de alta dimensionalidade; e

6. Conclusão, que discute todos os resultados obtidos, apresenta as publicações advindas dessa pesquisa e identifica possíveis trabalhos futuros.

Capítulo 2

Consultas por Similaridade

2.1 Espaços métricos

Um espaço de distâncias é, de forma geral, um espaço onde todas as distâncias entre os possíveis elementos de dados que ali possam existir ou ser representados são conhecidas [42, 43]. Um *Espaço Métrico* particulariza esse conceito de espaço de distâncias, com a diferença que cada objeto só pode ser representado por um único elemento de dados. Além disso, a distância entre objetos será sempre positiva e simétrica, de acordo com os axiomas definidos para uma função de distância dita métrica [26]. Formalmente, um Espaço Métrico $\mathcal{M} = \langle \mathbb{O}, \delta \rangle$ é definido para um domínio de dados $\mathbb{O}$ e uma função de distância $\delta : \mathcal{M} \times \mathcal{M} \rightarrow \mathbb{R}_+$ se, e somente se, para quaisquer objetos o_i, o_j e $o_h \in \mathbb{O}$, δ obedece as seguintes propriedades:

1. **Simetria:** $\delta(o_i, o_j) = \delta(o_j, o_i)$;
2. **Positividade:** $o_i \neq o_j \Rightarrow \delta(o_i, o_j) > 0$;
3. **Refletividade:** $\delta(o_i, o_i) = 0$;
4. **Não-negatividade:** $\delta(o_i, o_j) \geq 0$; e
5. **Desigualdade Triangular:** $\delta(o_i, o_j) \leq \delta(o_i, o_h) + \delta(o_h, o_j)$.

Uma das principais vantagens do modelo por Espaços Métricos é a de que qualquer subconjunto obtido a partir do domínio dos dados irá obedecer as mesmas cinco propriedades listadas anteriormente [42]. Essa características permitem subdividir (ou *particionar*) os dados modelados e aplicar diversas otimizações oriundas das comparações de distância

para realizar buscas por distâncias que servem como modelo base às *consultas por similaridade* [61]. Dessa forma, a modelagem por Espaços Métricos se torna particularmente interessante para complementar o motor de busca, armazenamento e indexação de sistemas gerenciadores de bases de dados [51, 55, 102].

O primeiro parâmetro do modelo de Espaços Métricos é o domínio dos dados, que precisa ser o mesmo domínio de entrada de uma dada função de distância. Na prática, esse nem sempre é o caso. Por exemplo, a função de distância Euclidiana L_2, $L_2 : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_+$ requer como domínio de entrada um espaço d-dimensional, que não é o mesmo espaço onde estão representadas imagens ou vídeos. Nesse sentido, é comum encontrar na literatura *funções de extração de características* que mapeiam o dado original em um domínio onde possa ser aplicada a função de distância. De forma geral, essas funções de extração de características podem ser divididas em três grandes grupos, a saber:

1. Dados brutos serializados: Esse grupo de funções essencialmente serializam o dado original para que este se "enquadre" (*cast*) no domínio da função de distância, o que pode levar a perda de informação, espacial ou de profundidade. Essas estratégias são normalmente empregadas para a representação de dados em espaços multidimensionais na realização de tarefas de classificação [24].

2. Engenharia de características: Essas funções realizam transformações aritméticas e de agregação sobre os dados brutos, substituindo-os pelas representações obtidas. Esses métodos são construídos especificamente para cada domínio de dados com foco em capturar informações que possam ter alguma validade semântica para a etapa posterior de análise, ou seja, destacam padrões pré-concebidos que possam existir nos dados para usá-los como posterior critério de comparação. Exemplos de padrões de baixo nível que são usados por essas funções incluem: cor, textura e formato em imagens; variância de sons e timbres em dados de áudio; e contagem de palavras em textos. Muito embora essas estratégias sejam capazes de realizar um mapeamento do domínio dos dados para o de busca, os padrões usados na transformação não são exaustivos, podendo levar a perdas de informação [51, 89].

3. Aprendizado de características: As funções de aprendizado de características não assumem padrões pré-concebidos que possam existir nos dados, porém salientam padrões frequentes e correlacionados dentro de cada particular conjunto de dados, a depender do viés de aprendizado por trás da extração de características. Usualmente, essas funções atuam como métodos de transformação (linear ou não linear) entre espaços, sendo que o dado bruto é inicialmente serializado. O método de

Análise de Componentes Principais é um representante de um método de aprendizado linear, cujo viés é destacar os atributos não correlacionados [62]. Métodos não-lineares incluem *autoencoders* que utilizam arquiteturas de redes de aprendizado profundo para gerar uma representação multidimensional do dado bruto, sendo que essa representação não precisa depender de um processo de rotulagem [58].

O segundo parâmetro no modelo de Espaços Métricos é a *função de distância*, que embute o viés de discriminar quais são os objetos próximos e quais são os objetos afastados. As funções de distância mais conhecidas são aquelas derivadas da geometria espacial, aplicadas sobre domínios d-dimensionais e que formam a Família Minkowski de funções de distância (L_ρ) [26]. Dados dois objetos $o_1, o_2 \in \mathbb{R}^d$, a Família Minkowski de funções é regida pelo parâmetro $\rho \in \mathbb{N}$ como na Equação 2.1.

$$\delta(o_1, o_2) = L_\rho(o_1, o_2) = \left(\sum_{j=1}^{d} |o_{1j} - o_{2j}|^\rho \right)^{\frac{1}{\rho}} \tag{2.1}$$

onde j é a j-ésima dimensão de o_1 e o_2. As funções mais usadas da Família Minkowski são as funções L_1 (City-Block), L_2 (Euclidiana) e L_∞ (Chebyshev).

Para além do viés da geometria espacial de proximidade, outra gama de funções de distância são amplamente usadas na literatura. Embora o número de funções possíveis seja ilimitado na prática [26,42,72], a taxonomia em Deza e Deza (2009) divide as funções de distância conhecidas nos seguintes grupos:

1. Funções para diferenças absolutas: Que inclui as funções de Bray-Curtis, Gowel, Soergel, Kulczynski, Canberra e Lorentzian.

2. Funções para distância de massa de probabilidade: Que inclui as funções de Intersecção, Czekanowski, Motyka, Ruzicka e Tanimoto.

3. Funções para produto interno: Que inclui as funções do Produto Interno, Cosseno, de Kumar-Hassebrook, Jaccard e Dice.

4. Funções para afinidade: Que inclui as funções de Bhattacharyya, Hellinger, Squared-Chord e Matusita.

O relacionamento entre um par de objetos por uma função de distância que considera um subconjunto de seus atributos gera um valor real que pode ser usado por um critério

objetivo de similaridade com a finalidade de organizar classes ou grupos de objetos que estejam próximos ou afastados. Na prática, esses critérios (ou restrições) de similaridade constituem a base dos operadores de comparação (tais como os operadores de comparação por identidade e ordem) que são empregados em consultas por similaridade.

2.1.1 Consultas por similaridade

Assim como para consultas baseadas em operadores por identidade ou ordem, consultas por similaridade permitem comparar tanto objetos dentro de um mesmo conjunto de dados quanto um elemento externo com elementos de um conjunto de dados, desde que estes elementos estejam representados em um mesmo domínio. O resultado dessa comparação (`TRUE`, `FALSE`) depende do critério da consulta, que pode envolver uma variável (constante ou não) imposta pelo usuário [39]. Elementos do conjunto de dados que satisfazem o critério são, necessariamente, incluídos no conjunto resposta da consulta.

A consulta por similaridade básica que se vale dessas premissas é a chamada *consulta por abrangência* (*Range query*). Uma consulta por abrangência retorna todos os elementos do conjunto de dados que estão dentro de um determinado "raio de distância" a um dado objeto de referência. O "raio" nada mais é do que a medida máxima de distância tolerada para inclusão de um elemento do conjunto de dados na resposta. A Figura 2.1(a) ilustra uma consulta por abrangência para um elemento de consulta o_q (notado por uma estrela escura) e um raio ξ em um espaço de duas dimensões considerando três diferentes funções de distância da Família Minkowski, sendo $p = 1, 2, \infty$.

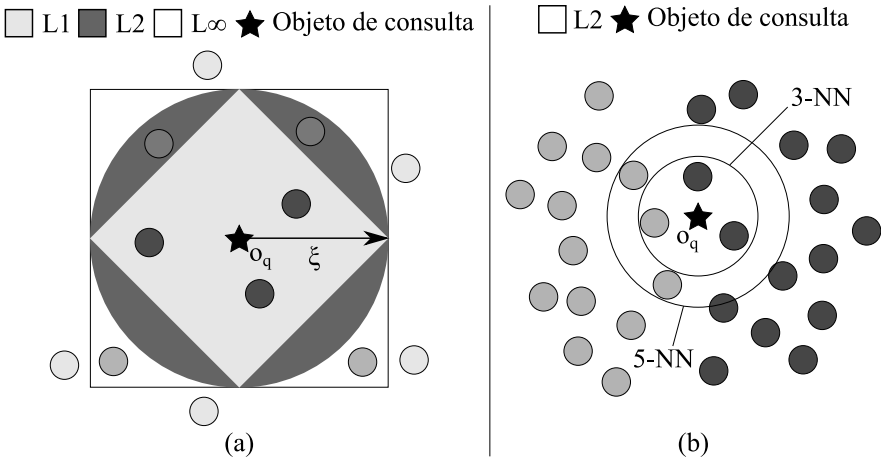


Figura 2.1: Consultas por similaridade. (a) Abrangência. (b) Vizinhança.

Definição 1 (Consulta por abrangência.) *Dado um Espaço Métrico* $\mathcal{M} = \langle \mathbb{O}, \delta \rangle$*, um conjunto de dados* $\mathcal{O} \subseteq \mathbb{O}$*, um objeto de consulta* $o_q \in \mathbb{O}$ *e um raio* $\xi \in \mathbb{R}_+$*, uma*

consulta por abrangência – Rg tem como resposta todos os elementos de $\mathcal{O}$ que não estão mais afastados do que ξ de o_q segundo δ, i.e., $Rg(o_q, \xi) = \{o_i \in \mathcal{O} \mid \delta(o_i,\ o_q) \leq \xi\}$.

Uma variação da consulta por abrangência é a consulta por vizinhança (ou dos k-Vizinhos mais Próximos) que não usa um valor explícito para o raio de distância tolerada, mas sim uma cardinalidade k esperada para o conjunto resposta. Nesse sentido, uma consulta por vizinhança nada mais é do que uma consulta por abrangência onde o raio ξ é tal que $|Rg| = k$ [86], sendo que, porém, esse valor de raio não é conhecido de antemão[1].

Definição 2 (Consulta por vizinhança.) *Dado um Espaço Métrico $\mathcal{M} = \langle \mathbb{O}, \delta \rangle$, um conjunto de dados $\mathcal{O} \subseteq \mathbb{O}$, um objeto de consulta $o_q \in \mathbb{O}$ e uma quantidade de vizinhos $k \in \mathbb{N}$, uma consulta por vizinhança – k-NN recupera os k elementos de $\mathcal{O}$ que são os mais próximos ao objeto o_q. Portanto, $k\text{-}NN(o_q, k) = \{o_1, o_2, \ldots, o_k\}$ onde:*

$$o_1 = \{o_i \in \mathcal{O} \mid \forall\ o_j \in \mathcal{O},\ \delta(o_i, o_q) \leq \delta(o_j, o_q)\},$$
$$o_2 = \{o_i \in \mathcal{O} \setminus \{o_1\} \mid \forall\ o_j \in \mathcal{O} \setminus \{o_1\},\ \delta(o_i, o_q) \leq \delta(o_j, o_q)\},$$
$$\ldots$$
$$o_k = \{o_i \in \mathcal{O} \setminus \{o_1, o_2, \ldots, o_{k-1}\} \mid \forall\ o_j \in \mathcal{O} \setminus \{o_1, o_2, \ldots, o_{k-1}\}, \delta(o_i, o_q) \leq \delta(o_j, o_q)\}.$$

A depender do conjunto de dados empregado na consulta, pequenas variações no raio podem ensejar na inclusão/remoção de muitos elementos no conjunto resposta (Ver Figura 2.1(a)). Portanto, consultas por vizinhança são alternativas às buscas por abrangência que permitem explorar o espaço de busca em função de um número relevante (da perspectiva do usuário) de entradas na resposta. A Figura 2.1(b) mostra duas consultas por vizinhança para $k = 3$ e 5 no mesmo espaço de busca da Figura 2.1(a).

2.1.2 Estruturas de indexação métricas

Ainda que seja possível resolver tanto consultas por abrangência quanto por vizinhança através da comparação direta do objeto de busca contra todo o conjunto de dados (com o algoritmo conhecido como busca sequencial[2] [39]), a modelagem por Espaços Métricos possibilita uma execução otimizada de algoritmos baseados em índices métricos que tiram vantagem das cinco propriedades de funções de distância para descartar regiões do espaço de busca que não incluem elementos candidatos ao conjunto resposta.

[1] Elementos equidistantes ao objeto de consulta na k-ésima posição (empates) são escolhidos arbitrariamente e, consequentemente, a resposta da consulta pode não ser única.

[2] O algoritmo de busca sequencial é a solução genérica que percorre todas as entradas no conjunto de dados para resolver uma consulta.

Todas estruturas de indexação métricas se valem do uso de *pivôs* $p \in \mathcal{P} \subseteq \mathcal{O} \subseteq \mathbb{O}$ e raios de cobertura $\xi_p \in \mathbb{R}_+$. Uma partição métrica $\langle p, \xi_p \rangle$ é dita ser uma bola fechada no Espaço Métrico se ela cobrir todos os elementos até uma distância ξ_p de p, *i.e.*, $\langle p, \xi_p \rangle \equiv Rg(p, \xi_p)$. Assim, dado um índice e uma consulta é possível utilizar os pivôs para identificar quais partições devem ser inspecionadas e quais devem ser descartadas da busca, o que pode reduzir expressivamente o custo da busca devido ao número de comparações por distância e ao gerenciamento de memória. Uma partição pode ser descartada da busca se ela não satisfizer os Princípio dos Limites Inferior e Superior.

Lemma 1 (Princípio dos Limites Inferior e Superior.) *Dado uma partição indexada por um pivô $p \in \mathbb{O}$ e uma consulta por abrangência $Rg(o_q, \xi)$ sobre $\mathcal{O} \subseteq \mathbb{O}$, os elementos o_i que fazem parte do conjunto resposta respeitam o limite inferior $\delta(o_q, o_i) \geq |\delta(o_q, p) - \delta(p, o_i)|$ e superior $\delta(o_q, o_i) \leq \delta(o_q, p) + \delta(p, o_i)$.*

Prova (Adaptado de [42]). *Da Desigualdade Triangular, têm-se que $\delta(o_q, o_i) + \delta(o_i, p) \geq \delta(o_q, p)$, logo $\delta(o_q, o_i) \geq \delta(o_q, p) - \delta(p, o_i)$. Por outro lado, têm-se que $\delta(o_i, o_q) + \delta(o_q, p) \geq \delta(p, o_i))$, logo $\delta(o_q, o_i) \geq \delta(o_i, p) - \delta(p, o_q)$. Da definição do operador módulo [83], segue que $\delta(o_q, o_i) \geq |\delta(o_i, p) - \delta(p, o_q)|$. Por definição, os elementos no conjunto resposta também satisfazem o limite superior que é equivalente à propriedade Desigualdade Triangular.* ■.

A Figura 2.2 ilustra como o Princípio do Limite Inferior pode ser usado para descartar uma partição métrica com centro em um pivô p e raio ξ_p menor que $[\delta(p, o_q) - \xi]$ de uma busca por abrangência $Rg(o_q, \xi)$. Note que os elementos que não satisfazem o Limite Superior também devem ser excluídos do conjunto resposta.

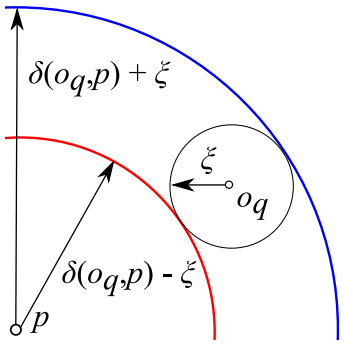


Figura 2.2: Princípio dos Limites Superior e Inferior.

Estruturas de indexação métricas se diferenciam entre si pela forma em que realizam o particionamento do conjunto de busca com o uso de pivôs. Embora diferentes taxonomias existam na literatura [43, 72, 102], a mais recente é a proposta no *survey* experimental

reportado por Chen *et al.* (2020), que divide os métodos existentes em três grandes categorias conforme destacado na Figura 2.3 [21].

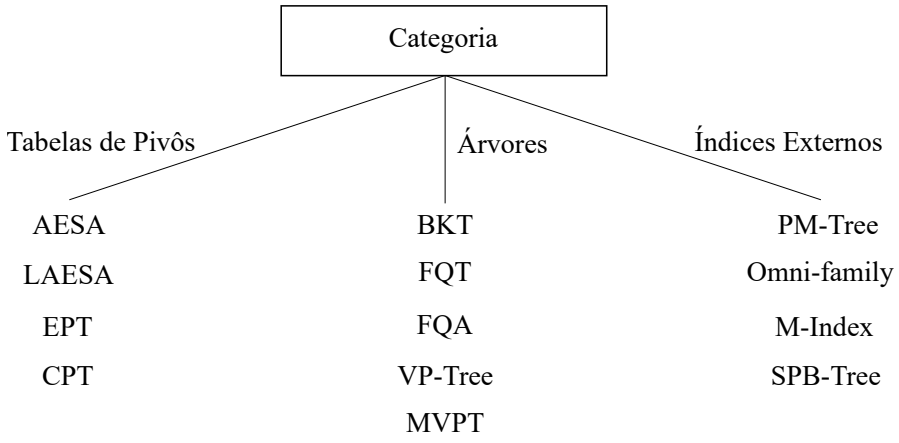


Figura 2.3: Taxonomia de índices métricos.

- Tabelas de pivôs. Essas estratégias empregam múltiplos pivôs sem hierarquia entre si e aplicam o Princípio do Limite Inferior para cada um dos pivôs de forma a reduzir o número de candidatos para avaliação por distância. Estruturas desse grupo incluem os métodos *Approximating and Eliminating Search Algorithm* (AESA) [75] e Linear AESA (LAESA) [57], que se constituem de *tabelas* com distâncias pré-computadas dos elementos do conjunto de dados aos pivôs. Ainda que exista uma diferença significativa de complexidade de espaço entre elas (quadrática para o AESA e linear para o LAESA), ambos os métodos são baseados na manutenção de uma matriz de distâncias (tabela) na memória principal, o que pode, na prática, inviabilizar seu uso para a consulta de grandes bases de dados.

- Árvores baseadas em pivôs. Árvores são uma alternativa às tabelas de pivôs, pois organizam as partições de forma hierárquica fornecendo um melhor desempenho para sistemas que contam com grande disponibilidade de memória. Um exemplo muito usado de árvores de pivôs é a *Vantage-Point Tree* (VP-Tree) [37,53,97] e suas variações VP-Forest [98] e Multiple VP-Trees [13]. VP-Trees são generalizações métricas de árvores binárias, o que as torna muito adequadas para tratar problemas relacionados à ordenação como se pode perceber por sua complexidade de busca [97].

- Índices externos baseados em pivôs. Estruturas de índices externos têm por objetivo *(i)* reduzir o número de comparações por distâncias e, principalmente, *(ii)* minimizar o número de acessos à memória secundária. Por exemplo, a estrutura *Pivoting M-tree* (PM-Tree) [82] gera árvores de baixa altura, com nós não-binários e partições

métricas sobrepostas. Já a Família Omni-Tree [88] divide, de forma explícita, a tabela de pivôs do conjunto de dados, sendo que a tabela de pivôs pode ser tratada por um índice acoplado, como, por exemplo, uma B-Tree$^+$ [39]. A *Space-filling curve and Pivot-based B^+-Tree* (SPB-Tree) [19] adiciona ainda uma terceira camada sobre o Família Omni: uma curva de Hilbert sobre a tabela de pivôs, gerando um mapeamento unidimensional para cada elemento de dados que é, posteriormente, inserido em uma Árvore-B$^+$ para otimizar o acesso ao disco.

Uma vez que parte do objetivo dessa dissertação é o de utilizar estruturas de indexação métricas para reduzir o número de comparações por distância e minimizar a quantidade de elementos inspecionados, as avaliações experimentais desse trabalho são construídas sobre o pressuposto que serão usados sistemas com uma quantidade suficientemente grande de memória principal. Os índices mais adequados para esse contexto são as *árvores baseadas em pivôs*, embora todos os outros métodos possam ser empregues sem prejuízo dos algoritmos propostos e suas construções teóricas.

Com relação às árvores baseadas em pivôs, o *survey* de Chen *et al.* (2020), após a comparação experimental de trinta e oito índices na realização de consultas por abrangência e vizinhança, indica que para o uso de estruturas em memória principal os métodos da família VP-Tree são os mais eficientes e os de mais baixo custo de construção na comparação com os competidores[3]. De acordo com esses resultados da literatura e para efeitos de implementação, no restante desse trabalho, será considerada a estrutura do índice VP-Tree [97], especificamente sua variante VP^{sb}-Tree, que permite controlar, otimizar e ajustar o número de elementos por nó-folha (e, consequentemente, a altura da árvore), o balanceamento da estrutura e a estratégia de escolha de pivôs [13, 53].

A rotina de construção de uma VP-Tree divide um conjunto de dados $\mathcal{O}$ de forma disjunta e hierárquica. Inicialmente, todos os elementos do conjunto são atribuídos a um nó-raiz e um elemento pivô $p \in \mathcal{O}$ é escolhido. Nesse ponto, calcula-se a *distribuição de distâncias* dos elementos do conjunto $\mathcal{O}$ para o pivô p, de onde se obtém a mediana, μ_p, e a distância máxima de qualquer elemento do conjunto ao pivô, M_p. O conjunto de dados é então divido em duas partições disjuntas, o ramo *esquerdo* e o ramo *direito*. Objetos cuja distância a p caem no intervalo $[0, \mu_p)$ são atribuídos ao ramo esquerdo, enquanto que os objetos restante que estão no intervalo $[\mu_p, M_p]$ são colocados no ramo direito. Uma vez que os ramos também podem ser vistos como conjuntos de dados (na razão da metade

[3]Os autores em [21] se referem a "escalabilidade" no sentido de lidar grandes conjuntos em uma única máquina e não em uma arquitetura distribuída que poderia fornecer ainda mais memória principal.

do conjunto anterior), eles são recursivamente divididos até que um número mínimo de elementos por partição seja alcançado. Nesse ponto, os ramos se tornam nós-folha e a construção da árvore termina.

A Figura 2.4 apresenta um exemplo de uma VP-Tree construída para um conjunto de dados em duas dimensões, com a distância L_2, pivôs escolhidos aleatoriamente dentro de cada partição e um máximo permitido de dois elementos por nó-folha. Em um primeiro momento, escolhe-se um pivô aleatório $p_1 = o_{14}$ do conjunto original $\mathcal{O} = \{o_1, o_2, \ldots, o_{20}\}$ que é incluído no conjunto de pivôs $\mathcal{P} = \{p_1\}$ e armazenado no nó-raiz da VP-Tree. Na sequência, calcula-se a mediana μ_{p_1} das distâncias de p_1 para todos elementos em $\mathcal{O} \setminus \mathcal{P}$. Esse procedimento é repetido para os conjuntos $\mathcal{O}_{esq_{p_1}} = \{o_i \in \mathcal{O} \setminus \mathcal{P} \mid \delta(o_i, p_1) < \mu_{p_1}\}$ e $\mathcal{O}_{dir_{p_1}} = \{o_i \in \mathcal{O} \setminus \mathcal{P} \mid \delta(o_i, p_1) \geq \mu_{p_1}\}$, gerando o segundo nível da árvore para os pivôs escolhidos aleatoriamente $p_2 = o_{17}$ e $p_3 = o_8$, que são incluídos em $\mathcal{P}$ e armazenados em nós-diretório. O procedimento prossegue até que não mais que dois elementos se encontrem no conjunto a ser particionado, tal como ilustra a Figura 2.4(c).

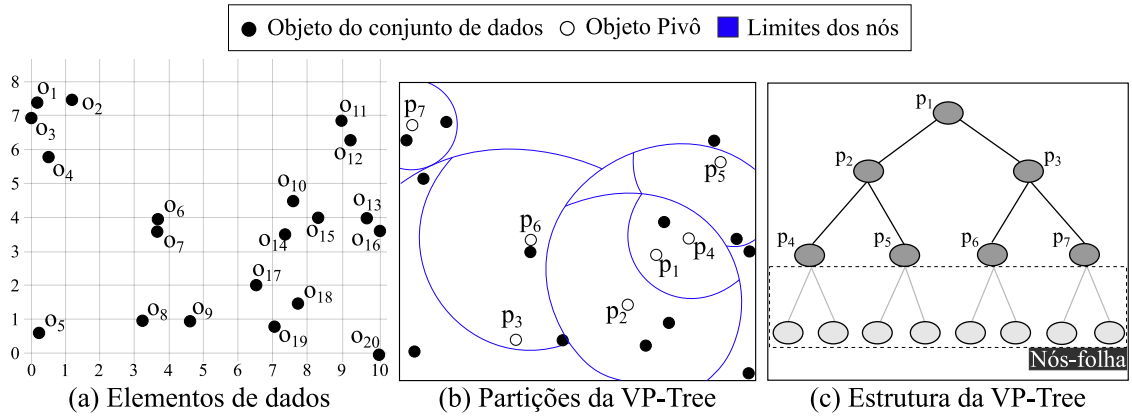


Figura 2.4: Exemplo de construção de uma VP^{sb}-Tree. (a) Conjunto de dados original. (b) Fronteira sem hierarquia das partições. (c) Estrutura hierárquica do índice métrico.

2.1.3 Algoritmos de busca baseados em índices

Uma vez que os dados consultados estejam divididos em partições, *algoritmos de busca por similaridade* podem ser empregados para tirar máximo proveito do Princípio dos Limites Inferior e Superior para reduzir o número de cálculos de distância, de partições visitadas (acessos à memória) ou uma combinação de ambos [21, 43]. Algoritmos para resolver consultas por abrangência partilham da mesma lógica: Todas as partições do menor nível hierárquico do índice que interceptarem o raio da consulta, *i.e.*, com a possibilidade de conter algum elemento que satisfaça o Princípio do Limite Inferior, precisam ser necessariamente inspecionadas, enquanto todas as demais partições podem ser descartadas. Note

que esse raciocínio por sobreposição de regiões não inclui analisar uma ou outra partição com prioridade, pois o raio que define a intersecção não se altera com o processamento da consulta.

O Algoritmo 1 apresenta uma rotina de processamento recursiva para a resolução de consultas por abrangência em um índice VP-Tree. Em essência, o algoritmo verifica primeiro se a consulta por vizinhança intercepta o lado esquerdo da árvore, o lado direito ou ambos, de acordo com o Princípio dos Limites Inferiores e Superiores calculados a partir da distância ao pivô que representa cada nó e o raio da consulta por abrangência.

Dados: Raiz VP-Tree nóRaiz, elemento de consulta $o_q \in \mathbb{O}$ e raio ξ.
Retorno: Conjunto resposta $Rg(o_q, \xi)$.

1 *buscaPorAbrangênciaRecursiva*$(nóRaiz, o_q, \xi)$

buscaAbrangênciaRecursiva(nó, o_q, ξ)
2 $ConjResp \leftarrow \emptyset$;
3 **if** *éFolha(nó)* **then**
4 $\quad ConjResp \leftarrow ConjResp \cup filtrarConjunto(nó, o_q, \xi)$;
5 **else**
6 $\quad$ **if** $\delta(o_q, nó.pivô) < nó.mediana$ **then**
7 $\quad\quad$ **if** $\delta(o_q, nó.pivô) - \xi \leq nó.mediana$ **then**
8 $\quad\quad\quad ConjResp \leftarrow ConjResp \cup buscaAbrangênciaRecursiva(nó.esquerdo, o_q, \xi)$;
9 $\quad\quad$ **if** $\delta(o_q, nó.pivô) + \xi \geq nó.mediana$ **then**
10 $\quad\quad\quad ConjResp \leftarrow ConjResp \cup buscaAbrangênciaRecursiva(nó.direito, o_q, \xi)$;
11 $\quad$ **else**
12 $\quad\quad$ **if** $\delta(o_q, nó.pivô) + \xi \geq nó.mediana$ **then**
13 $\quad\quad\quad ConjResp \leftarrow ConjResp \cup buscaAbrangênciaRecursiva(nó.direito, o_q, \xi)$;
14 $\quad\quad$ **if** $\delta(o_q, nó.pivô) - \xi \leq nó.mediana$ **then**
15 $\quad\quad\quad ConjResp \leftarrow ConjResp \cup buscaAbrangênciaRecursiva(nó.esquerdo, o_q, \xi)$;
16 **return** $ConjResp \cup filtrarConjunto(\{nó.pivô\}, o_q, \xi)$;

filtrarConjunto($\mathcal{O}, o_q, \xi$)
17 $rSet \leftarrow \emptyset$;
18 **for** $o_i \in \mathcal{O}$ **do**
19 $\quad$ **if** $\delta(o_i, o_q) \leq \xi$ **then**
20 $\quad\quad rSet \leftarrow rSet \cup \{o_i\}$;
21 **return** rSet;

Algoritmo 1: Algoritmo recursivo de busca por abrangência sobre VP-Trees – Adaptado do original proposto em [97].

Algoritmos de busca para resolver consultas por vizinhança, no entanto, não podem se beneficiar imediatamente da exclusão de regiões por sobreposição como os de busca por abrangência. Isso porque embora seja possível reduzir na teoria uma consulta por vizinhança a uma por abrangência, o raio que delimita a vizinhança não é conhecido de

Dados: Raiz VP-Tree nóRaiz, elemento de consulta o_q, quantidade de vizinhos k
Retorno: Conjunto resposta $kNN(o_q, k)$.

1 $buscaVizinhançaRecursiva(nóRaiz, o_q, k, ConjResp = \emptyset, \xi = \infty)$

buscaVizinhançaRecursiva($nó, o_q, k, ConjResp, \xi$)
2 **if** *éFolha(nó)* **then**
3 $\quad filtrarConjunto(nó, o_q, \xi, resposta)$;
4 **else**
5 $\quad filtrarConjunto(\{nó.pivô\}, o_q, \xi, ConjResp)$;
6 $\quad$ **if** $\delta(o_q, nó.pivô) < nó.mediana$ **then**
7 $\quad\quad$ **if** $\delta(o_q, nó.pivô) - \xi \leq nó.mediana$ **then**
8 $\quad\quad\quad buscaVizinhançaRecursiva(nó.esquerdo, o_q, k, ConjResp, \xi)$;
9 $\quad\quad$ **if** $\delta(o_q, nó.pivô) + \xi \geq nó.mediana$ **then**
10 $\quad\quad\quad buscaVizinhançaRecursiva(nó.direito, o_q, k, ConjResp, \xi)$;
11 $\quad$ **else**
12 $\quad\quad$ **if** $\delta(o_q, nó.pivô) + \xi \geq nó.mediana$ **then**
13 $\quad\quad\quad buscaVizinhançaRecursiva(nó.direito, o_q, k, ConjResp, \xi)$;
14 $\quad\quad$ **if** $\delta(o_q, nó.pivô) - \xi \leq nó.mediana$ **then**
15 $\quad\quad\quad buscaVizinhançaRecursiva(nó.esquerdo, o_q, k, ConjResp, \xi)$;

filtrarConjunto($\mathcal{O}, o_q, *\xi, *ConjResp$);
16 **for** $o_i \in \mathcal{O}$ **do**
17 $\quad$ **if** $tamanho(ConjResp) < k \vee \delta(o_i, o_q) < \xi$ **then**
18 $\quad\quad ConjResp \leftarrow ConjResp \cup \{o_i\}$;
19 $\quad\quad$ **if** $tamanho(ConjResp) > k$ **then**
20 $\quad\quad\quad ConjResp \leftarrow ConjResp \setminus \{elementoMaisDistante(o_q, ConjResp)\}$;
21 $\quad\quad\quad \xi \leftarrow maiorDistância(o_q, ConjResp)$;

Algoritmo 2: Algoritmo *branch-and-bound* recursivo de busca por vizinhança sobre VP-Trees – Adaptado do original proposto em [97].

antemão e precisa ser *delimitado* durante o processo da busca. Nesse sentido, a *ordem* em que as partições são visitadas é extremamente relevante pois impacta diretamente no descobrimento do raio da consulta por vizinhança. Assim, ao contrário de buscas por abrangência, o problema de ordenação está presente nos algoritmos de buscas por vizinhança e o uso de estruturas de árvores binárias semelhantes a VP-Tree podem auxiliar na otimização do desempenho dessas buscas[4].

Duas estratégias são comumente empregadas por esses algoritmos de busca: *(i)* iniciar o raio da consulta como infinito $\xi = \infty$ e reduzi-lo dinamicamente toda vez que atingir um nó-folha, ou ainda *(ii)* assumir um raio inicial nulo $\xi = 0$ e incrementá-lo dinamicamente quando o próximo vizinho for encontrado. A proposta original para VP-Trees é baseada na

[4]Todas as estruturas de árvore (excluídas as tabelas de pivôs) usadas para indexação métricas fornecem suporte implícito para ordenação por distâncias, sendo que estruturas de árvores binárias (como VP-Trees) tem complexidade média conhecida na resolução do problema da busca [39, 97].

primeira estratégia e emprega uma heurística *branch-and-bound* para descartar partições do espaço de busca a medida que reduz o tamanho do raio [21,97]. Esse algoritmo percorre os nós-folha em ordem crescente e hierárquica de distância, descartando aqueles que não interceptam o raio estimado da consulta no momento em que são avaliados no percurso da árvore. O Algoritmo 2 apresenta a rotina de busca *branch-and-bound* para a resolução de consultas por vizinhança sobre VP-Trees.

Em essência, o algoritmo percorre a árvore por profundidade da esquerda para direita marcando para inspeção os ramos que interceptem o raio da consulta, cujo valor é definido no momento do percurso em que os nós-diretório (os pais dos ramos) são avaliados. A Figura 2.5 apresenta a execução do Algoritmo 2 para uma consulta por vizinhança com $k = 1$, distância L_2 e o mesmo conjunto de dados indexado no exemplo da Figura 2.4. Para essa consulta, foi utilizado o objeto de consulta $o_q = o_{11}$ (um elemento do próprio conjunto de dados) para que se pudesse identificar o comportamento do algoritmo para uma *point query* [18, 42], uma consulta equivalente a busca por abrangência $Rg(o_{11}, 0)$.

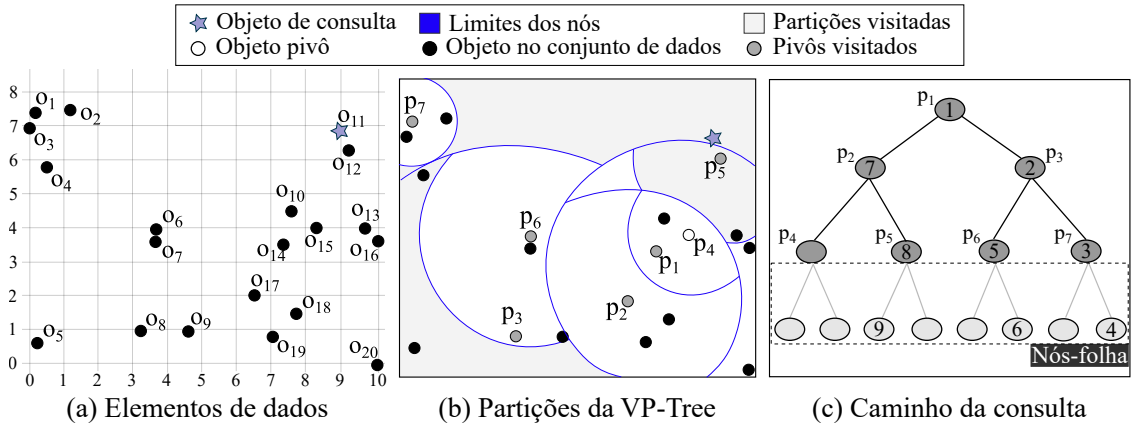


Figura 2.5: Exemplo de uma consulta por vizinhança para $k = 1$ e $o_q = o_{11}$ executada pelo Algoritmo 2.5 e sua rotina *branch-and-bound*.

Inicialmente, o Algoritmo 2.5 verifica a partição indexada por p_1 e marca para análise tanto o espaço fora quanto espaço dentro do limite imposto por μ_{p_1}. O raio de cobertura ξ é atualizado para $\xi = \delta(o_q, p_1)$. Na sequência, o algoritmo analisa a região indexada por p_3 e também marca para análise tanto a região de p_6 quanto de p_7, em todas análises não há redução do raio de cobertura. A região de p_7 é analisada, sendo o lado com limite inferior a mediana descartado e o lado direito inspecionado. Nesse ponto, o primeiro nó-folha é analisado e o elemento o_{11} é escolhido como o mais próximo, o que gera uma redução do raio de cobertura para $\xi = \delta(o_q, o_{11}) = 0$. Na sequência, é avaliada a região fora do limite da mediana de p_6, onde os elementos o_5 e o_4 são examinados e descartados, assim como a região interna à mediana de p_6 é também descartada. Nesse ponto, o percurso retorna à

análise de p_2, descartando a partição limitada por μ_{p_2} e analisando p_5. A partição externa a μ_{p_5} é, então, podada da busca e o nó-folha interno a p_5 contendo o_{13} é inspecionado. Ao todo foram realizados 10 cálculos de distância ($p_1, p_3, p_7, o_{11}, p_6, o_5, o_4, p_2, p_5, o_{13}$) para a obtenção da resposta da consulta (de possíveis 20).

Um algoritmo potencialmente mais eficiente para essas consultas seria o que considera a premissa inversa (raio inicial nulo) conhecido como k-NN incremental (por sua propriedade de recuperar um vizinho próximo de cada vez) formalizado como o algoritmo *distance browsing* em sua proposta completa [44]. No melhor do nosso conhecimento [21, 22, 42, 43, 72, 102], não há comparação experimental do desempenho de VP-Trees com a adoção do algoritmo *distance browsing* ao invés do algoritmo *branch-and-bound*, pois a implementação incremental requer modificações no procedimento de consulta[5].

O algoritmo *distance browsing* é *ótimo* no número mínimo de cálculos de distância para a realização de uma consulta por vizinhança [44]. Na prática, porém, o algoritmo depende do uso intensivo de duas filas de prioridade que requerem alto consumo e gerenciamento de memória [85, 92]. Essas duas filas de prioridade ordenam separadamente *(i)* os elementos inspecionados por suas distâncias ao objeto de consulta e *(ii)* as partições a serem visitadas por seus limites de distância *mínimos* e *máximos* ao objeto de consulta. Esses limites *mínimo* (δ_{min}) e *máximo* (δ_{max}) são específicos do formato e do tipo da partição métrica e, consequentemente, precisam ser definidas para cada índice métrico.

Ainda que, no melhor do nosso conhecimento, os proponentes da VP-Tree não tenham demonstrado explicitamente esses limites para essas estruturas de árvore ou variantes [13, 98], é possível obtê-los diretamente da estrutura das partições geradas por esse tipo de índice. Em particular, apresentamos na Definição 3 uma solução para o problema de se calcular os limites de distância mínimo e máximo por uma decomposição em função da localização de um objeto de consulta qualquer $o_q \in \mathbb{O}$.

Definição 3 (Limites δ_{min} e δ_{max} para uma partição VP-Tree.) *Dado um objeto de consulta $o_q \in \mathbb{O}$ e um conjunto de dados $\mathcal{O} \subseteq \mathbb{O}$, tem-se da definição do particionamento gerado por um pivô $p \in \mathcal{O}$ de uma VP-Tree, a obtenção de dois conjuntos $\mathcal{O}_{esq_p}$ e $\mathcal{O}_{dir_p}$ a partir da mediana de distâncias μ_p com $\mathcal{O}_{esq_p} \cup \mathcal{O}_{dir_p} = \mathcal{O}$ e $\mathcal{O}_{esq_p} \cap \mathcal{O}_{dir_p} = \emptyset$. Uma vez que $\mathcal{O} \cup \mathcal{O}^C = \mathbb{O}$ e $\mathcal{O} \cap \mathcal{O}^C = \emptyset$, por trivial tem-se que o_q está localizado de forma exclusiva ou em $\mathcal{O}_{esq_p}$, ou em $\mathcal{O}_{dir_p}$, ou em $\mathcal{O}^C$, pois $\mathcal{O}_{esq_p}, \mathcal{O}_{dir_p}$ e $\mathcal{O}^C$ formam*

[5]Dado que o algoritmo incremental se propõe a ser *ótimo* no número de cálculos de distâncias e que VP-Trees também foram projetadas para otimizar esse aspecto em detrimento aos acessos a disco, é esperado (em termos de comparações por distância) que o desempenho de VP-Trees e variantes com o *distance browsing* seja igual ou superior ao reportado no *survey* experimental de Chen *et al.* (2020).

uma partição disjunta de $\mathbb{O}$. Esse comportamento é o mesmo observado para qualquer subconjunto de $\mathcal{O}$ a ser particionado como um nó de uma VP-Tree.

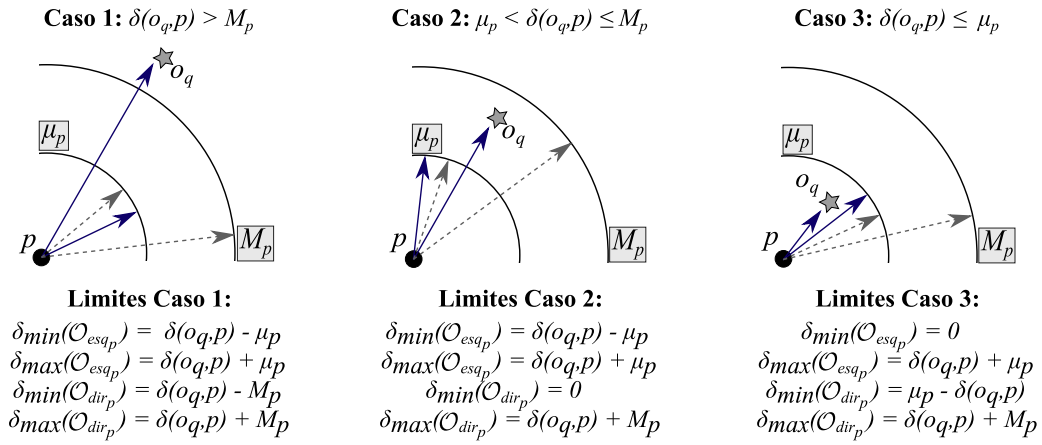


Figura 2.6: Limites de distância mínimo δ_{min} e máximo δ_{max} para partições à esquerda e direita de um pivô de uma VP-Tree considerando a localização do objeto de consulta o_q.

A Figura 2.6 ilustra o problema de se obter os limites de distância a partir da decomposição gerada pela Definição 3. Note que os valores dos limites são diretamente ligados ao posicionamento do objeto de consulta o_q com relação à partição da VP-Tree gerada por p, o que é facilmente determinado por $\delta(o_q, p)$. Caso $\delta(o_q, p)$ seja superior ao valor máximo M_p obtido da distribuição de distâncias que gera a partição determinada por p, então o_q está localizado em $\mathcal{O}^C$, *i.e.*, *fora* da partição dada por p – Figura 2.6(a). Por outro lado, se $\delta(o_q, p)$ é menor que M_p porém maior ou igual a mediana de distâncias, então o_q se encontra na região do "anel" gerado por p, *i.e.*, $o_q \in \mathcal{O}_{dir_p}$ – Figura 2.6(b). Por último, caso $\delta(o_q, p)$ seja menor que a mediana de distâncias, então o_q está localizado na partição métrica dada por $\langle p, \delta(o_q, p)\rangle$, *i.e.*, $o_q \in \mathcal{O}_{esq_p}$ – Figura 2.6(c).

A menor distância possível do objeto de consulta para a partição na qual ele se localiza é nula (zero), pois o próprio objeto está coberto pela referida partição. As demais distâncias são obtidas diretamente pelo Princípio do Limite Inferior e pela propriedade da Desigualdade Triangular. Para o primeiro caso na Figura 2.6(a), o limite mínimo de distância para $\mathcal{O}_{dir_p}$ e $\mathcal{O}_{esq_p}$ é dado por $\delta_{min} = \delta(o_q, p) - M_p$ e $\delta_{min} = \delta(o_q, p) - \mu_p$. Já para o segundo caso na Figura 2.6(b), tem-se que o limite mínimo para $\mathcal{O}_{dir_p}$ é nulo pois $o_q \in \mathcal{O}_{dir_p}$, enquanto que a distância para $\mathcal{O}_{esq_p}$ é dada por $\delta_{min} = \delta(o_q, p) - \mu_p$. Analogamente, para o terceiro caso na Figura 2.6(c), tem-se que δ_{min} para $\mathcal{O}_{esq_p}$ é nulo e $\delta_{min} = \mu_p - \delta(o_q, p)$ para $\mathcal{O}_{dir_p}$. Há de se destacar que *(i)* os limites superiores são dados pelas somas $\delta(o_q, p) + M_p$ e $\delta(o_q, p) + \mu_p$ para as partições $\mathcal{O}_{dir_p}$ e $\mathcal{O}_{esq_p}$, respectivamente, e *(ii)* os limites superiores devem considerar o menor valor de δ_{max} calculado o conjunto de pivôs, no caso de serem percorridos mais de um pivô em uma busca na VP-Tree.

Na execução de uma busca por vizinhança, o algoritmo *distance browsing* usa os limites de distância mínima δ_{min} para manter ordenada a fila de partições candidatas a serem inspecionadas (os desempates ocorrem pelo menor valor de δ_{max}) e, de forma complementar, uma segunda fila é mantida para ordenar os elementos candidatos por distância ao objeto de consulta. O algoritmo inicializa a lista de elementos candidatos como uma fila vazia e insere o nó-raiz da árvore na fila de partições candidatas. A partir desse momento, são realizadas iterações incrementais até que os k elementos mais próximos sejam encontrados. A cada iteração, o *distance browsing* examina o topo das duas filas de prioridade e compara o menor limite de distância mínima da fila de partições com a distância do objeto de consulta ao candidato mais próximo na fila de elementos.

Dados: Raiz VP-Tree nóRaiz, elemento de consulta o_q, quantidade de vizinhos k.
Retorno: Conjunto resposta $kNN(o_q, k)$.

```
1  nóRaiz.δ_min ← 0.0;
2  nóRaiz.δ_max ← ∞;
3  ConjResp ← ∅;
4  filaPrNós ← {nóRaiz};/* Fila de prioridade para partições          */
5  filaPrElem ← ∅;/* Fila de prioridade para objetos                  */
6  while (|filaPrNós| > 0 ∨ |filaPrElem| > 0) ∧ |ConjResp| < k do
7      if |filaPrElem| = 0 then
8          nó ← removeTopo(filaPrNós);
9          filaPrElem ← filaPrElem ∪ {nó.pivô};
10         if éFolha(nó) then filaPrElem ← filaPrElem ∪ {nó.elementos} ;
11         else
12             calcularδ_minδ_max(nó.esquerdo);
13             calcularδ_minδ_max(nó.direito);
14             filaPrNós ← filaPrNós ∪ {nó.esquerdo} ∪ {nó.direito};
15     else if |filaPrNós| > 0 ∧ δ_min(topo(filaPrNós)) < δ(o_q,topo(filaPrElem)) then
16         nó ← removeTopo(filaPrNós);
17         filaPrElem ← filaPrElem ∪ {nó.pivô};
18         if éFolha(nó) then
19             filaPrElem ← filaPrElem ∪ {nó.elementos};
20         else
21             calcularδ_minδ_max(nó.esquerdo);
22             calcularδ_minδ_max(nó.direito);
23             filaPrNós ← filaPrNós ∪ {nó.esquerdo} ∪ {nó.direito};
24     else
25         o_i ← removeTopo(filaPrElem);
26         ConjResp ← ConjResp ∪ {o_i};
27 return ConjResp;
```

Algoritmo 3: Algoritmo incremental *distance browsing* para buscas por vizinhança sobre VP-Trees – Construído sob o algoritmo geral proposto em [44].

Caso o melhor candidato na fila de elementos esteja mais próximo do objeto de consulta do que o menor limite mínimo de distância dentre todas as partições, então o candidato é retornado como o vizinho mais próximo e o número de vizinhos necessário para responder a consulta é decrementado de uma unidade. Caso contrário, a partição é removida da fila e inspecionada. Se esta for um nó-diretório, os seus nós-filho são inseridos na fila de prioridade de partições de acordo com os valores δ_{min} e δ_{max} calculados a partir dos seus pivôs, senão os elementos dentro do nó-folha são carregados em memória, comparados por distância ao objeto o_q e inseridos na fila de elementos candidatos. O Algoritmo 3 apresenta a rotina de busca incremental para a resolução de consultas por vizinhança.

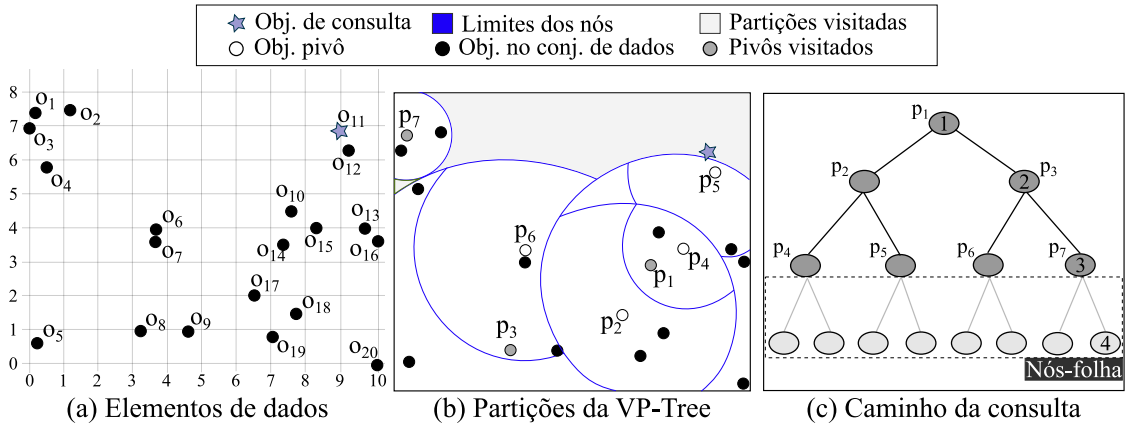


Figura 2.7: Exemplo de uma consulta por vizinhança para $k = 1$ e $o_q = o_{11}$ executada pelo Algoritmo 3 e sua rotina incremental.

A Figura 2.7 apresenta a execução do Algoritmo 3 para a mesma consulta do exemplo da Figura 2.5 para $k = 1$ e $o_q = o_{11}$. Inicialmente, o nó-raiz representado pivô p_1 é inserido na fila de partições e, na primeira iteração, é removido para dar lugar aos nós-diretório-filho que representam as partições $\mathcal{O}_{esq_{p_1}}$ (com $\delta_{min} = 0.06$) e $\mathcal{O}_{dir_{p_1}}$ (com $\delta_{min} = 0$). O pivô p_1 é incluído na fila de candidatos com $\delta(o_q, p_1) = 1.69$. Como $\delta_{min} = 0 < \delta(o_q, p_1) = 1.69$, o nó-diretório $\mathcal{O}_{dir_{p_1}}$ representado pelo pivô p_3 é então removido e seus filhos $\mathcal{O}_{esq_{p_3}}$ (com $\delta_{min} = 1.57$) e $\mathcal{O}_{dir_{p_3}}$ (com $\delta_{min} = 0$) são inseridos na fila de prioridade. O pivô p_3 é adicionado ao final fila de candidatos com $\delta(o_q, p_3) = 4.87$. De forma análoga, a partição $\mathcal{O}_{dir_{p_3}}$ é removida na próxima iteração para dar lugar as partições geradas por p_7, $\mathcal{O}_{esq_{p_7}}$ (com $\delta_{min} = 4.51$) e $\mathcal{O}_{dir_{p_7}}$ (com $\delta_{min} = 0$). O pivô p_7 também é inserido ao final da fila de candidatos com $\delta(o_q, p_7) = 5.39$.

Na próxima iteração, a partição $\mathcal{O}_{dir_{p_7}}$ é escolhida para inspeção e, como se trata de um nó-folha, os seus elementos $\mathcal{O}_{dir_{p_7}} = \{o_{11}\}$ são comparados diretamente com o_q e inseridos na lista de candidatos. Após essa operação, a lista de candidatos passa a ser encabeçada por $o_{11}(\delta(o_q, o_{11}) = 0)$, seguido pelos elementos $p_1 = o_{14}, p_3 = o_8, p_7 = o_1$, enquanto que

a lista de partições contém $\mathcal{O}_{esq_{p_1}}, \mathcal{O}_{esq_{p_3}}$ e $\mathcal{O}_{esq_{p_7}}$, nessa ordem. Na próxima iteração, o Algoritmo 3 compara o topo das duas filas e retorna o_{11} como o elemento mais próximo, finalizando a execução da consulta. Ao todo, o *distance browsing* realizou 4 comparações por distância (p_1, p_3, p_7, o_{11}) contra os 10 cálculos do algoritmo *branch-and-bound*, uma redução que pode impactar diretamente o tempo total da busca.

2.2 Concentração de Distâncias

Índices métricos combinam o acesso à partições do conjunto de dados com o Princípio dos Limites Inferior e Superior para diminuir o número de comparações por distância na resolução de consultas por similaridade. Essa estratégia é baseada na premissa que é possível distinguir valores diferentes de distância entre o objeto de consulta e elementos do conjunto de dados e que objetos "distantes" podem ser afastados de objetos "próximos". No entanto, existe um caso particular conhecido, que também é o domínio usual de características extraídas por redes de aprendizado profundo, onde as distâncias entre quaisquer elementos podem não ser significativas o suficiente para que objetos próximos sejam separados de objetos distantes. Esse cenário de estresse acontece quando determinadas funções de distância, em especial as da Família Minkowski [36], são usadas para medir a proximidade de objetos em espaços métricos e vetoriais de alta dimensionalidade $d \to \infty$. Nesses espaços, as *distâncias* entre os objetos tendem a assumir *valores reais* que estão *concentrados* em um intervalo tão pequeno que nenhuma diferença significativa entre eles pode ser encontrada na maioria das comparações entre pares de valores [66].

Esse fenômeno é conhecido como *concentração de distâncias* [7, 66] e é caracterizado pela distribuição de distâncias do conjunto de dados, que tende a convergir para uma única frequência na medida em que a dimensionalidade usada para representar os dados aumenta. A Figura 2.8 ilustra esse fenômeno para um conjunto de dados sintético de $|\mathcal{O}| = 1.000$ elementos, cujos atributos d dimensionais são gerados aleatoriamente no intervalo $[0, 1]$ segundo uma função de densidade de probabilidade uniforme (iid.), e comparados pela distância L_2 de onde se extrai a distribuição de distância entre todos os pares de elementos, *i.e.*, os valores (primeiro eixo) incluem todos os distintos valores de $\delta(o_i, o_j), o_i \neq o_j, o_i, o_j \in \mathcal{O}$, e as frequências (segundo eixo) incluem a contagem normalizada das distâncias observadas para os pares de elementos comparados. Para um espaço de "baixa" dimensionalidade $d = 2$ (Figura 2.8(a)) é possível perceber que os valores das distâncias são facilmente separáveis, pois existe grandes diferença entre o menor, o mais frequente e o maior valor. Essas diferenças são rapidamente reduzidas com o aumento

da dimensionalidade (Figura 2.8(b)), até que se *concentram* de forma quase exclusiva (pequena abertura e desvio-padrão) ao redor da média das frequências (Figura 2.8(c)).

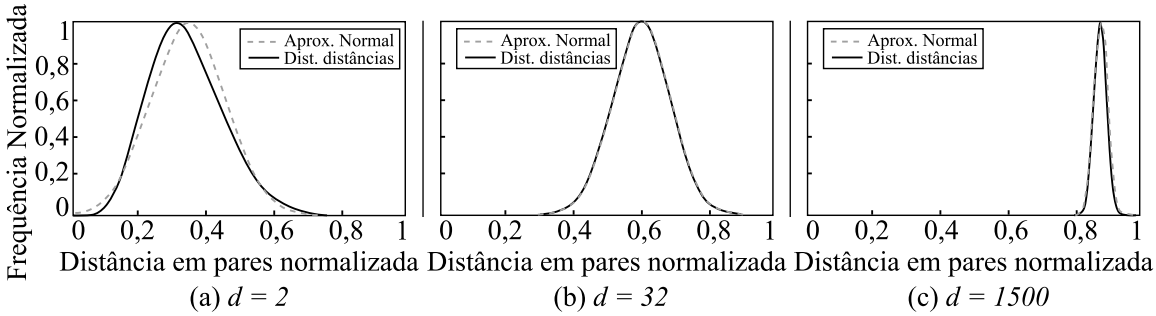


Figura 2.8: Distribuição de distâncias e variância relativa para um conjunto de dados sintético iid. de 1.000 elementos comparados pela métrica L_2.

Na presença de concentração de distâncias, o desempenho de índices métricos é comprometido, podendo se tornar inclusive inferior ao uso do algoritmo de *busca sequencial*, devido à incapacidade de se descartar análises com o uso de um limiar de referência (distância ao pivô), que não é significativamente diferente da distância do objeto de consulta à maioria das partições. Portanto, é importante relacionar a concentração de distâncias com a eficiência do índice métrico de forma que esse possa ser ou não usado na resolução particular de uma consulta por similaridade.

Beyer *et al.* (1999) foram os pioneiros na caracterização da concentração de distâncias ao propor uma quantificação para o fenômeno conhecida como *contraste relativo*. Essa medida é obtida com a diferença normalizada entre a maior e a menor distância encontrada entre pares de elementos dentro de um conjunto de dados que tende a zero para um número crescente de dimensões e em um espaço vetorial iid. (Prova disponível em [10]). Embora forte, esse resultado não é diretamente aplicável a conjuntos de dados reais representados em espaços vetoriais que normalmente não seguem uma distribuição iid. [60]. Uma caracterização que remove as restrições iid. e mede a concentração de distâncias como um índice normalizado entre $[0, 1]$ (zero sendo a distribuição menos concentrada) é a proposta de François *et al.* (2007) conhecida como *variância relativa*. A variância relativa (VR) é expressa como a razão entre o desvio-padrão e a média da distribuição normalizada de distância entre pares de objetos, tal qual na Equação 2.2 (Prova em [36]).

$$VR(\mathcal{O}) = \frac{\sigma\left(\{\delta(o_i, o_j), o_i, o_j \in \mathcal{O}; o_i \neq o_j\}\right)}{\mu\left(\{\delta(o_i, o_j), o_i, o_j \in \mathcal{O}; o_i \neq o_j\}\right)}, \text{com} \lim_{d\to\infty} VR(\mathcal{O}) = 0. \tag{2.2}$$

A Figura 2.8 também ilustra como uma distribuição normalizada de distâncias em espaços iid. se aproxima rapidamente da Distribuição Normal, com um desvio-padrão

cada vez menor e média crescente. Para conjuntos de dados reais, a concentração tende a ocorrer de forma mais lenta, *i.e.*, as distâncias se concentram menos para representações de maior dimensionalidade pois pode existir correlação e dependência entre os dados. Portanto, para além da medida de variância relativa, caracteriza-se também a *dificuldade* de se *realizar uma busca* por similaridade em dados reais imersos em espaços de alta dimensionalidade por meio da *dimensionalidade intrínseca*, que expressa o número mínimo de dimensões no qual o conjunto de dados consultado possa vir a ter um comportamento análogo ao de um conjunto de dados iid. [34, 60].

2.2.1 Dimensionalidade intrínseca

A dimensionalidade intrínseca de um conjunto de dados é um indicador do comportamento esperado para a sua distribuição de distâncias, o que permite classificar um conjunto de dados como "mais fácil" ou "mais difícil" de se consultar com base no fenômeno de concentração de distância[6]. Ao contrário da variância relativa, no entanto, ainda não há uma expressão exata para o cálculo da dimensionalidade intrínseca de um conjunto de dados, sobrando apenas fórmulas aproximativas [18, 60, 66].

Uma aproximação direta dessa medida pode ser obtida pelo parâmetro de correlação entre um valor de distância e a frequência acumulada da distribuição de distância associada ao valor [7]. Para se obter esse parâmetro linear, métodos da literatura usam o gráfico de valores de distância por suas frequências na distribuição acumulada. Usualmente, ambos os eixos são representados em escala logarítmica pois é esperado que a frequência acumulada aumente exponencialmente seguindo o valor da distância. A dimensionalidade intrínseca, nesse caso, é estimada como o arredondamento da inclinação da reta que aproxima os pontos no gráfico [60, 66].

Porém, dado que o conjunto de dados analisado pode incluir apenas uma amostra parcial com relação ao domínio analisado, a estimativa por correlação não é a mais indicada para se calcular a dimensionalidade intrínseca de conjuntos verdadeiramente não-massivos de dados [34]. Alternativamente, a estimativa proposta por Chávez *et al.* (2001) é baseada na aproximação da distribuição de distância para com a Distribuição Normal e permite obter uma estimativa contínua a partir de amostras da distribuição de distâncias. Uma vez que para um conjunto iid. a média de distâncias aumenta e o desvio-padrão reduz a medida que aumenta o número de dimensões, os estudos em [18, 65, 93] obser-

[6] A abstração da dimensionalidade intrínseca baseada na análise da distribuição de distâncias também vale para espaços métricos não-vetoriais [66]

varam que o valor estimado da dimensionalidade intrínseca $\mathcal{D}$ se mantém o mesmo para $\mathcal{D} = c \cdot \mu\left(\{\delta(o_i, o_j), o_i, o_j \in \mathcal{O}; o_i \neq o_j\}\right) / \sigma\left(\{\delta(o_i, o_j), o_i, o_j \in \mathcal{O}; o_i \neq o_j\}\right)$ para alguma constante $c \in \mathbb{R}$. De acordo com essa observação, Chávez *et al.* (2001) propôs estimar a dimensionalidade intrínseca de uma amostra do domínio de dados a partir da expressão detalhada na Equação 2.3 batizada posteriormente como ρ–score [63].

$$\mathcal{D} \approx \rho\text{–score} = \frac{\mu^2\left(\{\delta(o_i, o_j), o_i, o_j \in \mathcal{O}; o_i \neq o_j\}\right)}{2 \cdot \sigma^2\left(\{\delta(o_i, o_j), o_i, o_j \in \mathcal{O}; o_i \neq o_j\}\right)} \tag{2.3}$$

O estudo de Pestov (2009) apresenta uma comparação teórica detalhada dos métodos de estimativa de dimensionalidade intrínseca por correlação e estimativa ρ de acordo com axiomas e propriedades que precisam ser satisfeitos. O estimador ρ–score da Equação 2.3 é o único a satisfazer todas as restrições teóricas, razão pela qual é o método que será empregado para medir a dificuldade global de consultar conjuntos de dados reais na parte experimental desse trabalho.

2.2.2 Dimensionalidade intrínseca local

Mesmo em conjuntos de dados com dimensionalidade intrínseca alta, *i.e.*, conjuntos "difíceis" de serem consultados com apoio de índices, é possível encontrar elementos pontuais com vizinhos distinguíveis e, consequentemente, que podem se beneficiar do uso de partições métricas [4,6,45]. A medida de *dimensionalidade intrínseca local* (LID) permite separar elementos para os quais as distâncias dos demais objetos do conjunto de dados são representativas, ou ainda elementos "fáceis" de consultar, e ordená-los em uma escala de menor para maior valor. Essa separação é particularmente útil, pois permite detalhar o comportamento de algoritmos e estruturas de indexação métricas para além dos cenários de médio e pior caso, e indica que o uso dessas técnicas não depende apenas do conjunto de dados e da concentração de distâncias entre seus elementos, mas também do objeto de consulta e da cardinalidade do conjunto reposta.

Dado um elemento de consulta o_q e um número k de vizinhos, a dimensionalidade intrínseca local calcula a representatividade da proximidade entre o_q e seu k-ésimo vizinho, que é relativizada pelo número de elementos entre os dois objetos (outros vizinhos) e suas distâncias a o_q. Valores de LID que indicam uma densa vizinhança, *i.e.*, pequenas diferenças relativas de distância, têm como consequência uma maior probabilidade tanto de *(i)* o k-ésimo vizinho não ser único, quanto de *(ii)* não ser possível distinguir elementos próximos no contexto da consulta, o que inviabiliza o uso de estruturas de índice. Assim

como no caso da dimensionalidade intrínseca, um método de estimativa de LID adequado deve ser capaz de aproximar o valor da LID a partir das distâncias discretas observadas entre o elemento de consulta e sua k vizinhança [4, 87].

O método de estimativa proposto por Amsaleg *et al.* (2015), denominado *Maximum-Likelihood Estimator* (MLE), utiliza as premissas da variância relativa e do método ρ–score para aproximar o valor da LID de uma consulta k-$NN(o_q, k)$ de um objeto o_q para seus $o_1, o_2, \ldots o_k \in \mathcal{O}$ vizinhos, tal como expresso na Equação 2.4.

$$LID(o_q, k, \delta, \mathcal{O}) = -\left(\frac{1}{k} \cdot \sum_{i=1}^{k} \ln \frac{\delta(o_q, o_i)}{\delta(o_q, o_k)}\right)^{-1} \tag{2.4}$$

A Equação 2.4 usa uma razão inversa e, portanto, uma estimativa MLE de baixo valor de LID indica um objeto que é de "fácil" consulta, *i.e.*, possui vizinhos facilmente distinguíveis por distância, enquanto que uma estimativa MLE de alto valor indica um objeto de "difícil" consulta, *i.e.*, distâncias concentradas. A estabilidade do método é discutida experimentalmente no estudo de Amsaleg *et al.* (2019) e em diversos estudos experimentais que comparam a capacidade do estimador em discernir faixas relevantes de LID na comparação com outros métodos da literatura [4, 45, 87]. Os resultados indicam que o método é estável e possui uma boa capacidade de estimativa de LID para amostras de conjuntos de dezenas de milhares de elementos com o parâmetro de cem vizinhos [16].

Essa aproximação da LID possibilita não apenas indicar quais consultas podem se beneficiar de índices métricos, mas também organizar a avaliação experimental com mais critérios ao separar os elementos testados de acordo com sua "dificuldade", o que torna as análises experimentais de algoritmos e estruturas de indexação mais justas e coerentes em se tratando de espaços de alta dimensionalidade[7]. Estudos recentes [4,6] têm reexaminado experimentalmente o desempenho de algoritmos e estruturas de índice para consultas por vizinhança à luz de experimentos projetados para tratar de forma mais equânime elementos de consulta "fáceis" e "difíceis". Os resultados encontrados mostram que considerar apenas o valor da média de um grupo aleatório (*batch*) de elementos consultados acaba por não destacar as diferenças entre as estruturas e algoritmos comparados. Da mesma forma, os resultados indicam que usar apenas a estimativa de LID não é suficiente, por si só, para caracterizar o comportamento de algoritmos de busca por vizinhança.

[7]Usualmente encontra-se reportado a *média* do tempo de consulta/número de comparações por distância. Como a média é uma medida sensível a valores extremos, elementos de "difícil" consulta influenciam diretamente o resultado final de comparações do desempenho de estruturas de dados e algoritmos de busca para conjuntos imersos em espaços de alta dimensionalidade.

Assim, nesse trabalho, serão usadas as estimativas de *(i)* dimensionalidade intrínseca global, ou *(ii)* de LID por objeto de consulta, para a criação e análise de *batches* de teste sempre que conjuntos de dados do mundo real imersos em espaços de alta dimensionalidade forem usados para obtenção de uma ou mais medidas de comparação experimental.

2.3 Conclusões parciais

Esse capítulo apresentou conceitos importantes da modelagem por Espaços Métricos para o desenvolvimento da proposta, tanto da perspectiva do arcabouço teórico quanto dos critérios usados para a avaliação experimental. De forma resumida, os principais pontos aqui levantados e que serão usados na elaboração do novo método de consulta são:

- Consultas por vizinhança são uma variação de consultas por abrangência que permitem controlar a cardinalidade do conjunto resposta;
- Algoritmos de busca por vizinhança também precisam resolver com eficiência o problema de ordenação;
- VP-Trees e variantes são estruturas baratas de construir, fáceis de ajustar e que permitem a execução otimizada de consultas por similaridade quando há boa quantidade de memória disponível;
- O algoritmo *distance-browsing* é o algoritmo estado-da-arte que possui maior potencial para otimizar consultas por vizinhança sempre que houver boa quantidade de memória disponível;
- Consultas por vizinhança em espaços de alta dimensionalidade são um caso limite onde índices métricos tendem a entregar baixo desempenho; e
- O uso da dimensionalidade intrínseca local para separar e categorizar objetos de busca permite comparar e avaliar com mais detalhes o comportamento de algoritmos de busca e estruturas de indexação em espaços de alta dimensionalidade.

Capítulo 3

Consultas por Diversidade

3.1 Diversificação de resultados

Por mais que critérios de consultas por similaridade permitam recuperar objetos que são similares mas não idênticos, eles ainda são demasiado estritos para consultas com caráter exploratório a partir de um objeto de referência. Critérios que adicionam um nível de diversidade entre os elementos retornados são uma alternativa para essas buscas exploratórias, pois permitem que os elementos retornados não apenas guardem relação de proximidade com o objeto de consulta, mas que incluam diferentes conceitos ou representem diferentes grupos de objetos dentro do conjunto consultado.

Diversas definições para esses critérios de diversidade podem ser encontradas na literatura [28, 29, 103, 104]. Por exemplo, uma forma de modelar esse problema é considerar regiões de exclusão do espaço de busca que são construídas iterativamente a partir da inclusão de elementos no conjunto resposta. Nesse caso, cada elemento incluído leva consigo um pedaço do espaço de busca que é descartado da análise nos próximos passos.

A taxonomia de Drosou *et al.* (2017) define por diversidade a capacidade de recuperar por afinidade elementos de diferentes coleções do conjunto consultado de acordo com uma função objetivo genérica de diversidade f_{div}. Essa definição separa as abordagens existentes na literatura em três grandes grupos:

1. Métodos baseados em distância [12];
2. Métodos baseados em novidade [27, 50, 52, 71, 95, 96, 101, 105, 106]; e
3. Métodos baseados em cobertura [2, 14, 23, 46, 73, 77, 80, 84, 90, 100, 107].

Formalmente, essa categorização parte do pressuposto que a diversidade pode ser quantificada por uma função f_{div} que mede a qualidade de um conjunto resposta completo $\mathcal{R}$ construído para um objeto de consulta $o_q \in \mathbb{O}$ sobre um conjunto $\mathcal{O} \subseteq \mathbb{O}$ de acordo com um critério adicional de busca como, por exemplo, vizinhança. A Equação 3.1 apresenta essa formulação para um conjunto de k vizinhos diversificados.

$$\mathcal{R} = \text{argmax}_{\mathcal{R}' \subseteq \mathcal{O},\ |\mathcal{R}'|=k}\ f_{div}(\mathcal{R}', o_q) \tag{3.1}$$

Uma primeira abordagem para o problema de otimização descrito na Equação 3.1 é considerar uma função objetivo global que desconsidera o objeto de consulta para encontrar o subconjunto $\mathcal{R}'$ que maximiza a Equação 3.1. Em linhas gerais, essa é a premissa dos métodos baseados em distância. Uma segunda abordagem é associar uma função de distância exclusivamente para diversidade e outra à similaridade, sendo que os valores obtidos por ambas são combinados de acordo como uma função de *trade-off* f_{div} que precisa ser maximizada. Essa premissa é a empregada em métodos baseados em novidade. Por último, f_{div} pode ser modelada com uma função de exclusão de regiões do espaço de busca, sendo maximizada a partir do maior descarte de porções do espaço de busca. Essa última premissa é utilizada como base dos métodos baseados em cobertura.

A Figura 3.1 apresenta uma linha do tempo agrupada por categoria dos principais métodos de diversificação de resultados revisados nesse trabalho. Para a sua construção foram revisadas no início desse trabalho (final do ano de 2018) as bases de artigos Google

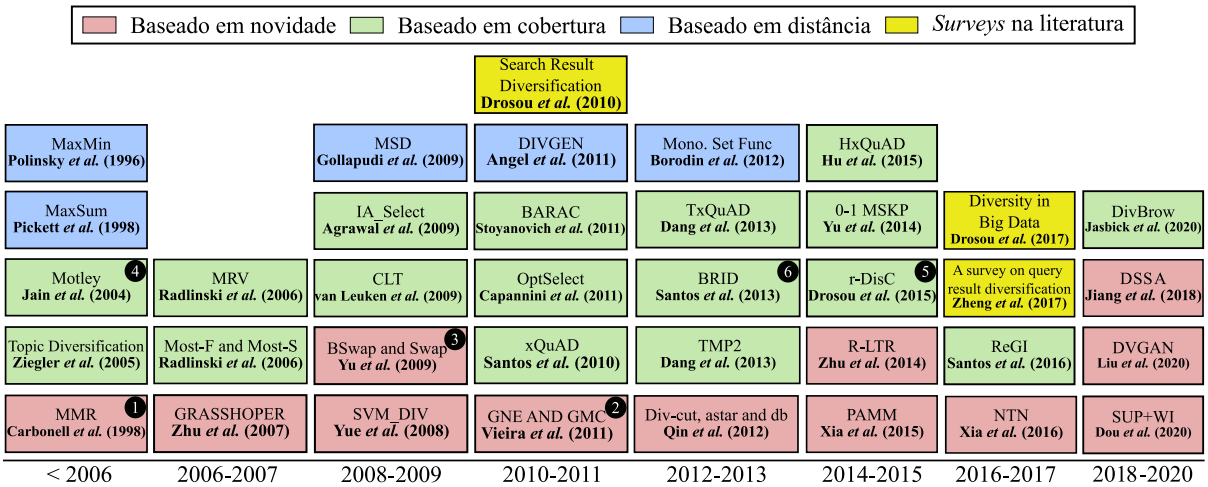


Figura 3.1: Linha do tempo agrupada de acordo com a taxonomia em [28] com os métodos de diversificação de resultados revisados nesse trabalho. Os métodos enumerados guardam maior relação com a proposta e são revisados em detalhes nessa seção.

Scholar[1] e DBLP[2] com os dois termos chave para consultas por similaridade diversificada em inglês *result diversification* e *diversified similarity searching*, sendo que os resultados foram *(i)* filtrados para eliminar métodos derivados de outras áreas que não guardam relação com a Teoria de Espaços Métricos e *(ii)* separados por número de citação maior ou igual a dez. Os melhores 30 artigos foram incluídos na linha do tempo, além de quatro novos estudos que surgiram depois do início desse trabalho.

3.1.1 Métodos baseados em distância

As abordagens dessa categoria modelam a função objetivo f_{div} como uma função global sobre $\mathcal{O}$ desconsiderando o objeto de consulta. Dessa forma, o problema torna-se encontrar a combinação de k elementos de $\mathcal{O}$ que maximizam um determinado critério de distâncias agregadas. Exemplos desses critérios incluem a soma das distâncias máximas (MaxSum) entre os k elementos e a soma das distâncias mínimas (MaxMin), dentre outras. Formalmente, o método de diversidade MaxMin [70] representa o argumento f_{div} da Equação 3.1 como $f_{div}(\mathcal{R}', o_q) = 1/|\mathcal{R}'|^2 \cdot \sum_{o_i,o_j \in \mathcal{R}'} \delta(o_i, o_j)$, enquanto que o método MaxSum [67] formula o mesmo argumento como $f_{div}(\mathcal{R}', o_q) = \min_{o_i,o_j \in \mathcal{R}', o_i \neq o_j} \delta(o_i, o_j)$. Ambas as modelagens podem ser reduzidas ao problema da p-dispersão, cuja solução computacionalmente ótima foi estudada extensivamente em análise numérica [68, 91].

Outros métodos representativos da categoria baseado em distância incluem o algoritmo MaxSumDispersion (MSD) [41] e o algoritmo DIVGEN [5]. O método MSD busca maximizar a soma de duas funções associadas a similaridade e diversidade, respectivamente, utilizando uma função linear que pondera a combinação de ambas as funções. No entanto, essas funções são globais e diferentes das funções de distância entre um par de objetos. Por outro lado, o algoritmo DIVGEN realiza uma série de acessos ao conjunto de dados para construir limiares inferiores e superiores de acordo com a *função de utilidade* dos elementos, reduzindo o número de elementos candidatos a cada iteração até que a utilidade dos k elementos encontrados sejam superiores às dos elementos descartados.

Os métodos baseados em distância têm baixos custos computacionais, porém apresentam um grave problema semântico ao desconsiderar a localidade da busca e retornar o mesmo conjunto resposta independentemente do objeto de consulta. Essa característica impede uma análise exploratória da perspectiva do objeto consultado, um ponto que é tratado tanto por métodos baseados em novidade quanto por cobertura.

[1] Disponível em `scholar.google.com.br/`.
[2] Disponível em `dblp.org/`.

3.1.2 Métodos baseados em novidade

Algoritmos baseados em novidade usam diretamente a função f_{div} como uma pontuação a ser maximizada na avaliação do conjunto diversificado $\mathcal{R}'$. A essa pontuação são associadas explicitamente funções de distância δ e δ_{div} (que podem ser iguais $\delta = \delta_{div}$) à similaridade e a diversidade, respectivamente. Comumente, f_{div} é modelado como uma combinação linear entre similaridade e diversidade, de acordo com um escalar externo $\lambda, \lambda \in [0, 1]$ fornecido pelo usuário como na Equação 3.2.

$$f_{div}(\mathcal{R}', o_q) = (k-1) \cdot (1-\lambda) \cdot \sum_{o_i \in \mathcal{R}'} \left(\delta(o_q, o_i) + 2 \cdot \lambda \cdot \sum_{o_j \in \mathcal{R}', o_i \neq o_j} \delta_{div}(o_i, o_j) \right) \tag{3.2}$$

Encontrar o conjunto reposta exato $\mathcal{R} = \mathcal{R}'$ com a maior pontuação é um problema NP-difícil [25, 28] e, portanto, abordagens heurísticas [81] são empregadas na prática. Essas abordagens procuram encontrar conjuntos resposta suficientemente diversificados e, para tal, reduzem o número de candidatos ao conjunto $\mathcal{R}'$ por meio de uma amostra $\mathcal{O}' \subseteq \mathcal{O}$ do conjunto original, tornando a execução desses métodos um processamento de duas fases cujo primeiro estágio é a amostragem [28, 91].

Um exemplo dessa classe de algoritmos heurísticos baseados em novidade é o *Maximal Marginal Relevance* (MMR – ❶) [15]. O MMR procura o vizinho mais próximo ao objeto de consulta e o escolhe como o primeiro vizinho diversificado, incluindo-o no conjunto resposta parcial $\mathcal{R}'$. A partir desse ponto, o método avalia cada elemento $o_i \in \mathcal{O} \setminus \mathcal{R}'$ de acordo com a Equação 3.3. O elemento com maior valor $MMR(o_i, o_q)$ é escolhido como próximo vizinho diversificado e o algoritmo reinicia a avaliação dos elementos em $\mathcal{O} \setminus \mathcal{R}'$ nas próximas $k-2$ iterações.

$$MMR(o_i, o_q) = (1-\lambda) \cdot \delta(o_i, o_q) + \frac{\lambda}{\mathcal{R}'} \cdot \sum_{o_j \in \mathcal{R}'} \delta_{div}(o_i, o_j) \tag{3.3}$$

O Algoritmo 4 apresenta a solução para uma consulta por vizinhança diversificada de acordo com o método MMR, onde ao final de cada uma das k iterações, o elemento o_i com o maior de $MMR(o_i, o_q)$ é incluído no conjunto resposta parcial $\mathcal{R}'$.

A Figura 3.2(a – f) mostra um teste-de-mesa reduzido para o Algoritmo 4 considerando um conjunto de dados de duas dimensões, $\delta = \delta_{div} = L_2$, uma vizinhança diversificada de $k = 6$ e um parâmetro linear de diversidade $\lambda = 0{,}25$ para o elemento de consulta o_6.

Dados: Conjunto $\mathcal{O}$, k vizinhos, objeto de consulta o_q e função de distância δ.
Retorno: Conjunto resposta $\mathcal{R}$.

$\mathcal{R}' \leftarrow \emptyset$;
while $|\mathcal{R}'| < k \wedge \mathcal{O} \setminus \mathcal{R} \neq \emptyset$ **do**
  $o_i \leftarrow \{o_i \in \mathcal{O} \setminus \mathcal{R}' \mid o_j \in \mathcal{O} \setminus \mathcal{R}', MMR(o_i, o_q) \geq MMR(o_j, o_q)\}$;
  $\mathcal{R}' \leftarrow \mathcal{R}' \cup \{o_i\}$;
return $\mathcal{R} \leftarrow \mathcal{R}'$;

Algoritmo 4: Algoritmo MMR para consultas por vizinhança. Adaptado de [15].

Para essa execução, o primeiro vizinho mais próximo o_6 é incluído no conjunto resposta na primeira iteração, seguido pelo elemento o_7 pois sua função objetivo MMR tem o maior valor na comparação com os demais objetos do conjunto de dados – Figura 4(b). Nas próximas iterações, são escolhidos os elementos o_8, o_9, o_{17} e o_{19} que obtiveram os valores de MMR de 2,33, 4,53, 7,09 e 9,45, respectivamente – Figura 4(f).

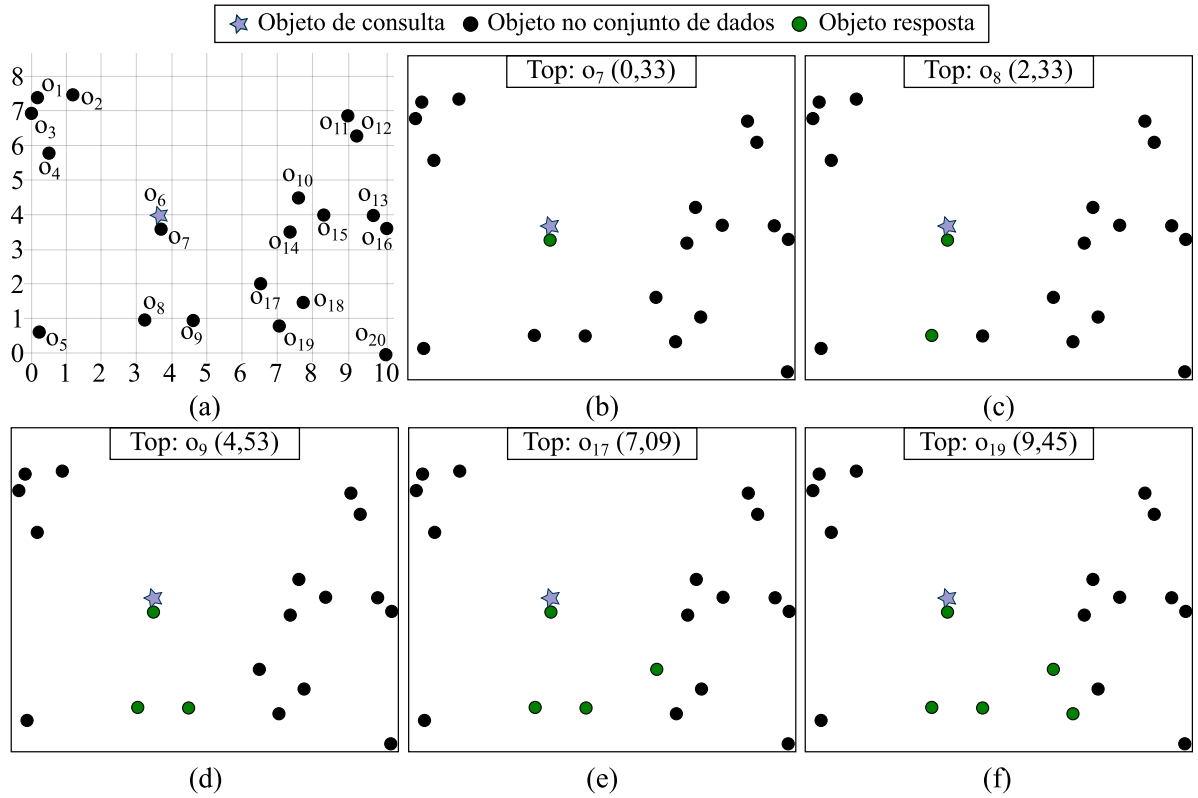


Figura 3.2: Exemplo de uma consulta MMR para $k = 6, o_q = o_6$ e $\lambda = 0{,}25$.

O algoritmo *Greedy Marginal Contribution* (GMC – ❷) [91] estende o método MMR ao ponderar o também impacto dos elementos deixados fora do conjunto resposta parcial $\mathcal{R}'$ com a função MMC. Essa ponderação considera uma terceira função de distância δ_{div_2} que mede a contribuição dos $k - |\mathcal{R}'|$ maiores valores para elementos deixados de fora do conjunto resposta parcial $\delta_{div_2}(o_i, o_j) : o_j \in \mathcal{O} \setminus \mathcal{R}'$ como na Equação 3.4.

Dados: Conjunto $\mathcal{O}$, k vizinhos, objeto de consulta o_q e funções de distância $\delta, \delta_{div}, \delta_{div_2}$.
Retorno: Conjunto resposta $\mathcal{R}$.

```
1 R' ← ∅;
2 while |R'| < k ∧ O \ R' ≠ ∅ do
3 |   o_i ← {o_i ∈ O \ R' | o_j ∈ O \ R', MMC(o_i, o_q) ≥ MMC(o_j, o_q)};
4 |   R' ← R' ∪ {o_i};
5 return R ← R';
```

Algoritmo 5: Algoritmo GMC para consultas por vizinhança. Adaptado de [91].

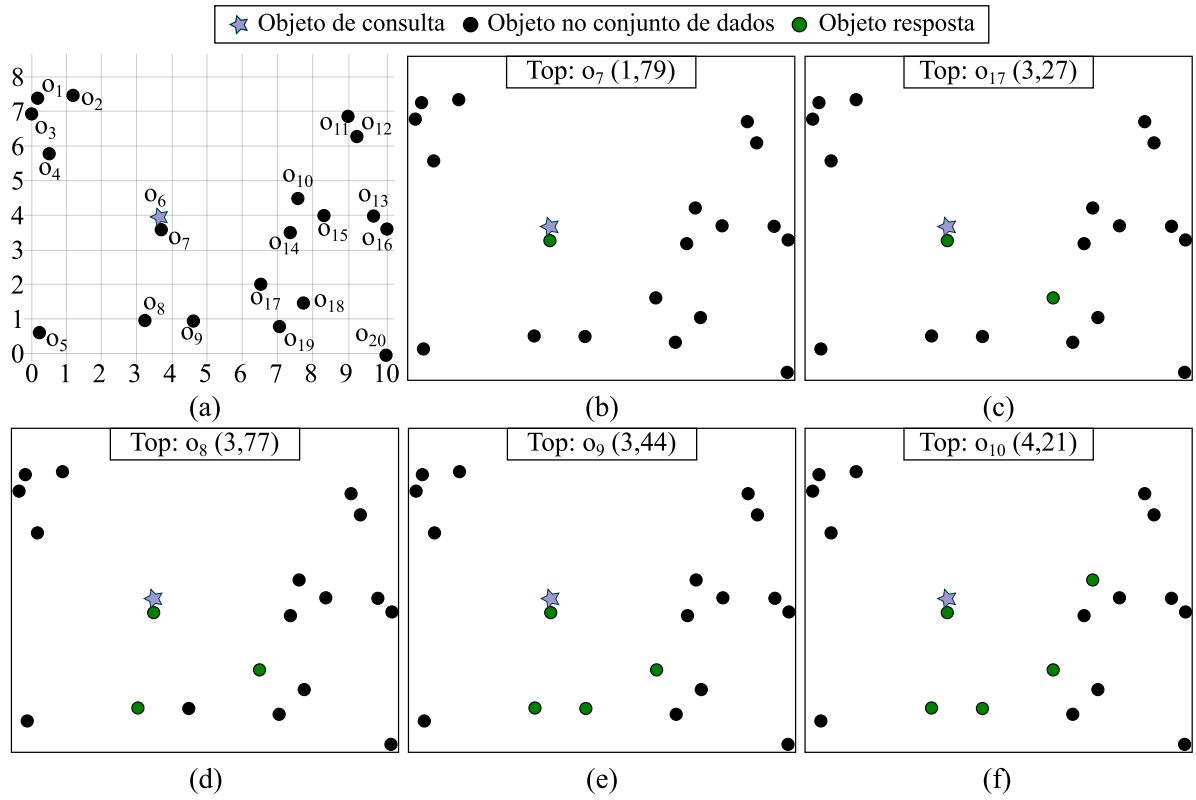


Figura 3.3: Exemplo de uma consulta GMC para $k = 6, o_q = o_6$ e $\lambda = 0{,}25$.

$$MMC(o_i, o_q) = (1-\lambda)\cdot\delta(o_i, o_q) + \frac{\lambda}{(k-1)}\cdot\left(\sum_{o_j \in \mathcal{R}'} \delta_{div}(o_i, o_j) + \sum_{o_j \in \mathcal{O}\setminus\{o_i\}}^{j \leq k-|\mathcal{R}'|} \delta_{div_2}(o_i, o_j)\right) \quad (3.4)$$

O Algoritmo 5 mostra a rotina de execução de uma consulta por vizinhança com o método GMC, enquanto a Figura 3.3 apresenta um teste-de-mesa para uma consulta GMC com os mesmos parâmetros da consulta da Figura 3.2 e $\delta_{div} = \delta_{div_2}$. Depois de o_6, o elemento o_7 é incluído como o elemento que maximiza a função MMC. Nas próximas iterações, são selecionados os elementos o_{17}, o_8, o_9 e o_{10}, respectivamente.

Dados: Conjunto $\mathcal{O}$, k vizinhos, objeto de consulta o_q e funções de distância $\delta, \delta_{div}, \delta_{div_2}$.
Retorno: Conjunto resposta $\mathcal{R}$.

$\mathcal{R} \leftarrow \mathcal{R}' \leftarrow \emptyset$;
for $i \leftarrow \{1, 2, \ldots,$ `max_iterações`$\}$ **do**
  $\mathcal{R}' \leftarrow$ GNE-Construção$(\mathcal{O} \setminus \mathcal{R}', k, o_q, \delta)$;
  $\mathcal{R}' \leftarrow$ GNE-BuscaLocal$(\mathcal{O}, \mathcal{R}', k, o_q, \delta)$;
  if $\sum_{o_i \in \mathcal{R}'} MMC(o_i, o_q) > \sum_{o_i \in \mathcal{R}} MMC(o_i, o_q)$ **then** $\mathcal{R} \leftarrow \mathcal{R}'$;
return $\mathcal{R}$;

GNE-Construção$(\mathcal{O}, k, o_q, \delta)$;

Dados: Conjunto $\mathcal{O}$, k vizinhos, objeto de consulta o_q e função de distância δ.
Retorno: Conjunto resposta parcial $\mathcal{R}'$.

$\mathcal{R}' \leftarrow \emptyset$;
while $|\mathcal{R}'| < k$ **do**
  $o_{max} \leftarrow \{o_i \in \mathcal{O} \setminus \mathcal{R}' \mid o_j \in \mathcal{O} \setminus \mathcal{R}', MMC(o_i, o_q) \geq MMC(o_j, o_q)\}$;
  $o_{min} \leftarrow \{o_i \in \mathcal{O} \setminus \mathcal{R}' \mid o_j \in \mathcal{O} \setminus \mathcal{R}', MMC(o_i, o_q) \leq MMC(o_j, o_q)\}$;
  $\mathcal{C} \leftarrow \{o_j \in \mathcal{O} \mid MMC(o_j, o_q) \geq (o_{max} - \alpha(o_{max} - o_{min}))\}$;
  $o_i \leftarrow$ `escolha_aleatória`$(\mathcal{C})$;
  $\mathcal{R}' \leftarrow \mathcal{R}' \cup \{o_i\}$;
return $\mathcal{R}'$;

GNE-BuscaLocal$(\mathcal{O}, \mathcal{R}', k, o_q, \delta)$;

Dados: Conjuntos $\mathcal{O}$ e $\mathcal{R}'$, k vizinhos, objeto de consulta o_q e função de distância δ.
Retorno: Conjunto resposta parcial $\mathcal{R}'$.

for $o_i \in \mathcal{R}'$ **do**
  $\mathcal{R}'' \leftarrow \mathcal{R}'$;
  for $o_j \in \mathcal{R}' \wedge o_j \neq o_i$ **do**
    $\mathcal{A} \leftarrow \emptyset$;
    while $|\mathcal{A}| \leq k - 1$ **do**
      $o_l \leftarrow \{o_l \in \mathcal{O} \setminus \mathcal{A} \mid o_m \in \mathcal{O} \setminus \mathcal{A}, \delta_{div_2}(o_i, o_l) \geq \delta_{div_2}(o_i, o_m)\}$;
      $\mathcal{A} \leftarrow \mathcal{A} \cup \{o_l\}$;
      if $o_l \notin \mathcal{R}'$ **then**
        $\mathcal{R}''' \leftarrow (\mathcal{R}'' \setminus \{o_j\}) \cup \{o_l\}$;
        if $\sum_{o_i \in \mathcal{R}'''} MMC(o_i, o_q) > \sum_{o_i \in \mathcal{R}''} MMC(o_i, o_q)$ **then** $\mathcal{R}'' \leftarrow \mathcal{R}'''$;
  if $\sum_{o_i \in \mathcal{R}''} MMC(o_i, o_q) > \sum_{o_i \in \mathcal{R}'} MMC(o_i, o_q)$ **then** $\mathcal{R}' \leftarrow \mathcal{R}''$;
return $\mathcal{R}'$;

Algoritmo 6: Algoritmo GNE baseado em heurística GRASP para consultas por vizinhança diversificada. Adaptado de [91].

Outra extensão da abordagem MMR é o algoritmo *Greedy Randomized with Neighborhood Expansion* (GNE – ❷) [91] que usa uma meta-heurística *Greedy Randomized Adaptive Search Procedure* (GRASP) [74] para evitar potenciais mínimos locais oriundos

da avaliação incremental de $\mathcal{R}'$ pelo método GMC. A Algoritmo 6 apresenta a resolução de uma consulta por vizinhança diversificada com o método GNE de acordo com as etapas da busca GRASP. A função de custo de diversidade f_{div} usado pelo método GNE é a mesma do método GMC (Equação 3.4), porém, durante a fase de construção a busca (Algoritmo 6 na sub-rotina GNE-Construção) ao invés de selecionar o elemento o_i com o maior valor de MMC em cada uma das $k-1$ iterações, seleciona um elemento aleatório entre os mais bem pontuados.

Esse raciocínio gera uma construção de duas fases, na qual uma fila de candidatos é mantida na memória e é refinada gulosamente por seleções aleatórias de elementos bem colocados. Para determinar o quão aleatória será a construção um parâmetro $\alpha, 0 \leq \alpha \leq 1$ precisa ser informado pelo usuário, sendo que a configuração $\alpha = 0$ faz com que o GNE se comporte de forma equivalente ao GMC e a configuração $\alpha = 1$ torna a construção GNE em uma escolha completamente aleatória. Por fim, durante a fase de busca local (Algoritmo 6 na sub-rotina GNE-BuscaLocal), o conjunto resposta é iterativamente aprimorado com a aplicação de modificações locais na vizinhança do conjunto resposta parcial $\mathcal{R}'$. Para isso, a fase de busca local executa uma série de trocas entre elementos de $\mathcal{R}'$ pelos elementos que mais aumentam a diversidade de $\mathcal{R}'$.

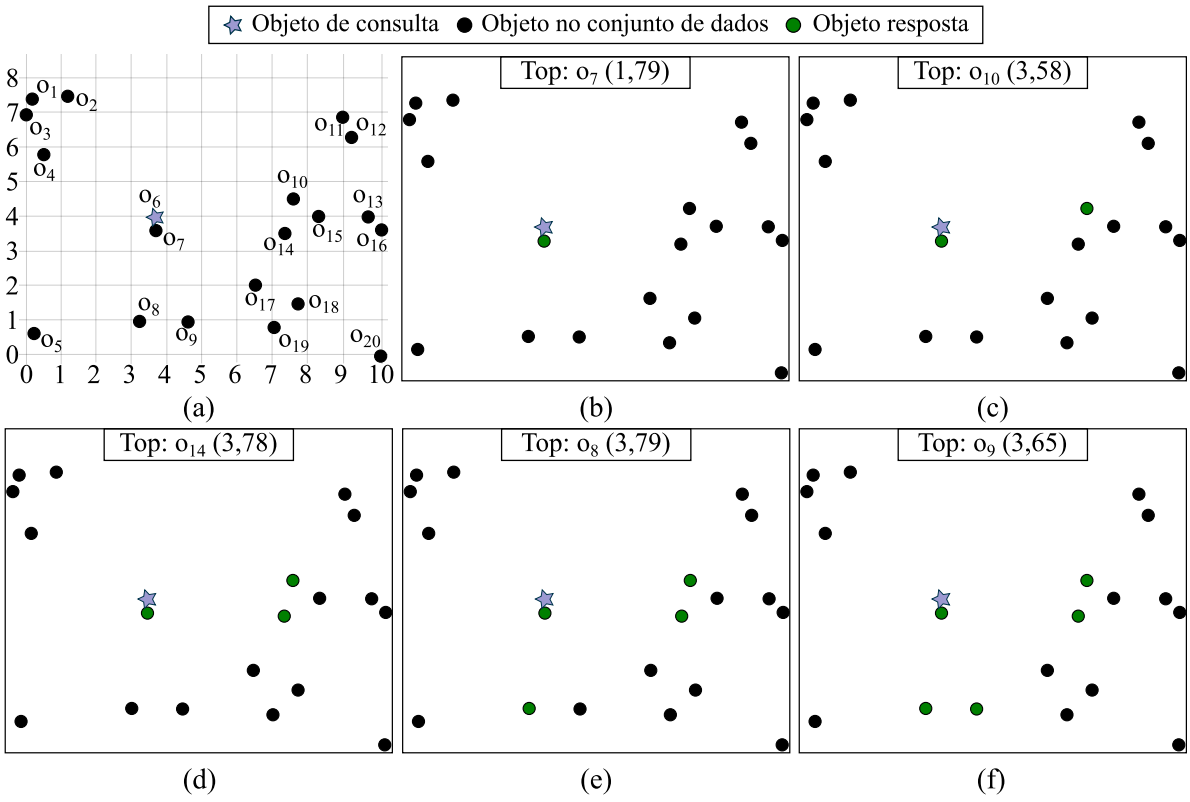


Figura 3.4: Exemplo de uma consulta GNE para $k = 6, o_q = o_6$ e $\lambda = 0{,}25$.

A Figura 3.4 mostra um exemplo de execução para uma consulta por vizinhança diversificada com os mesmos parâmetros de busca da consulta na Figura 3.3. O primeiro elemento selecionado pela execução é o vizinho mais próximo ao objeto de consulta o_6, seguido do o_7, cujo valor da função objetivo é 1,79 – Figura 3.4(b). Em seguida, são retornados os elementos o_{10}, o_{14}, o_8 e o_9 cujos valores da função objetivo são 3,58, 3,78, 3,79 e 3,65, respectivamente – Figuras 3.4(c–f).

Outro algoritmo que busca evitar o máximo local de busca é o método Swap (❸) [99] e sua variante BSwap. Esse método realiza um processamento em duas fases para recuperar elementos que aumentem uma função objetivo, que pode ser a mesma empregada no critério GMC dada na Equação 3.4. Inicialmente, um conjunto resposta com os objetos que maximizam a função objetivo é selecionado emulando o algoritmo GMC. Na sequência, uma série de trocas iterativas é realizada no conjunto resposta, sendo cada objeto do conjunto de dados permutado ao menos uma vez. A cada iteração, o elemento no conjunto resposta que contribui menos para a diversidade é trocado pelo elemento que aumenta a função objetivo, até que todos os elementos do conjunto consultado sejam avaliados.

O Algoritmo 7 apresenta uma rotina para a execução do método Swap acoplado à função objetivo MMC, enquanto a Figura 3.5 mostra o passo-a-passo da execução dessa rotina para uma consulta exemplo com os mesmos parâmetros de busca da Figura 3.3. Inicialmente, são selecionados para o conjunto parcial os elementos mais similares ao objeto de consulta o_q, a saber: o_6, o_7, o_8, o_9, o_{17} e o_4 – (Figura 3.5(b)). Na sequência, o

Dados: Conjunto $\mathcal{O}$, k vizinhos, objeto de consulta o_q e funções $\delta, \delta_{div}, \delta_{div_2}$.
Retorno: Conjunto resposta $\mathcal{R}$.

```
1  R ← A ← ∅;
2  while |R| < k ∧ O \ R ≠ ∅ do
3  |  o_i ← {o_i ∈ O \ R | o_j ∈ O \ R, δ(o_i, o_q) ≤ δ(o_j, o_q)};
4  |  R ← R ∪ {o_i};
5  while |A| ≤ |O| − k do
6  |  o_i ← {o_i ∈ O \ (R ∪ A) | o_j ∈ O \ (R ∪ A), δ(o_i, o_q) ≤ δ(o_j, o_q)};
7  |  A ← A ∪ {o_i};
8  |  R' ← R;
9  |  for o_j ∈ R do
10 |  |  if Σ_{o_m ∈ (R \ o_j) ∪ {o_i}} MMC(o_q, o_m) > Σ_{o_m ∈ R'} MMC(o_q, o_m) then
11 |  |  |  R' ← (R \ o_j) ∪ {o_i};
12 |  if Σ_{o_m ∈ R'} MMC(o_q, o_m) > Σ_{o_m ∈ R} MMC(o_q, o_m) then
13 |  |  R ← R';
14 |  return R;
```

Algoritmo 7: Algoritmo para consultas por vizinhança Swap. Adaptado de [99].

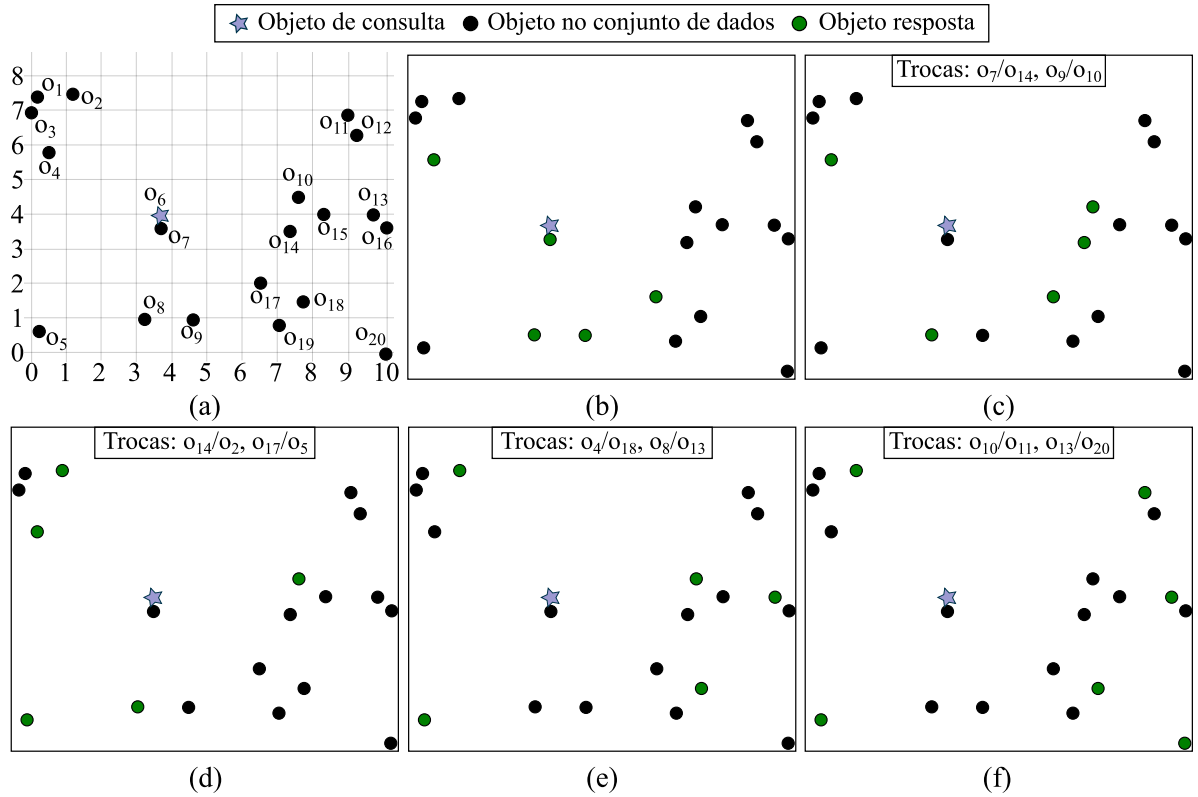


Figura 3.5: Exemplo de uma consulta Swap para $k = 6, o_q = o_6$ e $\lambda = 0{,}25$.

elemento que menos contribui para a diversidade segundo a função MMC (o_7) é trocado pelo elemento o_{14}. Esse procedimento é repetido e realizas as trocas de o_9 por o_{10}, o_{14} por o_2, o_{17} por o_5, o_4 por o_{18}, o_8 por o_{13}, o_{10} por o_{11} e o_{13} por o_{20}, encerrando a execução do Algoritmo 7 – Figuras 3.5(c–f).

A variante do algoritmo Swap, o método BSwap emprega um outro critério de troca de elementos a cada iteração. Nesse algoritmo, o elemento com maior valor de função objetivo MMC é permutado com o elemento que contribui menos para a diversidade até que todos os elementos sejam avaliados.

3.1.3 Métodos baseados em cobertura

Métodos de diversificação de resultados baseados em cobertura modelam a função f_{div} a partir de regiões de exclusão geradas por elementos incluídos no conjunto resposta parcial. A motivação desses métodos é eliminar candidatos que estão "cobertos" por uma área que já se encontra representada pelo elemento no conjunto resposta.

Da perspectiva da Teoria de Espaços Métricos, essa "cobertura"é diretamente modelada por raio que, associado ao elemento no conjunto resposta, gera uma área de exclusão

no espaço de busca [33, 42]. Um primeiro método que usa essa estratégia para induzir diversidade durante a execução de uma busca por vizinhança é o algoritmo *Motley* (❹) [47]. Esse método executa uma busca gulosa para excluir regiões do espaço de busca a partir dos vizinhos mais próximos ao objeto de consulta de acordo com um raio fixo ξ informado pelo usuário. O Algoritmo 8 detalha essa rotina de busca e retorna os elementos vizinhos diversificados como o conjunto $\mathcal{R}$, eliminado os elementos cobertos.

A Figura 3.6 exemplifica a execução do algoritmo Motley para uma consulta com vizinhança $k = 5$, raio $\xi = 3{,}6$ e $\delta = L_2$. De acordo com o Algoritmo 8, o primeiro vizinho diversificado é o elemento mais próximo ao elemento de consulta – o_7. Todos os elementos

Dados: Conjunto $\mathcal{O}$, k vizinhos, objeto de consulta o_q, função de distância δ e raio ξ.
Retorno: Conjunto resposta $\mathcal{R}$.

1 $\mathcal{R} \leftarrow \emptyset$;
2 **while** $|\mathcal{R}| < k \wedge \mathcal{O} \setminus \bigcup_{o_m \in \mathcal{R}} Rg(o_m, \xi) \neq \emptyset$ **do**
3 $\quad o_i \leftarrow \{o_i \in \mathcal{O} \setminus \bigcup_{o_m \in \mathcal{R}} Rg(o_m, \xi) \mid o_j \in \mathcal{O} \setminus \bigcup_{o_m \in \mathcal{R}} Rg(o_m, \xi), \delta(o_i, o_q) \leq \delta(o_j, o_q)\}$;
4 $\quad \mathcal{R} \leftarrow \mathcal{R} \cup \{o_i\}$;
5 **return** $\mathcal{R}$;

Algoritmo 8: Algoritmo Motley para consultas por vizinhança. Adaptado de [47].

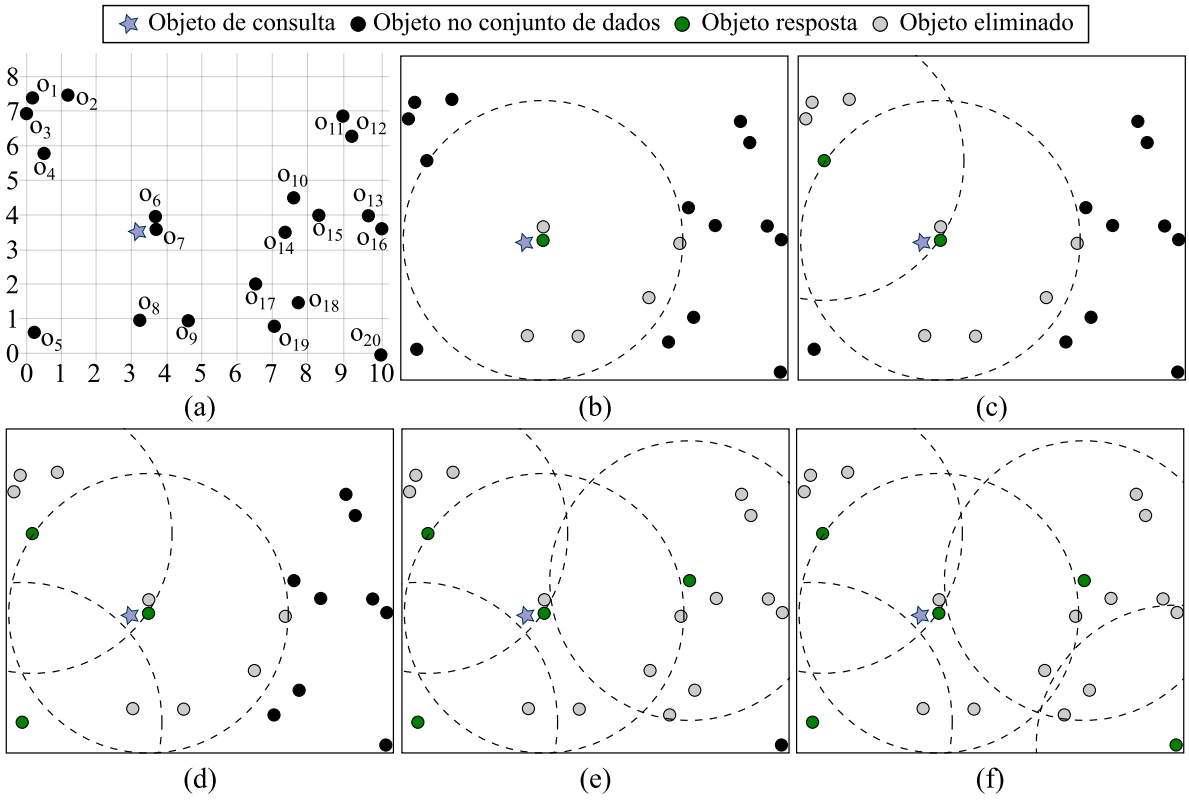


Figura 3.6: Exemplo de uma consulta Motley para $k = 5, \xi = 3{,}6$ e $\delta = L_2$.

que estejam dentro do conjunto resposta da consulta por abrangência $Rg(o_7, \xi)$ passam a fazer parte de uma região de exclusão e são automaticamente descartados do conjunto de busca $\mathcal{O} \setminus Rg(o_7, \xi)$. No exemplo da Figura 3.6(b), os elementos o_6, o_8, o_9, o_{17} e o_{14} estão dentro da região de exclusão delimitada por $Rg(o_7, \xi)$ e são eliminados da avaliação já na primeira iteração do algoritmo. Esse processo se repete em *loop* para os próximos vizinhos fora das regiões eliminadas até que um determinado número k de elementos seja selecionado ou que o conjunto de busca se torne vazio. No exemplo das Figuras 3.6(c–f), os elementos o_4, o_5, o_{10} e o_{20} são retornados após o_7, eliminando os candidatos que estão em regiões excluídas $Rg(o_4, \xi)$ e $Rg(o_{10}, \xi)$, respectivamente, enquanto as regiões $Rg(o_5, \xi)$ e $Rg(o_{20}, \xi)$ não eliminam nenhum elemento, visto que os elementos dentro dessas regiões já haviam sido descartados anteriormente.

Um método baseado em cobertura que estende o critério de exclusão da abordagem Motley (exclusão pelo raio ξ) é o algoritmo *r-DisC* (❺) [33]. O método r-DisC utiliza tanto o raio ξ para recuperar um conjunto $\mathcal{R}$ de elementos diversificados tal que $\forall\ o_i \in \mathcal{O}, \exists\ o_j \in \mathcal{R} \mid \delta(o_i, o_j) \leq \xi \wedge o_i \neq o_j$ (garantindo que pelo menos um elemento de $\mathcal{O}$ é coberto pela região de exclusão $Rg(o_j, \xi)$ além do próprio o_j), e que $\forall\ o_i, o_j \in \mathcal{R}, \delta(o_i, o_j) > \xi$ (garantindo que todos os elementos retornados são separados por mais do que ξ).

O Algoritmo 9 mostra uma rotina para a execução do método r-DisC para um valor de cobertura ξ informado pelo usuário. O algoritmo escolhe gulosamente o objeto $o_i \in \mathcal{O}$ que possui a maior quantidade de vizinhos dentro de um raio de cobertura e, então, exclui uma região do espaço de busca que corresponde a consulta por abrangência $Rg(o_i, \xi)$. Esse critério guloso é executado em *loop* até que o conjunto de busca se torne vazio ou todos os candidatos não tenham nenhum vizinho dentro de um raio de proximidade ξ.

Dados: Conjunto $\mathcal{O}$, função de distância δ e raio ξ.
Retorno: Conjunto resposta $\mathcal{R}$.

1 $\mathcal{R} \leftarrow \emptyset$;
2 **while** $\mathcal{O} \setminus \bigcup_{o_m \in \mathcal{R}} Rg(o_m, \xi) \neq \emptyset \wedge \exists\ o_m \in \mathcal{O} \setminus \bigcup_{o_m \in \mathcal{R}} Rg(o_m, \xi), |Rg(o_m, \xi)| > 1$ **do**
3 $\quad o_i \leftarrow \{o_i \in \mathcal{O} \setminus \bigcup_{o_m \in \mathcal{R}} Rg(o_m, \xi) \mid o_j \in \mathcal{O} \setminus \bigcup_{o_m \in \mathcal{R}} Rg(o_m, \xi), |Rg(o_i, \xi)| \geq |Rg(o_j, \xi)|\}$;
4 $\quad \mathcal{R} \leftarrow \mathcal{R} \cup \{o_i\}$;
5 **return** $\mathcal{R}$;

Algoritmo 9: Algoritmo r-DisC para consultas por vizinhança. Adaptado de [30].

A Figura 3.7 mostra um exemplo de execução do Algoritmo 9 para um raio $\xi = 2$ considerando o mesmo conjunto de dados da Figura 3.2. Inicialmente, o r-DisC seleciona o candidato o_{15} pois este é o elemento com o maior número de vizinhos (4) dentro do raio

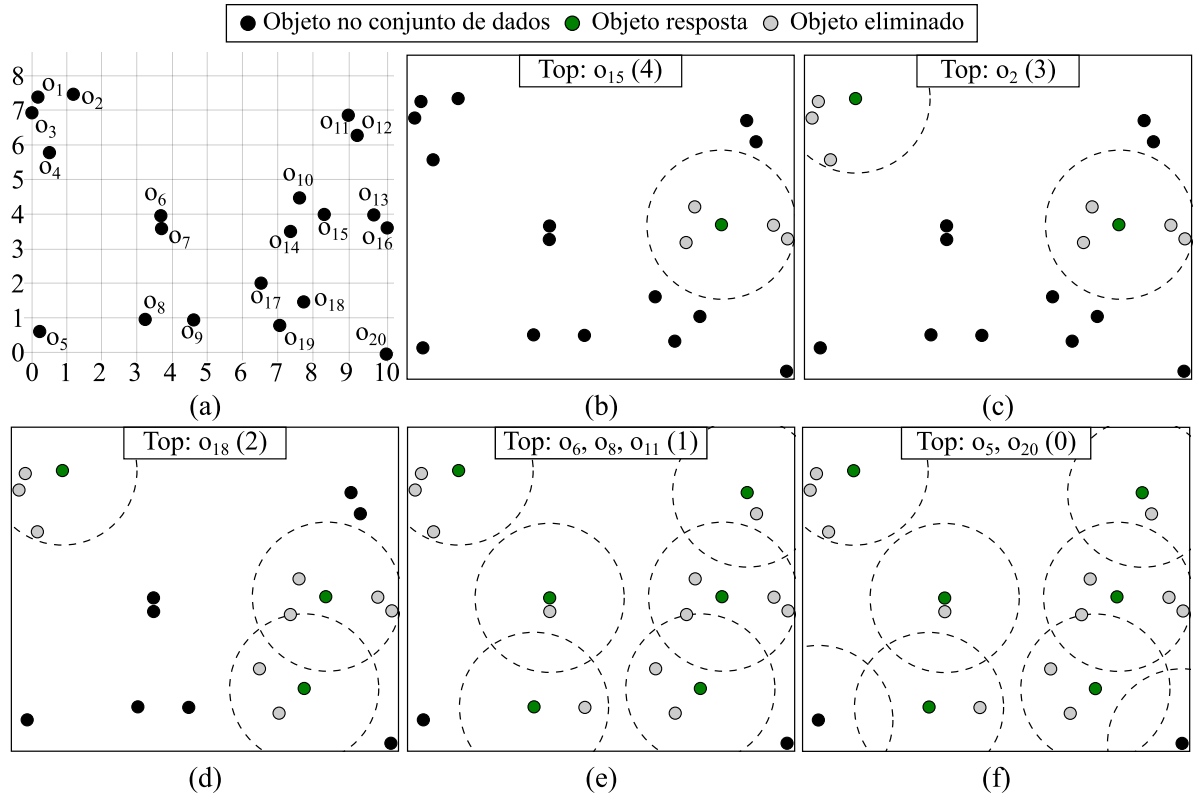


Figura 3.7: Exemplo de uma consulta r-DisC para $\xi = 2$ e $\delta = L_2$.

de cobertura $\xi = 2$. Após essa escolha, todos os elementos dentro da região delimitada por $Rg(o_{15}, 2)$ são excluídos do espaço de busca – Figura 3.7(b). Na próxima iteração, o candidato o_2 é selecionado como o elemento com o maior número de vizinhos (3) que são eliminados da busca – Figura 3.7(c). Esse processo se repete para o_{18} (com 2 vizinhos) e para o_6, o_8 e o_{11} (todos com 1 vizinho). Nesse ponto, a consulta é finalizada pois os elementos candidatos restantes o_5 e o_{20} não possuem vizinhos.

Embora o método r-DisC seja mais criterioso que o método Motley para excluir regiões *densas* do espaço de busca, *i.e.*, incluam alguma vizinhança além do elemento no conjunto resposta, ele não permite controlar a cardinalidade k do conjunto resposta nem capturar a diversidade considerando a perspectiva de um objeto de consulta. Além disso, tanto o método r-DisC quanto o Motley apresentam o inconveniente de necessitar de um raio de exclusão informado pelo usuário e definido de antemão o que, na prática, acaba por tornar-se uma etapa de pré-processamento e ajuste de parâmetros com custo computacional não-negligenciável [54, 104].

O método *Better Results with Influence Diversification* (BRID$_k$ – ❻) [79] fornece uma abordagem que não requer parâmetros externos do usuário, ao mesmo tempo que permite controlar a cardinalidade do conjunto resposta e capturar a perspectiva do objeto de

consulta. Esse método cria uma separação dinâmica entre objetos similares e diversificados através de um limiar de distância assumindo que o relacionamento ternário entre o conjunto de busca, o conjunto resposta e o objeto de busca não é independente. Dessa forma, o BRID$_k$ considera que elementos próximos se *influenciam* mutuamente em uma razão proporcional à distância entre eles. Essa intuição de relacionamento ternário define a base para a formalização do conceito de *Influência* que permite estender consultas por similaridade tradicionais, *e.g.*, vizinhança, em consultas por similaridade diversificada.

3.2 Consultas por similaridade diversificada

Formalmente, a Influência entre dois objetos no espaço de busca deve ser calculada de acordo com a Definição 4.

Definição 4 (Influência.) *Dados dois elementos* $o_i, o_j \in \mathcal{O} \subseteq \mathbb{O}, o_i \neq o_j$*, a Influência* $I(o_i, o_j)$ *entre* o_i *e* o_j *é calculada como* $I(o_i, o_j) = 1/\delta(o_i, o_j)$

A função de Influência permite ao método BRID$_k$ usar dois critérios de ordenação para com elementos candidatos o_i em relação ao objeto de consulta o_q: o primeiro por distância (função $\delta(o_i, o_q)$) e o segundo por Influência (função $I(o_i, o_q)$). Além disso, o BRID$_k$ combina essas ordenações com o uso da função de Influência para definir um critério de exclusão de regiões do espaço de busca que são denominadas *regiões influenciadas*.

Suponha que um candidato o_i foi incluído no conjunto resposta $\mathcal{R}$ de uma consulta por similaridade diversificada. O método BRID$_k$ usa a Influência desse elemento para os demais objetos do conjunto de dados consultado para construir um Conjunto de Influência Forte. Esse conjunto representa uma região influenciada no espaço de busca que deve ser descartada, pois todos os elementos ali dentro estão representados pelo candidato no conjunto resposta – *influenciador*. Formalmente, um Conjunto de Influência Forte é construído de acordo com a Definição 5.

Definição 5 (Conjunto de Influência Forte.) *Sejam* $o_q \in \mathbb{O}$ *um objeto de consulta e* $o_i \in \mathcal{R} \subseteq \mathcal{O} \subseteq \mathbb{O}$ *um elemento do conjunto resposta, o Conjunto de Influência Forte do par* $\langle o_q, o_i \rangle$ *é dado por* $\ddot{I}_{o_i,o_q} = \{o_j \in \mathcal{O} \setminus \mathcal{R} \mid I(o_i, o_j) \geq I(o_i, o_q) \wedge I(o_i, o_j) \geq I(o_j, o_q)\}$.

Portanto, para resolver uma consulta por similaridade diversificada, o método BRID$_k$ primeiro ordena os elementos candidatos por distância ao objeto de consulta e, na sequência,

varre essa lista ordenada de candidatos descartando os objetos que não estão dentro do Conjunto de Influência Forte de nenhum elemento incluído no conjunto resposta $\mathcal{R}$. Como $\mathcal{R}$ é, inicialmente, vazio, o vizinho mais próximo do objeto de consulta é sempre incluído no conjunto resposta como um *influenciador* e passa a definir a primeira região de exclusão no espaço de busca. O algoritmo 10 apresenta a rotina de execução BRID$_k$ para resolução de consultas por vizinhança diversificada, enquanto a Figura 3.8 mostra um exemplo de execução para o BRID$_k$ com parâmetros $k = 6$ e $\delta = L_2$.

Dados: Conjunto $\mathcal{O}$, quantidade de vizinhos diversificados k e objeto de consulta o_q.
Retorno: Conjunto resposta $\mathcal{R}$.

---

1 $\mathcal{R} \leftarrow \emptyset$;
2 **while** $|\mathcal{R}| < k \wedge \mathcal{O} \setminus \cup_{o_m \in \mathcal{R}} Rg(o_m, \delta(o_m, o_q)) \neq \emptyset$ **do**
3 $\quad o_i \leftarrow \{o_i \in \mathcal{O} \setminus \cup_{o_m \in \mathcal{R}} Rg(o_m, \delta(o_m, o_q)) \mid o_j \in \mathcal{O} \setminus \cup_{o_m \in \mathcal{R}} Rg(o_m, \delta(o_m, o_q)), \delta(o_i, o_q) \leq \delta(o_j, o_q)\}$;
4 $\quad$ **if** $\forall\, o_j \in \mathcal{R}, o_i \notin \ddot{I}_{o_j, o_q}$ **then**
5 $\quad\quad \mathcal{R} \leftarrow \mathcal{R} \cup \{o_i\}$`/* Inclui apenas elementos não influenciados */`
6 **return** $\mathcal{R}$;

Algoritmo 10: Algoritmo BRID$_k$ para consultas por vizinhança. Adaptado de [79].

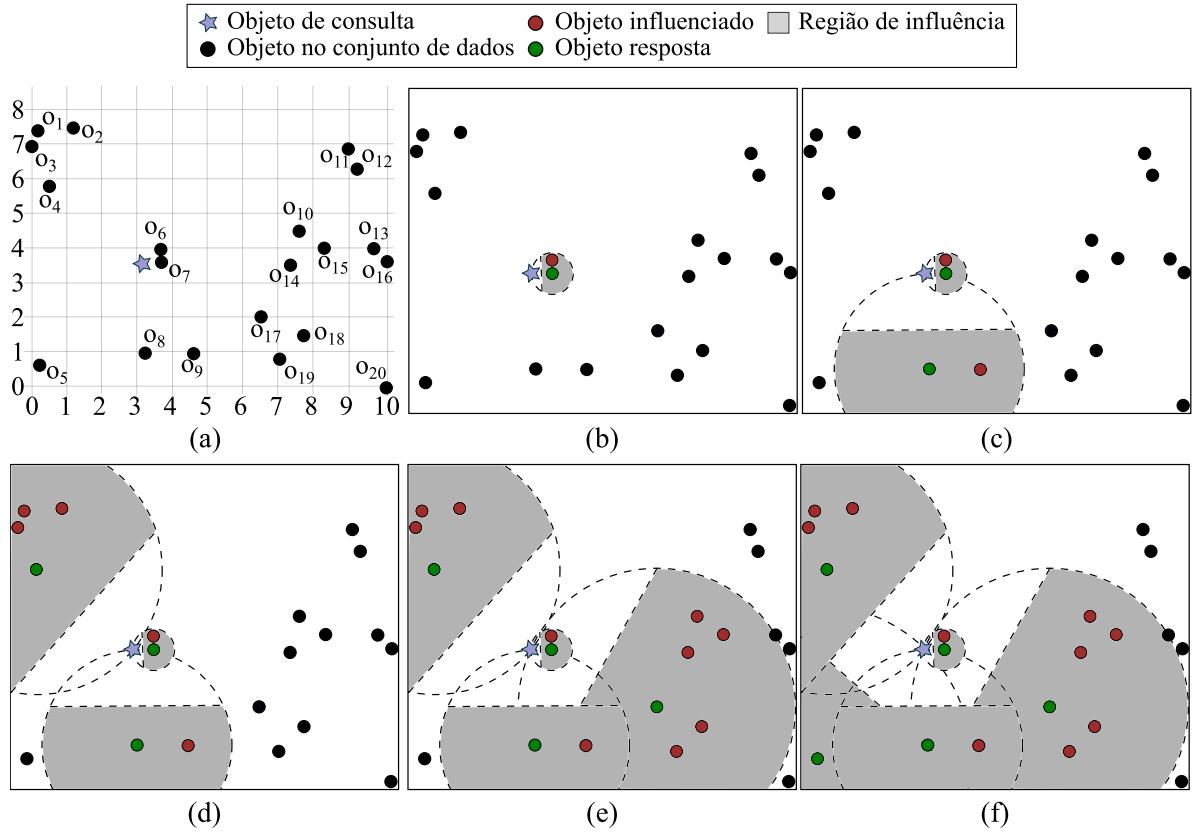


Figura 3.8: Exemplo de uma consulta BRID$_k$ para $k = 5$ e $\delta = L_2$.

Na primeira iteração, o algoritmo seleciona o vizinho mais próximo do objeto de consulta e o inclui no conjunto resposta tornando-o um elemento influenciador. Nesse caso a região influenciada é dada por $\langle o_7, \delta(o_7, o_q)\rangle$, gerando a região de exclusão no espaço de busca delimitada por $Rg(o_7, \delta(o_7, o_q))$ que elimina o_6. Na sequência, o próximo vizinho o_8 é selecionado como elemento candidato. Como o_8 não é influenciado por o_7, ele também é incluído no conjunto resposta e gera uma nova região de exclusão delimitada por $Rg(o_8, \delta(o_8, o_q))$. Na iteração seguinte, o vizinho mais próximo, o_9, é marcado como elemento candidato. No entanto, o_9 está influenciado por o_8 e é eliminado da busca como elemento influenciado – Figura 3.8(c). O próximo candidato é o vizinho o_4, que é incluído como influenciador e sua região de Influência é responsável por eliminar os candidatos o_3, o_2 e o_1 – Figura 3.8(d). O próximo vizinho não influenciado escolhido pelo algoritmo é o elemento o_{17}, cujo Conjunto de Influência Forte elimina os elementos o_{14}, o_{15}, o_{10}, o_{18} e o_{19} – Figura 3.8(e). Por fim, o próximo vizinho o_5 é selecionado, sendo que sua região de Influência não elimina nenhum outro candidato – Figura 3.8(f).

3.3 Comparação dos métodos revisados

Embora a taxonomia em [28] organize os diferentes métodos propostos na literatura para diversificação de resultados através da formulação geral da Equação 3.1, comparar de forma experimental e quantitativamente a qualidade desses métodos é uma tarefa muito mais complexa, e um ponto de pesquisa ainda em aberto [38, 54, 69]. Uma das principais dificuldades para essa comparação direta é o viés das próprias métricas de qualidade que tendem a favorecer determinados grupos de algoritmos de diversificação de resultados. Em particular, os resultados experimentais em [54] combinados com resultados de mineração visual de dados [32, 78] indicam que:

- Métricas orientadas à função objetivo tendem a favorecer os resultados de métodos baseados em novidade, *e.g.*, maximização de valores δ_{div} e MMR; e
- Métricas orientadas a agrupamento de dados tendem a favorecer métodos baseados em cobertura, *e.g.*, densidade e r-DisC.

Portanto, caso a escolha do método de diversificação de resultados dependa exclusivamente da semântica dos resultados esperados, deve ser levado em consideração o viés de construção dos conjuntos resposta de cada um dos algoritmos revisados. A Figura 3.9(a) mostra um exemplo prático desses vieses de construção para a realização de uma consulta

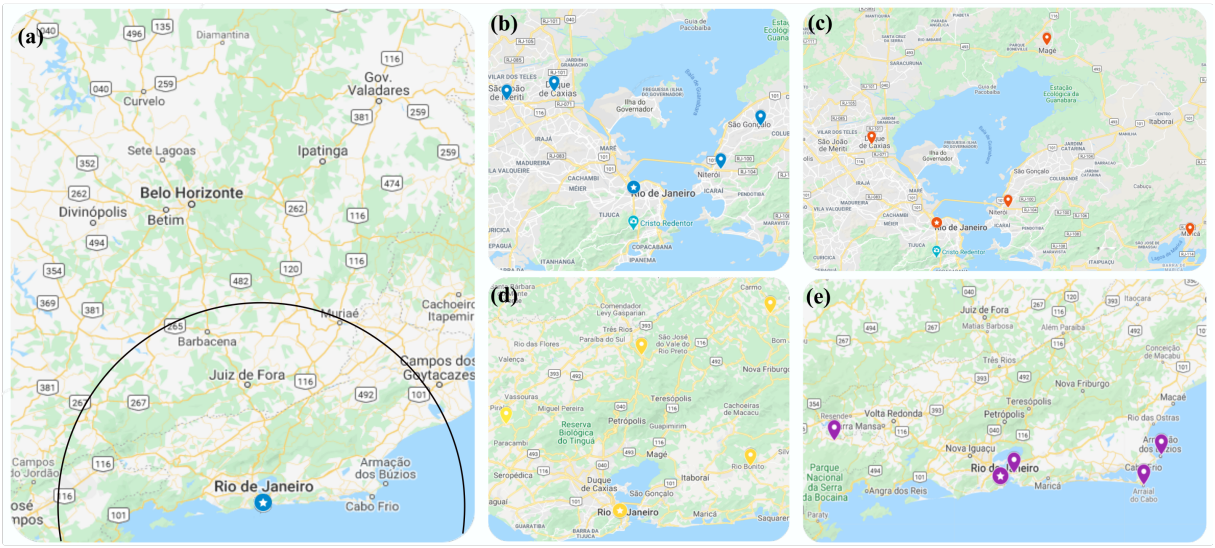


Figura 3.9: Consultas por vizinhança $k = 5$ de acordo com diferentes vieses de construção. (a) Delimitação do conjunto de busca e objeto de consulta. (b) Busca k-NN. (c) Busca BRID$_k$. (d) Busca Motley. (e) Busca GMC. O *trade-off* similaridade/diversidade aparece como o efeito de *zoom-out* nessa visualização.

por vizinhança sobre a área metropolitana da cidade do Rio de Janeiro/BR, passando de uma consulta por apenas similaridade a uma busca dominada por diversidade. Para uma busca pelas 05 cidades mais próximas ao Rio de Janeiro, uma consulta k-NN recupera as cinco cidades mais próximas que estão conurbadas com a capital estadual – Figura 3.9(b).

Por outro lado, o método BRID$_k$ adiciona novas cidades diversificadas à busca mantendo claro o critério de proximidade e a densidade de cidades ao redor de cada ponto no conjunto resposta – Figura 3.9(c), enquanto o método Motley permite recuperar cidades representativas do estado respeitada uma distância mínima – Figura 3.9(d). Finalmente, o método GMC pemite obter cidades nas bordas da região consultada, dando mais ênfase ao critério de diversidade à cidade consultada do que a proximidade –Figura 3.9(e).

Uma alternativa mais objetiva para a escolha do método de diversificação de resultado é por meio do custo e características do algoritmo de busca a ser executado e/ou acoplado a motores de busca que já são eficientemente usados para recuperar dados por identidade e similaridade. Dentro dessa perspectiva, destacam-se os critérios:

- Considerar o objeto de consulta: O algoritmo de execução da consulta por similaridade diversificada deve considerar o objeto de consulta como referência para o conjunto resposta, sendo sensível a localidade da busca;
- Resultado determinístico: O mesmo algoritmo executado duas vezes deve produzir a mesma resposta, garantindo a consistência como em buscas por identidade e ordem;

- Controle da cardinalidade: O algoritmo de busca deve usar o parâmetro de vizinhança k para limitar a cardinalidade do conjunto resposta, embutindo o critério de ordenação na consulta;

- Critério flexível de diversidade: O algoritmo de busca deve permitir a avaliação flexível de diversidade de acordo com o objeto de consulta e os elementos recuperados, ponderando automaticamente a diversidade retornada em função da cardinalidade do conjunto resposta;

- Parâmetros do usuário: Idealmente o ajuste flexível de diversidade não deve ser realizado por um parâmetro do usuário, mas sim por um critério do próprio algoritmo de busca, evitando etapas de pré-processamento e ajustes finos, tal como ocorre para os operadores de busca por identidade e ordem; e

- Otimização da consulta com índices métricos: Idealmente, o algoritmo de busca deve poder se beneficiar de estruturas indexação métricas para ter escalabilidade e lidar com grandes conjuntos de dados. Um exemplo de algoritmo que utiliza essa estratégia é o r-DisC [31] que indexa conjuntos resposta para diversos valores de ξ por meio de uma *Cover-Tree* [11], que, por sua vez, é utilizada para otimizar a execução de outras buscas. Métodos baseados em novidade também podem se beneficiar de índices como o *RC-Index* [94] onde os dados são indexados primeiro por similaridade (δ) e, em outra estrutura, indexados por diversidade (δ_{div}).

A Tabela 3.1 apresenta uma comparação geral dos métodos revisados nessa seção com relação aos critérios levantados e indica que os algoritmos existentes falham em implementar, pelo menos, uma das características destacadas.

Por exemplo, o método GNE não fornece respostas determinísticas e requer parâmetros externos do usuário, enquanto o método r-DisC fornece apoio à execução otimizada com o uso de um índice métrico, mas não permite a definição de um critério de busca onde são controlados a cardinalidade e a localidade do conjunto resposta.

A direção dessa dissertação é no sentido que um algoritmo escalável nos moldes das soluções existentes para identidade e ordem para o problema de consultas por similaridade diversificada deve satisfazer todos esses critérios simultaneamente. O algoritmo *diversity browsing*) proposto na sequência (Seção 4) segue estas orientações e está catalogado como a última entrada dos algoritmos listados na Tabela 3.1.

3.4 Conclusões Parciais

Esse capítulo apresentou conceitos complementares às consultas por similaridade para inclusão de diversidade em conjuntos resposta. Foram apresentadas as principais soluções da literatura, exemplificadas com algoritmos e estudos de caso. Em resumo, os conceitos discutidos nessa seção, necessários para a elaboração de um novo algoritmo de busca, são:

- Consultas por vizinhança diversificada podem ser implementadas por meio de diversos algoritmos de diversificação de resultados, porém nem todos esses métodos estendem a modelagem por Espaços Métricos revisadas na seção anterior;
- Características importantes para a diversificação de resultados incluem a *(i)* localidade do objeto de consulta, *(ii)* determinismo, *(iii)* controle da cardinalidade do conjunto resposta, e *(iv)* controle flexível entre similaridade e diversidade. O único método revisado da literatura que atende todos esses requisitos é o BRID$_k$;
- Uma boa solução para o problema de diversificação de resultados inclui um método que seja capaz de se beneficiar de técnicas de indexação sem requerer parâmetros externos ou de ajuste fino dos usuários; e
- No melhor do nosso conhecimento, nenhum dos métodos revisados foi avaliado experimental ou teoricamente nas condições limite de buscas sobre conjuntos de alta dimensionalidade que podem comprometer as separações baseadas em distância tanto para similaridade quanto para diversidade.

Tabela 3.1: Comparação qualitativa de características dos algoritmos revisados para diversificação de resultados.

Método	**Localidade objeto o_q**	**Determinismo da resposta**	**Cardinalidade Conj. Resp.**	**Dinamicidade do raio**	**Ausência parâmetros**	**Apoio de Índices**	**Referência**
MaxMin	⊗	⊗	⊗	⊗	✓	⊗	[70]
MMR	✓	✓	✓	✓	⊗	⊗	[15]
GMC	✓	✓	✓	✓	⊗	⊗	[91]
GNE	✓	⊗	✓	✓	⊗	⊗	[91]
Swap	✓	⊗	✓	✓	⊗	⊗	[99]
BSwap	✓	⊗	✓	✓	⊗	⊗	[99]
Motley	✓	✓	✓	⊗	⊗	⊗	[47]
r-DisC	⊗	✓	⊗	⊗	⊗	✓	[33]
$BRID_k$	✓	✓	✓	✓	✓	⊗	[79]
diversity browsing	✓	✓	✓	✓	✓	✓	[49]

Capítulo 4

Diversity browsing

4.1 Premissas do algoritmo

A taxonomia revisada no Capítulo 3 para os algoritmos de diversificação de resultados encontrados na literatura indica que alguns dos métodos baseados em novidade realizam um pós-processamento sobre um conjunto de busca inicial, enquanto que os métodos baseados em cobertura realizam a separação entre similaridade e diversidade durante um único processamento de busca. A vantagem decorrente dessa premissa é que métodos por cobertura são, potencialmente, menos custosos, indexáveis e permitem estender os critérios de comparação por similaridade de forma mais transparente.

De acordo com a primeira parte da Hipótese de Pesquisa, levantada no Capítulo 1, métodos por cobertura podem se beneficiar de estruturas de indexação métricas para descartar regiões de busca que não satisfaçam simultaneamente os critérios de *(i)* similaridade para o objeto consultado e *(ii)* dissimilaridade para os elementos no conjunto resposta. Por outro lado, destacam-se também as observações encontradas após a revisão da literatura considerando os algoritmos e índices para consultas por similaridade que precisam ser levados em consideração na investigação da Hipótese de Pesquisa. Em particular, essa investigação requer considerar as observações: *(i)* o algoritmo *distance-browsing* é ótimo no número de cálculo de distâncias e gera um resultado incremental; *(ii)* algoritmos de busca por vizinhança precisam lidar com o problema de ordenação; e *(iii)* índices VP-Trees se mostraram os mais eficientes e baratos para a execução de consultas por vizinhança em avaliações experimentais na oferta de grande quantidade de memória.

Esse capítulo apresenta o algoritmo *diversity browsing* como uma solução que permite a realização incremental de consultas por similaridade diversificada (com critérios análogos

ao método de cobertura BRID$_k$) e se beneficia diretamente de índices métricos. Ao final, o método *diversity browsing* é avaliado empiricamente com o uso de índices VP-Trees contra o algoritmo BRID$_k$ e métodos por novidade com resultados que indicam ganhos substanciais de desempenho, corroborando a primeira parte da hipótese pesquisada nesse trabalho. A ideia é estender o algoritmo *distance browsing* para avaliar a Influência gerada por um elemento no conjunto resposta, não apenas sobre objetos individuais no conjunto de busca, mas sim com relação a cada uma das partições indexadas por um índice métrico. Dessa forma, partições inteiras são descartadas tanto pelo Princípio do Limite Inferior quanto por estarem sob Influência de um elemento no conjunto resposta.

4.2 Fundamentação do algoritmo

O algoritmo proposto, *diversity browsing*, preserva todas as propriedades de ordenação e das otimizações trazidas pelo uso dos Princípios os Limites Inferior e Superior em Espaços Métricos. Nesse sentido, o *diversity browsing* estende a rotina *distance browsing* (detalhada na Seção 2.1.3) para apoiar a execução de consultas por vizinhança diversificada, seguindo os mesmos critérios das Definições 4 e 5, que são os mesmos adotados pela abordagem competidora baseada em cobertura BRID$_k$. De acordo com essas definições, os elementos no conjunto resposta são os elementos influenciadores a partir da perspectiva do objeto de consulta e são usados para descartar elementos influenciados (que estão dentro do Conjunto de Influência Forte formado a partir do par ⟨objeto de consulta, influenciador⟩). Dessa forma, um elemento (e, analogamente, uma partição) é descartado do conjunto resposta por *(i)* não ser um dos k-vizinhos mais próximos do objeto de consulta, ou *(ii)* estar sob Influência de um elemento do conjunto resposta.

A extensão da abordagem *distance browsing* fornece uma garantia para a resolução incremental ótima do problema da ordenação tão necessária às consultas por vizinhança. Isso acontece porque dado um conjunto de dados $\mathcal{O} \subseteq \mathbb{O}$ indexado por um índice métrico e um elemento de consulta $o_q \in \mathbb{O}$, o algoritmo *distance browsing* retorna o i-ésimo vizinho mais próximo de forma incremental, sempre respeitando a ordenação $\delta(o_q, o_i) \leq \delta(o_q, o_{i+1})$. Essa propriedade permite reduzir a cobertura de um Conjunto de Influência Forte à uma região delimitada por um elemento central e um raio fixo, *i.e.*, uma bola fechada, no Espaço Métrico. O Lemma 2 apresenta a redução de um Conjunto de Influência Forte $\ddot{I}_{o_i,o_q}$ para uma bola fechada $\langle o_i, \delta(o_i, o_q)\rangle$ considerando o conjunto dos elementos $o_j \in \mathcal{O} \setminus \{o_i\}$ que devem ser recuperados de forma incremental após o_i.

Lemma 2 (Redução do Conjunto de Influência Forte.) *Seja o elemento $o_i \in \mathcal{O}$ o último influenciador no conjunto resposta $\mathcal{R}$, o problema de verificar se o próximo vizinho obtido pelo algoritmo distance browsing $o_{i+1} \in \mathcal{O} \setminus \cup_{j \in \mathcal{R}} \ddot{I}_{o_j,o_q}$ está dentro do conjunto de Influência $\ddot{I}_{o_i,o_q}$ é reduzido a verificar se o_{i+1} está coberto pela bola fechada $\langle o_i, \delta(o_i,o_q)\rangle$ em $\mathcal{O} \setminus \cup_{j \in \mathcal{R}} \ddot{I}_{o_j,o_q}$, i.e., $\ddot{I}_{o_i,o_q} \equiv \langle o_i, \delta(o_i,o_q)\rangle$ com relação a $\mathcal{O} \setminus \cup_{j \in \mathcal{R}} \ddot{I}_{o_j,o_q}$.*

Prova. A rotina *distance browsing* garante que $\delta(o_i,o_q) \leq \delta(o_{i+1},o_q)$ para qualquer par de vizinhos consecutivos $o_i, o_{i+1} \in \mathcal{O}$ [42], de forma que $I(o_i,o_q) \geq I(o_{i+1},o_q)$ é válido e as desigualdades $I(o_i,o_{i+1}) \geq I(o_i,o_q) \geq I(o_{i+1},o_q)$ são verdadeiras sempre que ocorra $\delta(o_i,o_{i+1}) \leq \delta(o_i,o_q)$. ■

A Figura 4.1 mostra como a exclusão de $\cup_{j \in \mathcal{R}} \ddot{I}_{o_j,o_q}$ gera uma região vazia em $\mathcal{O}$, permitindo essa redução. Note que, caso os elementos do conjunto de dados não fossem examinados de forma incremental, não seria possível garantir a redução, uma vez que não é possível caracterizar de antemão a relação entre $I(o_i,o_q)$ e $I(o_{i+1},o_q)$ para qualquer par o_i, o_{i+1} não ordenado por distância.

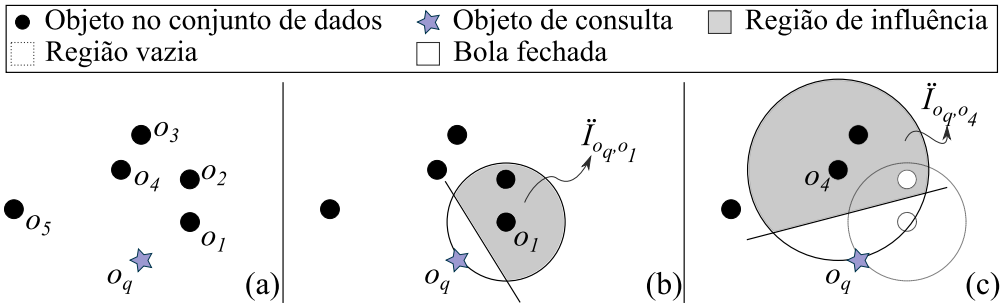


Figura 4.1: Conjunto de Influência Forte *vs.* partições definidas como bolas fechadas considerando a função de distância L_2. (a–b) Conjunto consultado $\mathcal{O}$ e objeto de consulta o_q. (c) Conjunto consultado $\mathcal{O} \setminus \ddot{I}_{o_1,o_q}$.

A partir da redução dada pelo Lemma 2 torna-se possível aplicar os Princípios dos Limites Inferior e Superior para gerir (descartar e incluir) elementos e partições do índice métrico que estejam sobrepostas à consulta. A primeira regra que pode-se inferir a partir do Limite Superior é a da admissão do vizinho mais próximo também como elemento influenciador (e, consequentemente, como o próximo *vizinho diversificado*) na resposta sem que haja a necessidade de se calcular sua Influência para qualquer outro objeto no conjunto resposta. O Lemma 3 formaliza essa regra de admissão.

Lemma 3 (Admissão direta de vizinho diversificado.) *Dados os elementos $o_1, o_2, \ldots, o_n \in \mathcal{O}$ ordenados por distância a o_q e seja o influenciador $o_i \in \mathcal{O}$ o vizinho mais*

distante de o_q em $\mathcal{R}$, então $o_{i+1} \in \mathcal{O}$ pode ser seguramente assumido como o seguinte influenciador mais próximo sempre que $\delta(o_{i+1}, o_q) > 2 \cdot \delta(o_i, o_q)$.

Prova. Como $\delta(o_{i+1}, o_q) > 2 \cdot \delta(o_i, o_q) > \delta(o_i, o_q)$, então o_{i+1} é um influenciador de o_i, caso contrário, $\delta(o_i, o_q) \geq \delta(o_{i+1}, o_i)$, *i.e.*, $o_{i+1} \in \ddot{I}_{o_i,o_q}$. Pela desigualdade triangular, segue que $2 \cdot \delta(o_i, o_q) \geq \delta(o_{i+1}, o_i) + \delta(o_i, o_q) \geq \delta(o_{i+1}, o_q)$, o que é um absurdo. Por fim, tem-se que $o_{i+1} \notin \ddot{I}_{o_j,o_q}, o_j \in \mathcal{R} \setminus \{o_i\}$, pois $\delta(o_i, o_q) \geq \delta(o_j, o_q)$. ■

Analogamente, tem-se a aplicação do Princípio do Limite Superior para descartar partições do espaço de busca (na forma de bolas fechadas) que estão totalmente sobrepostas a um Conjunto de Influencia Forte, sem que haja a necessidade de se calcular nem a distância nem a Influência dos elementos dentro dessa partição. O Lemma 4 apresenta essa regra para descartar regiões no espaço de busca.

Lemma 4 (Descarte de partições influenciadas.) *Uma partição $\langle p, \xi_p \rangle$ indexada como uma bola fechada no espaço de busca para um dado elemento pivô p e raio de cobertura ξ_p está dentro de um Conjunto de Influência Forte $\ddot{I}_{o_i,o_q}$ definida pelo objeto no conjunto resposta $o_i \in \mathcal{R}$ se $\delta(o_i, o_q) \geq \delta(p, o_i) + \xi_p$.*

Prova. Dada uma distância máxima M_p entre p e qualquer objeto em $\mathcal{O}$, se a bola fechada $\langle o_i, \delta(o_i, o_q) \rangle$ in $\mathcal{O} \setminus \cup_{j \in \mathcal{R}} \ddot{I}_{o_j,o_q}$ cobre $\langle p, \xi_p \rangle$, então $\delta(o_i, o_q) \geq \delta(o_i, p)$; $\delta(o_i, o_q) \geq \delta(o_i, o_j)$; e $\xi_p \geq \delta(p, o_j), \forall\ o_j \in \langle p, M_p \rangle$. Pela desigualdade triangular, tem-se que $\delta(o_i, o_q) \geq \delta(o_i, p) + \xi_p \geq \delta(o_i, p) + \delta(p, o_j) \geq \delta(o_i, o_j)$. ■

Por último, mas não menos importante, a aplicação do Princípio do Limite Inferior permite inferir a regra de parada da busca por vizinhança diversificada sem que haja, necessariamente, que se comparar por distância o objeto de consulta com cada elemento no espaço de busca. O Lemma 5 permite descartar todos as partições (e elementos) que não satisfaçam o Princípio do Limite Inferior para o último influenciador.

Lemma 5 (Apenas vizinhos próximos e diversos são considerados na busca.) *Se o conjunto resposta parcial e ordenado $\mathcal{R}$ contém $k-1$ elementos e a distância do candidato diverso $o_{k'} \in \mathcal{O} \setminus \cup_{o_j \in \mathcal{R}} \ddot{I}_{o_j,o_q}$ para o_q é conhecida, então a partição $\langle p, \xi_p \rangle$ pode ser podada da consulta diversificada se $\delta_{min}(\langle p, \xi_p \rangle, o_q) > \delta(o_{k'}, o_q)$.*

Prova. Dado que $o_{k'} \in \mathcal{O} \setminus \cup_{o_j \in \mathcal{R}} \ddot{I}_{o_j,o_q}$, então $\delta(o_{k'}, o_q) \geq \delta(o_i, o_q), \forall\ o_i \in \mathcal{R}$. Além disso, se $\delta_{min}(\langle p, \xi_p \rangle, o_q) > \delta(o_{k'}, o_q)$, então $\delta(o_i, o_q) > \delta(o_{k'}, o_q), \forall\ o_i \in \langle p, \xi_p \rangle$. Ou

seja, $\delta(o_h, o_q) > \delta(o_{k'}, o_q) \geq \delta(o_i, o_q), \forall\ o_i \in \mathcal{R}$ e a lista ordenada de influenciadores é $o_1, \ldots, o_{k-1}, o_{k'}, o_h$, onde cada elemento $o_h \in \langle p, \xi_p \rangle$ vem depois do k-ésimo candidato. ■

Repare que o Lemma 5 depende da função δ_{min} a ser definida de acordo com as particularidades de cada estrutura de indexação métrica. Para a implementação do algoritmo *diversity browsing* foi escolhido o índice VP-Tree, que é adequado para arquiteturas com alta disponibilidade de memória principal. É salutar destacar, no entanto, que a proposta não depende de nenhuma estrutura de indexação em particular, sendo necessário apenas definir as funções δ_{min} e δ_{max} (assim como no cenário do uso do algoritmo *distance browsing* para consultas apenas por similaridade).

4.3 Implementação do algoritmo

O Algoritmo 11 apresenta a implementação do *diversity browsing* sobre uma VP-Tree como uma rotina que inspeciona o conjunto de dados $\mathcal{O}$ com duas filas de prioridade que são utilizadas para ordenar:

1. os elementos candidatos o_i de acordo com suas distâncias para o_q e
2. as partições da VP-Tree de acordo com δ_{min}, com desempates por δ_{max}.

As partições são percorridas em ordem de proximidade para o elemento de consulta e os elementos do conjunto de dados são inseridos na fila de candidatos apenas quando o nó-folha a qual pertencem é acessado. Inicialmente, tanto a fila de candidatos quanto a de partições estão vazias, e o *diversity browsing* insere a partição correspondente à raiz da VP-Tree para avaliação. Na sequência, o algoritmo entra em um processo de *loop* onde a cada iteração examina o candidato ou a partição que esteja mais próximo a o_q.

Se um elemento candidato é escolhido para avaliação e é marcado como não influenciado pelos Lemmas 3–5, ele é então incluído no conjunto resposta. Elementos influenciados são descartados. Por outro lado, se uma partição é escolhida para avaliação, o algoritmo irá *(i)* dividi-la em dois pedaços disjuntos, caso se trate de uma partição representada por um nó-diretório, ou *(ii)* carregar seus elementos do disco e ordená-los na fila de prioridade de candidatos, caso se trate de uma partição representada por um nó-folha.

A Figura 4.2 exemplifica uma execução do *diversity browsing* considerando a mesma estrutura de VP-Tree construída na Seção 2.1.3 sobre um conjunto de dados de duas

Dados: Raiz VP-Tree nóRaiz, elemento de consulta o_q, quantidade de vizinhos diversificados k.
Retorno: Conjunto resposta diversificado $\mathcal{R}$.

```
1  nóRaiz.δ_min ← 0.0;
2  nóRaiz.δ_max ← ∞;
3  R ← Ø;
4  filaPrNós ← {nóRaiz};/* Fila de nós ordenados por δ_min e δ_max          */
5  filaPrElem ← Ø;/* Fila de para candidatos ordenados por δ                */
6  while (|filaPrNós| > 0 ∨ |filaPrElem| > 0) do
7      if |filaPrElem| = 0 then
8          nó ← removeTopo(filaPrNós);
9          if éFolha(nó) then
10             filaPrElem ← {nó.elementos}
11         else
12             calcularδ_min δ_max(nó.O_esq);
13             calcularδ_min δ_max(nó.O_dir);
14             filaPrNós ← filaPrNós ∪ {nó.O_esq} ∪ {nó.O_dir};
15     else if ((|filaPrNós| > 0) ∧ δ_min(topo(filaPrNós)) < δ(o_q, topo(filaPrElem)))
        then
16         nó ← removeTopo(filaPrNós);
           /* Poda pelos Lemmas 4-5                                          */
17         if nó ⊈ ∪_{o_j∈R} Ï_{o_j,o_q} ∧ |R| < k then
18             if éFolha(nó) then
19                 filaPrElem ← filaPrElem ∪ {nó.elementos};
20             else
21                 calcularδ_min δ_max(nó.O_esq);
22                 calcularδ_min δ_max(nó.O_dir);
23                 filaPrNós ← filaPrNós ∪ {nó.O_esq} ∪ {nó.O_dir};
24     else
25         o_i ← removeTopo(filaPrElem);
           /* Poda pelo Lemma 3                                              */
26         if o_i ∉ ∪_{o_j∈R} Ï_{o_j,o_q} ∧ |R| < k then
27             R ← R ∪ {o_i}
28 return R;
```

Algoritmo 11: Algoritmo incremental de busca por vizinhança diversificada *diversity browsing* para uma estrutura de índice VP-Tree acoplada.

dimensões com vinte objetos para realizar uma consulta por vizinhança diversificada com distância L_2 para uma vizinhança $k = 3$. A Figura 4.2(a-c) mostra, respectivamente, o conjunto bidimensional $\mathcal{O}$, as partições geradas pelo particionamento da VP-Tree e como os elementos de $\mathcal{O}$ se organizam no índice. A Figura 4.2(d-e) ilustra os elementos recuperados e seus Conjuntos de Influência Forte. De forma complementar, a Figura 4.3 mostra o passo-a-passo e o estado da filas de prioridade para a execução do Algoritmo 11 durante a realização da busca da Figura 4.2.

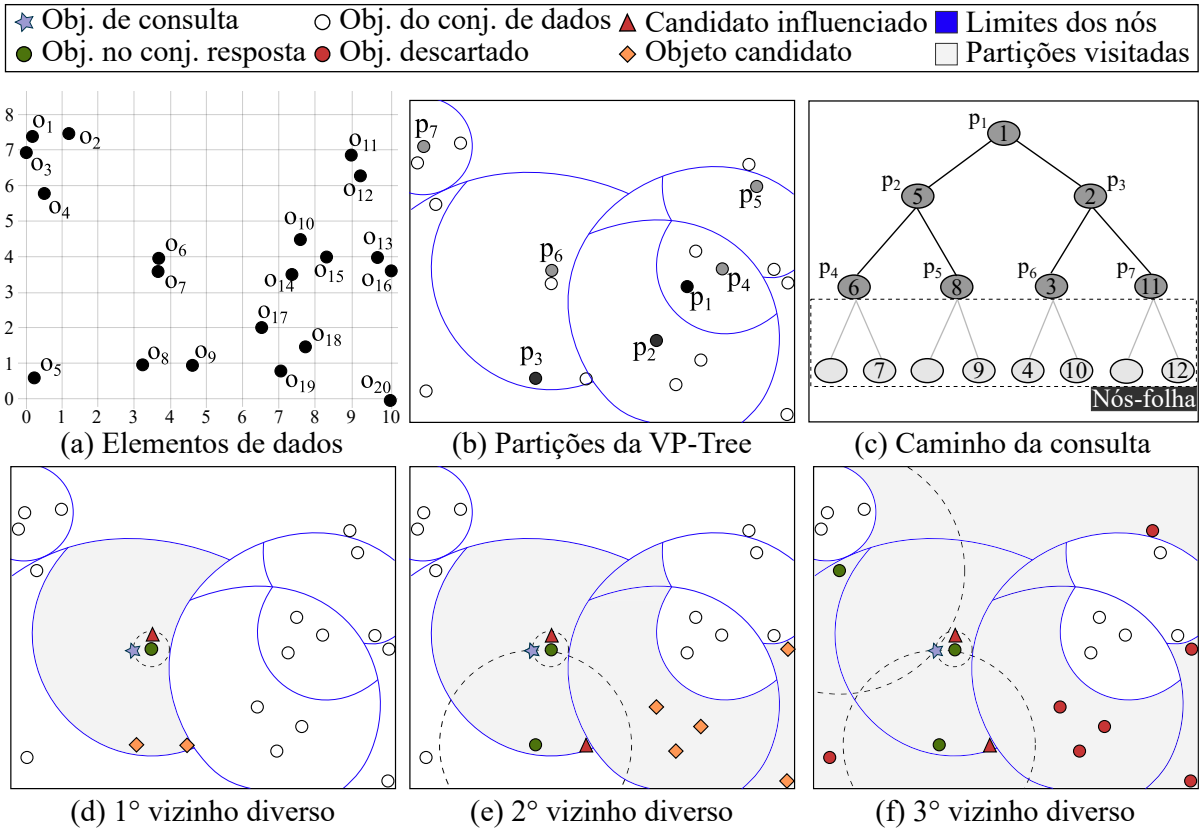


Figura 4.2: Exemplo de execução de uma consulta por vizinhança pelo *diversity browsing* considerando um conjunto de dados de duas dimensões indexado por uma VP^{sb}-Tree com capacidade para dois elementos por nó-folha e distância L_2. Nesse exemplo, os elementos pivôs são armazenados apenas nos nós-folha.

Inicialmente, a fila de partições está vazia e algoritmo *diversity browsing* insere o nó raiz indexado pelo pivô p_1 – Figura 4.3 Passo 1. Na sequência, o método inicia um *loop* onde remove o topo da fila de partições e o divide em suas duas partições $\mathcal{O}_{dir_{p_1}}$ (indexado por p_3) e $\mathcal{O}_{esq_{p_1}}$ (indexado por p_2). Na sequência, a partição $\mathcal{O}_{dir_{p_1}}$ é retirada do topo da fila de partições candidatas e dividida em $\mathcal{O}_{dir_{p_3}}$ (indexado por p_7) e $\mathcal{O}_{esq_{p_3}}$ (indexado por p_6) – Figura 4.3 Passo 2–3. Nesse ponto, a partição p_6 é retirada da fila de partições candidatas e dividida nas partições $\mathcal{O}_{dir_{p_6}}$ (para encurtar, r_{p_6}) e $\mathcal{O}_{esq_{p_6}}$ (para encurtar, l_{p_6}), que são nós-folha. Como l_{p_6} é a partição mais próxima ao objeto de consulta o_q e a fila de elementos candidatos está vazia, l_{p_6} é removida da lista de nós e seus elementos $l_{p_6} = \{o_6, o_7, o_8, o_9\}$ são carregados na lista de elementos de acordo com suas distâncias para o_q. Nesse ponto, o algoritmo precisa escolher entre examinar o candidato o_7 e a partição indexada por p_2 e decide por examinar o_7 dado que $\delta(o_q, o_7) < \delta_{min}(\mathcal{O}_{esq_{p_1}})$. O elemento o_7 é então recuperado como o primeiro vizinho diversificado incremental, pois o conjunto resposta parcial $\mathcal{R}$ está vazio e, consequentemente, não pode influenciar nenhum elemento candidato – Figura 4.3 Passo 5.

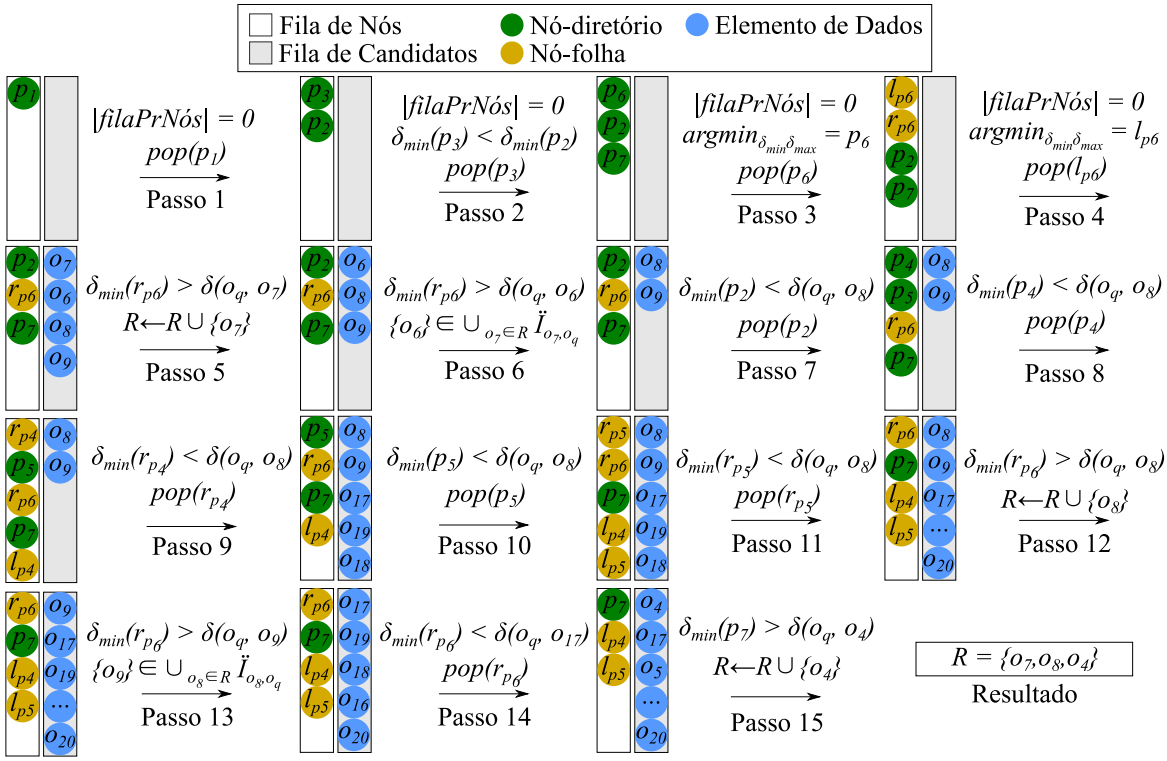


Figura 4.3: Teste de Mesa para o exemplo na Figura 4.2 considerando as duas filas de prioridade do algoritmo *diversity browsing* e o conjunto de resposta incremental $\mathcal{R}$.

Na próxima iteração do *loop* principal, o *diversity browsing* decide examinar o candidato o_6, pois $\delta(o_q, o_6) < \delta_{min}(\mathcal{O}_{esq_{p_1}})$. No entanto, percebe-se que o_6 é um elemento influenciado por $o_7 \in \mathcal{R}$ dado que $o_6 \in \ddot{I}_{o_7,o_q}$ e, consequentemente, o algoritmo descarta o_6 como vizinho diversificado removendo-o da lista de candidatos. Na sequência, o *diversity browsing* decide por examinar a partição indexada por p_2 em vez do próximo candidato o_8 visto que $\delta_{min}(\mathcal{O}_{esq_{p_1}}) \leq \delta(o_q, o_8)$, o que leva às divisões da partição $\mathcal{O}_{esq_{p_1}}$ nas partições $\mathcal{O}_{dir_{p_2}}$ (indexada por p_5) e $\mathcal{O}_{esq_{p_2}}$ (indexada por p_4) e, na sequência, na divisão de $\mathcal{O}_{esq_{p_2}}$ em $\mathcal{O}_{dir_{p_4}}$ (para encurtar, r_{p_4}) e $\mathcal{O}_{esq_{p_4}}$ (para encurtar, l_{p_4}).

Nesse ponto, o nó-folha r_{p_4} é a partição mais próxima do objeto de consulta o_q e, consequentemente, seus elementos $r_{p_4} = \{o_{17}, o_{19}, o_{19}\}$ são carregados na lista de candidatos de acordo com suas distâncias para o_q. Na próxima iteração, o algoritmo decide retirar a partição indexada por p_5 dado que $\delta_{min}(\mathcal{O}_{dir_{p_2}}) \leq \delta(o_q, o_8)$ e dividi-la em $\mathcal{O}_{dir_{p_5}}$ (para encurtar, r_{p_5}) e $\mathcal{O}_{esq_{p_1}}$ (para encurtar, l_{p_5}) que são nós-folha. Como r_{p_5} está mais próximo a o_q do que o candidato o_8, os elementos do nó-folha $r_{p_5} = \{o_{16}, o_{20}\}$ são carregados na lista de candidatos de acordo com sua distâncias ao objeto de consulta.

Na iteração seguinte, o candidato o_8 é removido da fila e avaliado, pois $\delta(o_q, o_8) < \delta_{min}(\mathcal{O}_{dir_{p_6}})$. O Lemma 3 permite admitir diretamente o_8 como o próximo vizinho não-

influenciado a ser incluído em $\mathcal{R}$ dado que $\delta(o_q, o_8) > 2 \cdot \delta(o_q, o_7)$. O candidato o_9, então, se torna mais próximo do objeto de consulta do que qualquer partição. No entanto, o_9 é influenciado por o_8 e o Algoritmo 11 o descarta como candidato à resposta.

Na sequência, o algoritmo examina a fila de partições, pois $\delta_{min}(\mathcal{O}_{dir_{p_6}}) \leq \delta(o_q, o_{17})$ e carrega os objetos ($\mathcal{O}_{dir_{p_6}} = \{o_4, o_5\}$) na fila de candidatos. O topo da primeira fila se torna o_4, enquanto a partição mais próxima ao objeto de consulta na segunda fila é indexada por p_7 com $\delta_{min}(\mathcal{O}_{dir_{p_3}}) \leq \delta(o_q, o_4)$. Logo, o *diversity browsing* divide $\mathcal{O}_{dir_{p_7}}$ em $\mathcal{O}_{dir_{p_7}}$ (para encurtar, r_{p_7}) e $\mathcal{O}_{esq_{p_7}}$ (para encurtar, l_{p_7}). O nó-folha $\mathcal{O}_{dir_{p_7}}$ é promovido ao topo da segunda fila, e como $\delta_{min}(\mathcal{O}_{dir_{p_7}}) \leq \delta(o_q, o_4)$, seu elemento $\mathcal{O}_{dir_{p_7}} = \{o_{11}\}$ é inserido na fila de candidatos de acordo com $\delta(o_q, o_{11})$.

Na próxima iteração, o Algoritmo 11 recupera o_4 como o próximo vizinho diversificado dado que $\delta(o_q, o_4) < \delta_{min}(\mathcal{O}_{esq_{p_7}})$ e que ele não é influenciado por nenhuma entrada em $\mathcal{R}$. Então, o algoritmo usa *(i)* o Lemma 4 para descartar a partição $\mathcal{O}_{esq_{p_7}}$ que está no topo da fila de partições, pois $\delta(o_4, o_q) \geq \delta(o_4, p_7) + \mu_{p_7}$ e *(ii)* o Lemma 5 para descartar as partições mais afastadas $\mathcal{O}_{esq_{p_4}}$ e $\mathcal{O}_{esq_{p_5}}$, pois $\delta_{min}(\mathcal{O}_{esq_{p_5}}) > \delta_{min}(\mathcal{O}_{esq_{p_4}}) > \delta(o_q, o_4)$. Como resultado, os elementos $\mathcal{R} = \{o_7, o_8, o_4\}$, nessa ordem, são retornados como os vizinhos mais próximos diversificados (ou influenciadores) para a consulta exemplo.

Ao final, foram visitadas 12 de 15 partições e foram feitas 12 comparações por distância e 3 por Influência, com um ganho de 40% tanto no número de cálculos de distância (12/20) quanto no número de cálculos de Influência (3/5) na comparação com o método BRID$_k$ que é o competidor base que também usa o conceito de diversidade por Influência.

4.4 Avaliação experimental

Para ilustrar numericamente o comportamento do algoritmo proposto, foi realizada uma comparação experimental do método *diversity browsing* contra o método competidor baseado em cobertura BRID$_k$ [76, 79], além de dois métodos baseados em novidade de grande representattividade da literatura: o método GMC (Algoritmo 5) e a abordagem GNE (Algoritmo 6), ambos detalhados na Seção 3.

A avaliação empírica foi realizada sobre um conjunto de dados sintético (denominado `SINT10`) e três conjuntos de dados reais (`MNIST`[1], `YAHOO`[2] e `COPHIR`[3]). A Tabela 4.1 des-

[1]Disponível em `yann.lecun.com/exdb/mnist/`
[2]Disponível em `webscope.sandbox.yahoo.com/catalog.php?datatype=i`
[3]Disponível em `cophir.isti.cnr.it/`

Tabela 4.1: Descrição dos conjuntos de dados usados na avaliação experimental.

Nome	$\lvert\mathcal{O}\rvert$	$\lvert\mathcal{O}'\rvert$	$\lvert\mathcal{O}''\rvert$	$\mathbb{R}^d$	$\boldsymbol{\delta}$	$\mathcal{D}$	**Descrição**
`SINT10`	$1M$	$70K$	$7K$	10	L_1	10	Dimensões *iid.* no intervalo [0,1].
`MNIST` [2]	$70K$	$70K$	$7K$	784	L_2	12	Dígitos escritos à mão.
`YAHOO` [3]	$2M$	$70K$	$7K$	400	L_2	08	Características de imagens do Yahoo®.
`COPHIR` [4]	$10M$	$70K$	$7K$	282	L_1	15	Características de cor de fotos.

creve esses conjuntos de dados com relação à sua cardinalidade $|\mathcal{O}|$, dimensionalidade $\mathbb{R}^d$, dimensionalidade intrínseca $\mathcal{D}$ e tamanho das amostras necessárias para a execução dos métodos baseados em novidade $|\mathcal{O}'|$ e $|\mathcal{O}''|$. A utilização de amostras é necessária devido a complexidade computacional (em função da cardinalidade) requerida para a avaliação das combinações dos elementos candidatos nos métodos GMC e GNE, sendo que sem o uso de amostras a execução dos algoritmos de otimização não terminaria dentro de um prazo razoável (semanas). Para reduzir o número de parâmetros, foi empregada a mesma função de distância δ para medir *similaridade* e *diversidade*, *i.e.*, $\delta = \delta_{sim} = \delta_{div_1} = \delta_{div_2}$. Todos os métodos foram implementados no mesmo *framework*, utilizando a linguagem Java com a JDK 13. Os experimentos foram executados no *cluster* ANOTi[4] de dois nós, cada um com 48 *cores* AMD Opteron 6320 com *Simultaneous Multithreading* (SMT), 96GB de memória compartilhada, um disco SATA de 01 TB e sistema operacional QLustar.

O objetivo da avaliação experimental é medir os ganhos de desempenho do *diversity browsing* em relação aos competidores considerando aspectos que *(i)* não envolvem características de implementação (medidas pela métrica de quantidade de cálculos de distância) e *(ii)* envolvem a resolução de consultas em ambientes do mundo real (medidas pela métrica de tempo de execução). Para a construção do *diversity browsing* foi considerado o índice VP-Tree em sua variante que permite armazenar mais de um elemento por nó-folha. Essa variante é paramétrica no balanceamento da árvore e na forma como são escolhidos os elementos pivôs. Para que a comparação contra os métodos da literatura fosse a mais justa possível, os resultados experimentais consideraram diversas dessas parametrizações da VP-Tree, ilustrando como o algoritmo proposto pode ser adequado com índice mais ou menos ajustados ao problema da busca.

Por último, mas não menos importante, os conjuntos de dados usados nos experimentos estão originalmente imersos em espaços de média e alta dimensionalidade, onde espera-se que índices métricos não sejam capazes de gerar partições de boa qualidade – Ver Seção 2.2. Para evitar esse possível problema, os conjuntos de dados foram reduzidos

[4]Cluster Laboratório ANOTi INFES/UFF: `anotilab.com`

para a sua dimensionalidade intrínseca por meio do algoritmo de Análise de Componentes Principais [1,62] em uma etapa pré-processamento. A dimensionalidade intrínseca foi estimada utilizando-se a média e desvio padrão da distribuição de distâncias do conjunto de dados, como descrito na Equação 2.3.

4.4.1 Ajuste de VP-Trees para o *diversity browsing*

Na primeira etapa dos experimentos, foram avaliados os desempenhos de diferente configuração de índices métricos VP-Tree com o objetivo de se identificar as configurações do índice mais e menos ajustadas à realização de consultas por similaridade diversificada. Em particular, foram examinadas as seguintes VP-Trees construídas a partir da combinação das diversas parametrizações do índice:

1. número de elementos por nó-folha igual a 100, o que gera árvores mais altas para conjuntos de dados maiores,
2. árvores balanceadas (`BAL`) ou não-balanceadas (`UBAL`), o que implica em dar suporte ao *overflow* de elementos em nós-folha de árvores forçadamente balanceadas, e
3. métodos de escolha de pivôs que podem variar entre os critérios de *aleatório com distribuição uniforme iid.* (`RND`), *fecho-convexo* (`CVX`, implementado como a proposta em [88]) ou *variância máxima* (`MAX`, implementado como a proposta em [97]).

As combinações desses parâmetros são consolidadas em seis tipos de estruturas, a saber: `VP_RND_BAL`, `VP_RND_UBAL`, `VP_CVX_BAL`, `VP_CVX_UBAL`, `VP_MAX_BAL` e `VP_MAX_UBAL`.

A Figura 4.4 mostra o desempenho médio do algoritmo *diversity browsing* na execução de consultas por vizinhança diversificada para os valores de $k = \{5, 10, 15, 20, 25\}$ sobre cada uma das parametrizações analisadas. As consultas foram executadas considerando a amostra do conjunto de dados $\mathcal{O}'$ de acordo com uma política de *holdout* onde 90% de elementos aleatórios da amostra foram indexados e os 10% restantes foram usados como objetos de consulta. Dentre os parâmetros analisados, o método de escolha de pivôs foi o que gerou maior impacto no desempenho do algoritmo *diversity browsing* na comparação com outras configurações.

Além disso, a construção `VP_MAX_BAL` obteve os melhores resultados na comparação contra a configuração `VP_RND_BAL` ao requerer menos cálculos de distância. A redução percentual de cálculos entre essas parametrizações foi de 18,36%, 12,83%, 14,76% e 9,26%

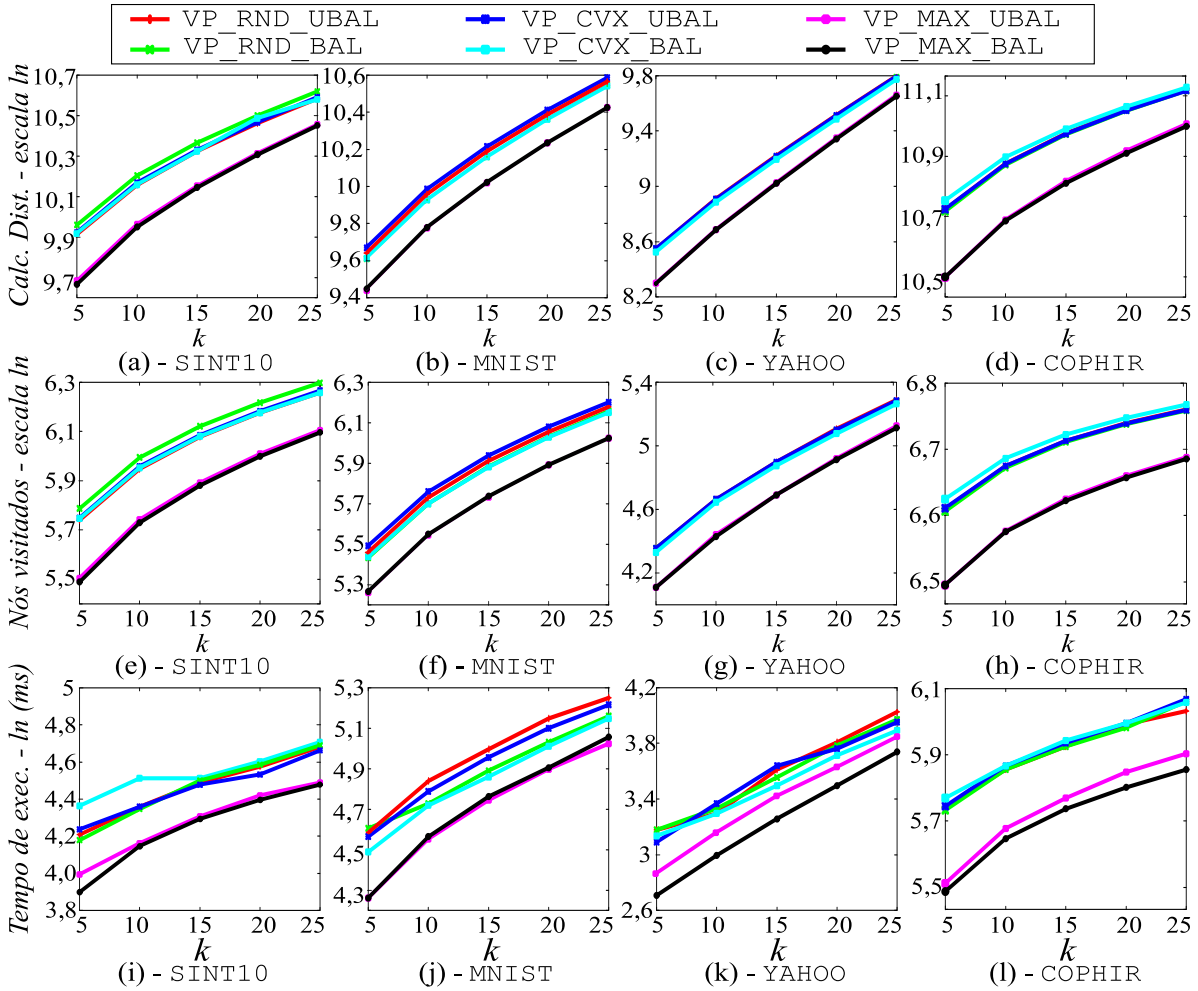


Figura 4.4: Valores médios em escala *log* natural para a execução do *diversity browsing* acoplado a diferentes configurações de índices métricos VP-Tree.

para os conjuntos `SINT10`, `MNIST`, `YAHOO` e `COPHIR`, respectivamente. O `VP_MAX_BAL` também superou o competidor `VP_CVX_BAL` nos mesmos conjuntos em, pelo menos, 15,76%, 10,12%, 12,92% e 12,21% – Figuras 4.4(a–d). Resultados similares foram observados com relação ao número de nós visitados (que emulam acessos a disco), onde o `VP_MAX_BAL` obteve melhor desempenho do que o `VP_RND_UBAL` por, pelo menos, 17,57%, 13,56%, 19,28% e 11,82% nos conjuntos de dados `SINT10`, `MNIST`, `YAHOO` e `COPHIR`, respectivamente – Figuras 4.4(a–d). Por último, mas não menos importante, nenhuma evidência de VP-Trees com fecho-convexo sendo sistematicamente superiores as VP-Trees com pivôs aleatórios foi observada, tanto com relação a cálculos de distância quanto no número de nós visitados.

O tempo de execução apresentou alguma variação com relação ao número de cálculos de distância, devido ao fato que o algoritmo de busca é composto não apenas de cálculos de distância mas também da manutenção das filas de prioridade, além dos cálculo dinâmicos que geram os Conjuntos de Influência Forte a partir dos elementos incrementalmente

recuperados. Ainda assim, os índices VP-Tree baseados em pivôs de variância máxima superaram por alguma margem todos os competidores em todos os conjuntos de dados analisados. Diferenças pouco significativas foram observada quanto ao balanceamento dos índices. Por exemplo, a construção `VP_MAX_UBAL` foi um pouco superior no conjunto `MNIST`, enquanto que a variante `VP_MAX_BAL` foi um pouco superior nos conjuntos `SINT10`, `YAHOO` e `COPHIR`. Portanto, para o critério que escolhe a configuração que obteve melhor desempenho na maioria dos conjuntos e consultas, tem-se que:

- A configuração mais ajustada é a construção `VP_MAX_BAL`, que será empregue na próxima comparação contra os métodos competidores; e
- a configuração menos ajustada é a construção `VP_RND_UBAL`, que será empregue para a comparação de pior desempenho do *diversity browsing*.

4.4.2 Diversity browsing *vs.* outros métodos de diversidade

Para comparar o algoritmo *diversity browsing* contra os métodos da literatura, foi necessário considerar um pequeno ajuste com relação às técnicas baseadas em novidade de-

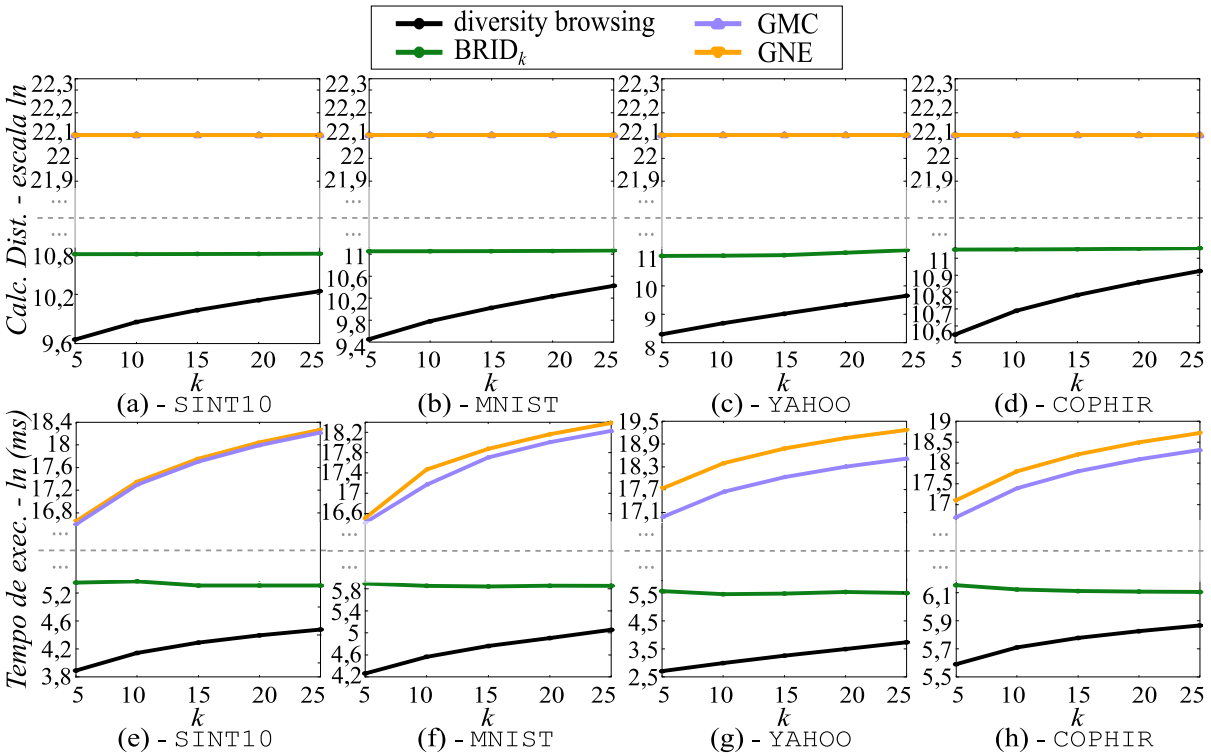


Figura 4.5: Valores médios em escala *log* natural para uma consulta por vizinhança diversificada executada pelos algoritmos *diversity browsing*, BRID$_k$, GMC e GNE considerando o parâmetro de diversidade $\lambda = 0{,}25$.

vido ao seu custo computacional. Enquanto os métodos *diversity browsing* e BRID$_k$ foram comparados usando o conjunto amostrado $\mathcal{O}'$, os métodos baseados em novidade GMC e GNE foram avaliados sobre amostras ainda menores de candidatos $\mathcal{O}''$, $|\mathcal{O}''| = 0.1 \times |\mathcal{O}'|$. Para essa segunda amostra, foi utilizada a mesma estratégia de avaliação *holdout* para a separação aleatória de 90% dos elementos do conjunto a serem indexados, sendo que os restante foram utilizados como elementos de consulta. Essa "vantagem" dada aos algoritmos baseados em novidade visa realçar ainda mais os ganhos dos métodos de cobertura com relação ao tempo de execução e quantidade de cálculos de distância.

A Figura 4.5 apresenta os resultados dos valores médios para comparação entre os métodos competidores, com os valores de cálculos de distância em tempo de execução em escala logarítmica (ln). O algoritmo *diversity browsing* teve um desempenho amplamente superior aos dos competidores para todos os conjuntos de dados e todos os valores de vizinhança. Em particular, o *diversity browsing* exigiu até 2,01, 0,59 e 3,79 vezes menos cálculos de distância que o competidor base BRID$_k$[5] com relação aos conjuntos de dados `SINT10`, `COPHIR` e `MNIST`, respectivamente. Resultados similares foram observados com relação ao tempo de execução, onde o *diversity browsing* foi, pelo menos, 87,51% mais rápido que o algoritmo base BRID$_k$.

Por último, mas não menos importante, a Figura 4.5 mostra os ganhos do algoritmo *diversity browsing* contra dois métodos baseados em novidade GMC e GNE. Nesses casos, o algoritmo proposto obteve resultados de, pelo menos, 5,67, 5,41, 6,07 e 4,91 ordens de magnitude de diferença com relação ao tempo de execução para o GMC para os conjuntos `SINT10`, `MNIST`, `YAHOO` e `COPHIR`, respectivamente. Analogamente, o *diversity browsing* obteve melhores resultados que o GNE também por 5,69, 5,42, 6,1 e 5 ordens de magnitude no tempo de execução. Em resumo, os resultados indicam:

- métodos por cobertura tem maior potencial de otimizar diversificação de resultados ao tratar o problema de diversidade *durante* a busca, e
- uma corroboração numérica para a primeira parte da Hipótese de Pesquisa desse trabalho que teorizava que estruturas de indexação métricas podem ser usadas com eficiência para melhorar o desempenho de algoritmos de busca por similaridade diversificada baseados em cobertura.

[5]Para uma comparação ainda mais justa, a implementação do BRID$_k$ em [79] foi modificada para contemplar o Lemma 3 (que fundamenta o *diversity browsing*) e que pode reduzir cálculos de Influência desnecessários também no BRID$_k$.

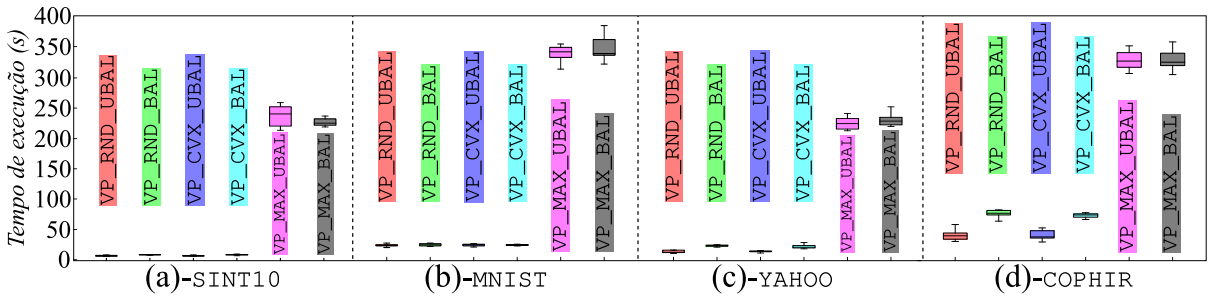


Figura 4.6: Tempos de construção para diferentes configurações de VP-Trees considerando 100 repetições para cada conjunto de dados.

4.4.3 Custos de ajuste fino dos índices métricos

O contra-ponto ao uso de estruturas de indexação é o custo de construção dos índices, que deve ser compensado com a realização das consultas para que a abordagem indexada seja válida, viável ou de interesse. Nesse experimento, são avaliados o tempo de construção das diferentes construções de VP-Trees e seu consequente impacto no algoritmo *diversity browsing*. A Figura 4.6 apresenta os tempos de construção para cada VP-Tree examinada nas avaliações anteriores, considerando 100 repetições de construção do índice para cada conjunto de dados. Os resultados são reportados na forma de *box-plots* com quatro quartis sem descartar nenhum valor extremo.

De forma geral, os resultados indicam que a construção do índice `VP_MAX_BAL` é a mais cara dentre as comparados, o que reflete o custo da escolha de seus pivôs que é superior aos demais métodos [48]. Por outro lado, a configuração `VP_RND_UBAL` apresentou o menor tempo de construção, espelhando um *trade-off* com relação à otimização do índice: quanto melhor o resultado em consultas por vizinhança diversificada, mais cara a construção da árvore. Mais uma vez, ressalta-se que os custos de construção de VP-Trees estão diretamente ligados aos métodos de escolha de pivôs, dado que a diferença entre índices balanceados e não balanceados foi pequena quando comparado com os pivôs.

Nos experimentos realizados na Seção 4.4.2, o tempo de construção das VP-Trees com pivôs selecionados pelo critério da variância máxima se paga quando o ganho acumulado com relação aos competidores ultrapassa o custo de construção do índice. Na avaliação anterior, o tempo acumulado gasto por consulta pelo *diversity browsing* somado ao tempo de construção do índice `VP_MAX_BAL` se torna menor que o tempo acumulado do BRID_k após $2.379, 1.104, 1.061$ e 2.351 consultas (correspondendo, na média geral, a 2,74% do número total de elementos indexados) para os conjuntos de dados `SINT10`, `MNIST`, `YAHOO` e `COPHIR`, respectivamente.

Em um outro comparativo, o desempenho do *diversity browsing* acoplado ao índice `VP_RND_UBAL` foi ainda, no *pior* dos casos examinados, 14% mais eficiente que o competidor mais próximo BRID_k considerando o tempo de execução de consulta, tendo o custo de construção do índice sido compensado após um número de consultas que corresponde a 6,12% do número total de elementos indexados, em uma média geral.

Em aplicações reais de recuperação por conteúdo e buscas por similaridade, essa razão tempo de consulta/construção do índice é geralmente muito maior, de forma que o uso de índices com ajuste fino, como a `VP_MAX_BAL`, acopladas ao *diversity browsing* é recomendado. Por outro lado, para ambientes de consultas *in-situ* ou em menor quantidade, estruturas de construção mais simples, como a `VP_RND_UBAL`, são as mais indicadas.

4.5 Conclusões parciais

Essa seção apresentou o algoritmo *diversity browsing*, que estende o estado-da-arte em diversificação de resultados seguindo os principais conceitos e recomendações levantados após uma cuidadosa revisão da literatura focada em consultas por similaridade e métodos baseados em cobertura (Capítulos 2 e 3). Uma avaliação experimental mostrou que o método proposto supera competidores por uma margem de desempenho significativa. De forma resumida, os principais destaques dessa seção são os seguintes:

- a proposição e a demonstração de quatro novos lemmas que permitem criar regras para descartar partições indexadas do espaço de busca;
- o projeto de um algoritmo incremental por similaridade diversificada que
  - reduz regiões de Influência a partições métricas e, consequentemente, permite utilizar os Princípios do Limite Inferior e Superior para descartar regiões indexadas que não satisfaçam simultaneamente o critério de proximidade e o critério de Influência para os elementos no conjunto resposta; e
  - estende o algoritmo de busca estado-da-arte de consultas por similaridade;
- uma avaliação numérica e experimental que corrobora a primeira parte da Hipótese de Pesquisa desse trabalho, ao mostrar que índices métricos podem ser acoplados a um algoritmo baseado em cobertura para aumentar expressivamente o desempenho de consultas por similaridade diversificada.

Capítulo 5

Diversity browsing e dimensionalidade intrínseca local

5.1 Concentração de distâncias e diversidade

Os trabalhos revisados na Seção 2.2 indicam que, na presença do fenômeno de concentração de distâncias, consultas por similaridade em espaços de alta dimensionalidade tendem a entregar resultados de baixo valor semântico e comprometer o uso de estratégias de busca baseados em índices métricos. Por outro lado, consultas por similaridade diversificada são, potencialmente, mais resistentes à concentração de distâncias pois podem usar dois critérios (em vez de um) para medir o grau de vizinhança: a proximidade e a Influência. Assim, ainda que se espere que algoritmos de busca por similaridade diversificada sejam mais dispendiosos do que algoritmos orientados somente à similaridade, *e.g.*, *diversity browsing vs. distance browsing*, a comparação desses algoritmos no caso limite de conjuntos de dados imersos em espaços de alta dimensionalidade é um ponto aberto na literatura e que precisa ser explorado para que se compreenda melhor os efeitos do fenômeno da concentração distâncias com relação a diversificação de resultados.

O conceito de dimensionalidade intrínseca local (LID – Seção 2.2.2) é empregado nessa seção para examinar experimentalmente o comportamento do *diversity browsing* em espaços de alta dimensionalidade. Mais especificamente, a LID é usada para categorizar a dificuldade de se consultar um particular objeto de referência, separando esses elementos em consultas "fáceis" e consultas "difíceis". Essa separação pode ser usada para definir níveis intermediários de dificuldade, o que permite analisar o comportamento do algoritmo de busca com mais detalhes, em vez de apenas um único valor médio para representar todo o conjunto consultado. No escopo da avaliação proposta, os objetos de

consulta são divididos em quatro níveis de dificuldade de acordo com seus valores de LID, gerando quatro conjuntos de consulta em vez de apenas um (Ver Seção 4.4).

A primeira avaliação realizada nessa seção é uma comparação direta entre as versões incrementais dos algoritmos de busca apenas por similaridade e por similaridade diversificada *distance browsing* e *diversity browsing*, respectivamente. O objetivo da comparação é mensurar a diferença do custo de execução entre essas duas consultas considerando buscas com baixo e alto LID. Uma discussão importante nessa comparação é quantificar o ônus real de se adicionar um critério de comparação extra (por Influência) em uma consulta por similaridade para contrapô-lo com o benefício de diversificação de resultados.

A segunda avaliação realizada nessa seção compara a "quantidade de diversidade" que pode ser encontrada para objetos de consulta mais difíceis (de maior LID) quando comparados a objetos de consulta mais fáceis (de menor LID). Essa medida é expressa como o número de vizinhos que podem ser encontrados para cada elemento de consulta e possui uma relação direta com a forma na qual os Conjuntos de Influência Forte são construídos para cada intervalo de LID. O objetivo dessa avaliação é combinar os resultados encontrados com os do primeiro experimento para tentar identificar se existe a vantagem de se realizar consultas por similaridade diversificada com resultados de maior potencial semântico a custos comparáveis aos de consultas apenas por similaridade.

Assim, o último ponto a ser avaliado experimentalmente é se o desempenho do algoritmo *diversity browsing* pode ser melhorado pelo ajuste fino dos índices métricos sob os quais a abordagem é acoplada para consultar elementos com diferentes valores de LID. Nessa comparação é empregado o índice VP-Tree e seus parâmetros de melhor e pior ajuste de acordo com os resultados apresentados na seção anterior, *i.e.*, pivôs escolhidos pelo critério de variância máxima *vs.* pivôs aleatórios. O objetivo dessa avaliação é identificar para quais intervalos de LID o ajuste dos índices tende a gerar ganhos substanciais e para quais intervalos de LID esse ajuste não é tão relevante. Dessa forma, sistemas de otimização de consultas capazes de realizar consultas sobre diversos métodos de acesso orientados à índices e construídos para diferentes partições de dados podem se beneficiar de processamento paralelo para reduzir o tempo de reposta das buscas.

Por último, essa seção apresenta uma estratégia de visualização de dados em espaços de alta dimensionalidade que leva em consideração tanto a perspectiva do objeto de consulta quanto partições definidas por intervalos de LID. Por essa perspectiva é possível explorar visualmente o conjunto de dados considerando a densidade da vizinhança dos elementos ao redor do objeto de consulta e, dessa forma, recupera parte da semântica

da proximidade tão necessária para consultas por vizinhança e que é comprometida pelo fenômeno de concentração de distâncias.

5.2 Materiais e métodos

Foram separados três conjuntos de dados do mundo real para a execução das avaliações experimentais nos casos limite de busca em espaços de alta dimensionalidade com o uso de funções de distância sujeitas ao fenômeno de concentração, a saber, `MNIST`[1], `COPHIR`[2] e `SIFT`[3]. A Tabela 5.1 apresenta os detalhes dos conjuntos examinados, incluindo sua cardinalidade $|\mathcal{O}|$, dimensionalidade $\mathbb{R}^d$, dimensionalidade intrínseca $\mathcal{D}$, variância relativa VR e função de distância associada δ.

Tabela 5.1: Descrição dos conjuntos de dados imersos em espaços de média e alta dimensionalidade usados na avaliação experimental.

Nome	$\lvert\mathcal{O}\rvert$	$\lvert\mathcal{O}'\rvert$	$\mathbb{R}^d$	$\mathcal{D}$	VR	$\boldsymbol{\delta}$	**Descrição**
`MNIST` [1]	$70K$	$70K$	784	12	0,14	L_2	Dígitos escritos à mão.
`COPHIR` [2]	$10M$	$70K$	282	15	0,14	L_1	Características de cor de fotos.
`SIFT` [3]	$01M$	$01M$	128	14	0,18	L_1	Valores SIFT extraídos de imagens.

A variância relativa foi calculada como na Equação 2.2, enquanto que a dimensionalidade intrínseca foi calculada de acordo com a estimativa ρ–score dada pela Equação 2.3. Note que, ao contrário da avaliação anterior, não foi necessário realizar dois níveis de amostragem, pois o algoritmo *diversity browsing* consegue lidar com conjuntos de maior cardinalidade em tempo polinomial. Tanto o *distance browsing* quanto o *diversity browsing* foram implementados em Java com a JDK 13. Os experimentos foram executados no *cluster* ANOTi[4] de dois nós, cada um com 48 *cores* AMD Opteron 6320 com *Simultaneous Multithreading*, 96GB de memória compartilhada, um disco SATA de 01 TB e sistema operacional QLustar.

Para a inferência dos intervalos da dimensionalidade intrínseca local (LID), os conjuntos de dados foram considerados tanto como conjuntos de busca quanto como conjuntos de objetos de consulta, ou seja, foi realizada uma busca por vizinhança para cada elemento $o_q \in \mathcal{O}$ considerando como conjunto de busca o próprio conjunto $\mathcal{O}$. A LID de cada objeto de consulta foi calculada de acordo com a Equação 2.4 para um valor de vizinhança

[1]Disponível em `yann.lecun.com/exdb/mnist/`
[2]Disponível em `cophir.isti.cnr.it/`
[3]Disponível em `corpus-texmex.irisa.fr/`
[4]Cluster Laboratório ANOTi INFES/UFF: `anotilab.com`

$k = 100$ de acordo com as recomendações da literatura [3,6]. Os valores calculados foram usados para dividir os elementos dos conjuntos de dados em *quartis* rotulados como:

1. Porção "fácil" (1QT), que compreende os elementos cuja LID se encontra no intervalo entre $[0, 25\%]$ dos valores de LID mais frequentes;
2. Porção "médio-fácil" (2QT), que inclui os elementos cuja LID se encontra no intervalo entre $(25\%, 50\%]$ dos valores de LID mais frequentes;
3. Porção "médio-difícil" (3QT), que compreende os elementos cuja LID se encontra no intervalo entre $(50\%, 75\%]$ dos valores de LID mais frequentes;
4. Porção "difícil" (4QT), que engloba os elementos cuja LID se encontra no intervalo entre $(75\%, 100\%]$ dos valores de LID mais frequentes.

5.3 Avaliação por diferentes faixas de LID

A Figura 5.1 mostra a distribuição das LIDs para os conjuntos de dados `MNIST`, `COPHIR` e `SIFT`, além dos valores limites (truncados) de sua divisão em quatro quartis. Essas distribuições são usadas para gerar quatro novos *folds* de avaliação que permitem analisar o desempenho da diversificação de resultados em função do grau de dificuldade da busca. O valor mediano da LID foi próximo ao valor da dimensionalidade intrínseca do conjunto

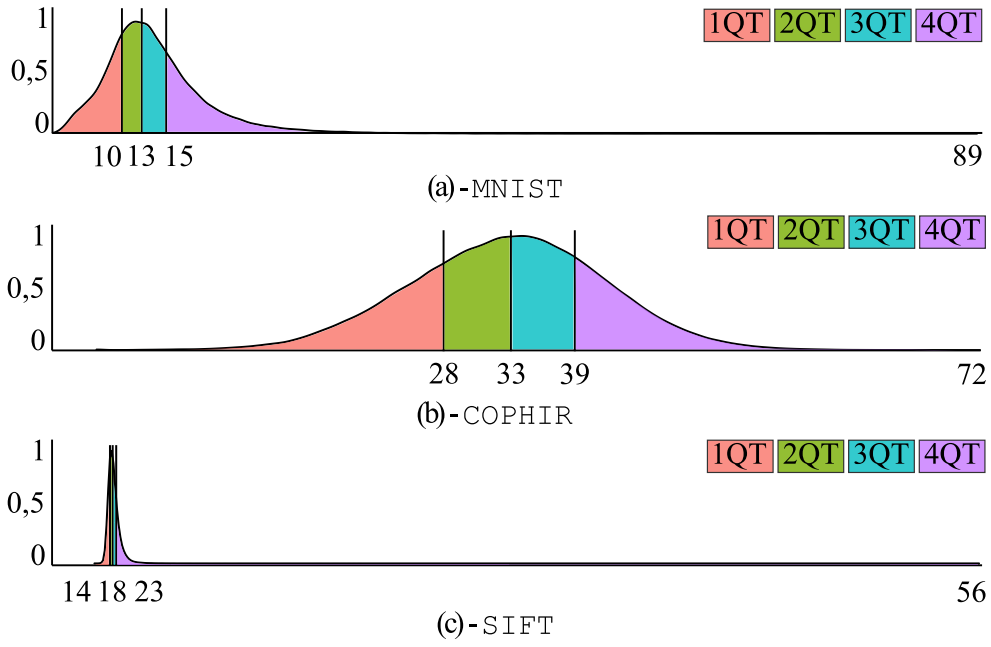


Figura 5.1: Distribuição de valores de dimensionalidade intrínseca local para os conjuntos de dados consultados, considerando cada elemento como um objeto de consulta.

em todos os casos ($\mathcal{D}$ – Tabela 5.1), destacando a qualidade da estimativa de aproximação da LID pela Equação 2.4.

Para os experimentos a seguir, quando aplicável, cada *fold* foi sub-dividido usando uma política de *holdout*, onde 90% dos elementos no *fold* foram indexados e os restantes 10% foram empregados como elementos de consulta. Por último, o número máximo de elementos em nós-folha de VP-Trees foi configurado como 100 em todos os cenários.

5.3.1 Diversity browsing *vs.* distance browsing

Como primeiro experimento, compara-se o desempenho do algoritmo *distance browsing* (estado-da-arte para resolver consultas por vizinhança) com o algoritmo *diversity browsing* (proposto nesse trabalho). Dado que o algoritmo de diversidade emprega comparações por proximidade e Influência, é esperado que o número de comparações (cálculos) por distância do *diversity browsing* seja maior do que os realizados pelo *distance browsing* para um mesmo valor de k. No entanto, quantificar essa diferença entre o número de cálculos depende também de características do conjunto de dados, sendo que um dos casos limite reportados na literatura é o de consultas realizadas em espaços de alta dimensionalidade *intrínseca*, que aqui são obtidos pelo *fold* 4QT que concentra os maiores valores de LID obtidos a partir dos objetos do conjunto de dados original.

A Figura 5.2 apresenta a diferença entre o valor médio do número de cálculos de distância exigidos por ambos *distance* e *diversity browsing* na execução de consultas por vizinhança sobre os valores extremos de LID representados pelos quartis 1QT e 4QT. Os

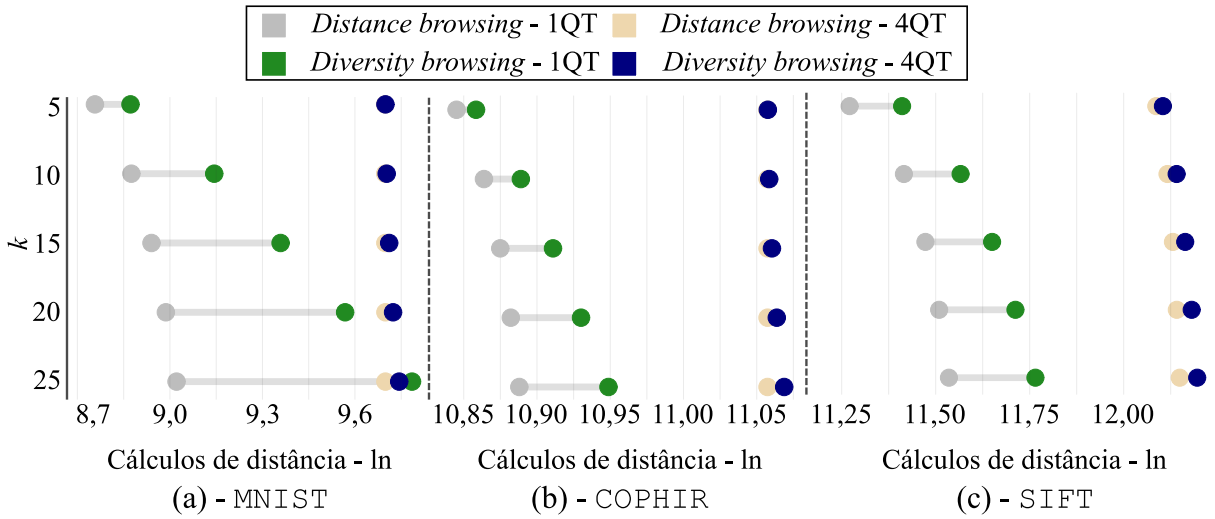


Figura 5.2: Diferenças entre as médias de desempenho dos algoritmos *distance browsing* e *diversity browsing* considerando o número de cálculos de distância.

resultados indicam que as menores diferenças são encontradas no 4QT e as maiores no 1QT, ou seja, quanto maior o valor da LID, mais os dois tipos de consulta tendem a ter o mesmo custo e quanto menor a LID, mais a consulta por similaridade tende a terminar primeiro do que a consulta por similaridade diversificada. Em particular, para o maior caso testado $k = 25$, a diferença entre os dois métodos foi de $68\%, 38\%$ e 59% menor no 4QT na comparação com o 1QT para os conjuntos `MNIST`, `COPHIR`, e `SIFT`, respectivamente. Observações importantes derivadas dessa análise experimental são:

- consultas por similaridade diversificada também são afetadas pelo problema de concentração de distâncias em seu número de cálculos de distância, pois consultas em espaços de alta dimensionalidade intrínseca (4QT) são expressivamente mais custosas do que aquelas em espaços de baixa dimensionalidade intrínseca (1QT); e
- consultas por similaridade e similaridade diversificada tendem a apresentar custos equivalentes para espaços de crescente dimensionalidade e, consequentemente, é conveniente escolher a que consiga melhor discernir objetos próximos e afastados.

A última observação indica que consultas por similaridade diversificada podem ser preferíveis às consultas por similaridade tradicionais para manter o viés de proximidade em espaços de alta dimensionalidade. Nesse sentido, faz-se necessário investigar como se comporta o algoritmo *diversity browsing* na consulta desses espaços, considerando valores crescentes de vizinhança e elementos descartados pelo critério de Influência.

5.3.2 A correlação entre diversidade e LID

Nesse experimento, foram realizadas consultas por vizinhança diversificada com o valor $k \to \infty$ com o objetivo de se traspassar todo o conjunto de dados com o algoritmo *diversity*

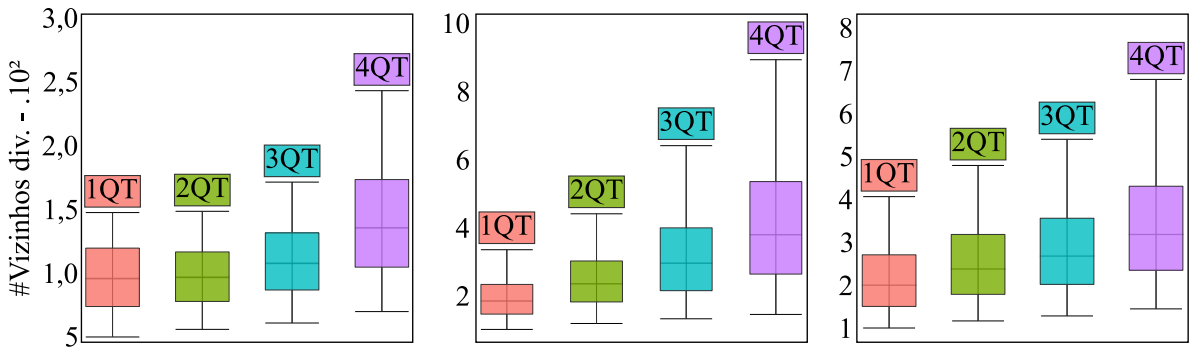


Figura 5.3: Número de vizinhos diversificados obtidos para consultas com $k \to \infty$.

browsing de modo que se pudesse analisar os descartes gerados pelos seus critérios de corte de proximidade e Influência. A Figura 5.3 mostra, na forma de *box-plots* com bigodes podados em 5%, a quantidade de vizinhos diversificados que foi recuperada para cada um dos objetos de consulta separados por seus *folds* de dificuldade.

Um primeiro resultado encontrado é que o número de vizinhos diversificados que podem ser encontrados é muito menor do que a cardinalidade do conjunto de dados consultado. Por exemplo, o *fold* 1QT do conjunto de dados `MNIST` contém $(70.000/4)\cdot 0,9 = 15.750$ elementos, sendo que uma consulta por vizinhança com $k \to \infty$ retornaria 15.750 objetos. No entanto, uma consulta por vizinhança diversificada retornou, no máximo, 1,500 elementos devido ao descarte de candidatos influenciados. Essa resposta ilustra a capacidade do *diversity browsing* para, dentro de um mesmo raio de distância, filtrar mais elementos do que o algoritmo *distance browsing* e, potencialmente, eliminar elementos candidatos muito similares entre si.

Um segundo resultado experimental é que a quantidade de vizinhos diversificados aumenta com a LID, *i.e.*, quanto maior a LID, mais vizinhos diversificados podem ser encontrados. Essa observação pode ser percebida pela variação dos limites dos *box-plots* na Figura 5.3 que aumentam na comparação com os crescentes intervalos de LID. Em um primeiro momento esse resultado pode parecer contra-intuitivo, uma vez que se espera que quanto maior a LID, maior seja a concentração de distâncias. No entanto, faz-se necessário considerar também a construção dos Conjuntos de Influência Forte, que são o critério adicional de poda para consultas por similaridade diversificada.

Como o raio de cobertura dessas regiões é definido em função da distância entre o elemento no conjunto resposta (influenciador) e o objeto de consulta, esse raio tende a crescer com o aumento do número de vizinhos. Consequentemente, a "área" de exclusão de candidatos influenciados também aumenta. Nesse sentido, os resultados da Figura 5.3 sugerem que a proporção do crescimento desse raio da região influenciada está relacionado a dimensionalidade intrínseca do conjunto consultado, *i.e.*, quanto menor a LID, maior o crescimento do raio que define os Conjuntos de Influência Forte e, consequentemente, maior a quantidade de elementos descartados.

Para confirmar essa observação, foi realizado um segundo experimento que mede o tamanho do raio do k-ésimo Conjunto de Influência Forte gerado pelo *diversity browsing* para cada um dos conjuntos e *folds* examinados, *i.e.*, $\xi = \delta(o_q, o_k)$. Nesse experimento, o valor de k foi limitado ao valor mediano truncado de cada intervalo de LID obtido da Figura 5.3, pois o número k de vizinhos é maior para objetos de consulta do último

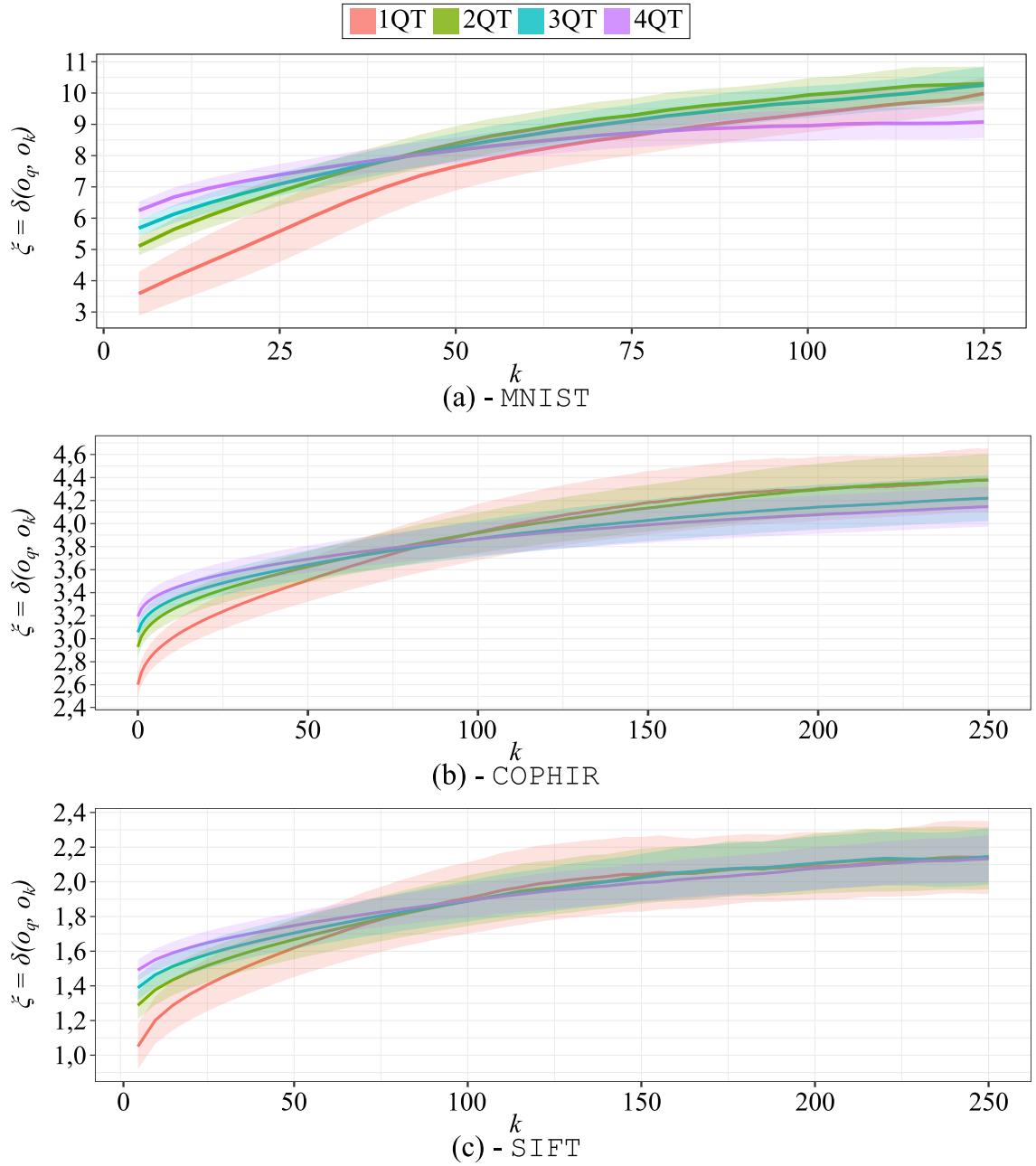


Figura 5.4: Mediana e limites inferior e superior do segundo e terceiro quartil para a distribuição dos raios de cobertura dos k-ésimos Conjuntos de Influência Forte para todos os conjuntos de dados e intervalos de LID examinados.

fold. A Figura 5.4 apresenta os resultados dessa avaliação considerando a consolidação do segundo e terceiro quartis construídos para a distribuição de raios máximos gerados pelos objetos de consulta para cada valor de vizinhança.

Os resultados indicam que os raios das regiões de Influência são menores para valores iniciais de k e crescem rapidamente para objetos de consulta de baixa LID. Esse comportamento vai se invertendo com o aumento da dimensionalidade intrínseca, como se vê no comportamento do algoritmo *diversity browsing* na Figura 5.3. De forma análoga, os

raios de regiões de Influência para conjunto de busca de alta dimensionalidade intrínseca crescem de forma menos acelerada com a quantidade de vizinhos. Esse resultado está em conformidade com o comportamento esperado para distâncias concentradas (alta LID), dado que *(i)* a função de distância tem um intervalo de valores mais restrito para assumir do que em cenários não-concentrados (baixa LID); *(ii)* os valores dos raios precisam ser monotonicamente crescentes pois os elementos no conjunto resposta são tais que $\delta(o_q, o_{i-1}) \leq \delta(o_q, o_i) \leq \delta(o_q, o_k)$ pela Definição 2; e *(iii)* raios menores geram regiões de exclusão também menores.

Cabe ressaltar também que em todos os casos analisados, os crescimentos das regiões de exclusão apresentaram um comportamento sublinear, sendo que os Conjuntos de Influência Forte aumentaram muito rapidamente com k para *folds* de baixa LID cobrindo rapidamente o espaço de busca. O crescimento para objetos de alta LID apresentou um comportamento mais suave, mas também sublinear. Além disso, os resultados também mostraram que o crescimento foi mais acentuado para conjuntos de maior dimensionalidade intrínseca global (`COPHIR`– 15, `SIFT`– 14 e `MNIST`–12). De forma geral, os resultados apresentados nas Figuras 5.3 e 5.4 indicam que:

- o número de vizinhos diversificados é menor que o número de vizinhos por similaridade, sendo que a ordem dessa diferença depende da cardinalidade do conjunto e da forma de crescimentos dos Conjuntos de Influência Forte;
- quanto maior a dimensionalidade intrínseca local, mais vizinhos diversificados podem ser encontrados, *i.e.*, há uma correlação suave entre o número de vizinhos diversificados e a LID; e
- o critério de Influência tende a gerar *(i)* regiões de exclusão maiores para conjuntos de baixa LID e (ii) regiões de exclusão menores para conjuntos de alta LID, suavizando o relacionamento de proximidade entre os objetos.

A combinação desses resultados com os encontrados no experimento anterior indicam que consultas por similaridade diversificada permitem discernir mais criteriosamente vizinhos próximos em espaços de alta dimensionalidade a um custo computacional que tende a ser parecido à similaridade para intervalos crescentes de LIDs.

Um último ponto em aberto que pode se beneficiar de uma análise experimental é se o ajuste fino de estruturas de indexação métricas pode melhorar o desempenho do algoritmo *diversity browsing*, dado que a capacidade de poda de partições desses índices se degenera com o aumento da dimensionalidade intrínseca do conjunto pesquisado.

5.3.3 Ajuste fino de índices métricos por LID

Nesse experimento, foi considerado o ajuste fino de árvores VP-Tree tais como estudadas nas seções anteriores. Em particular, os resultados da Seção 4.4 indicam que o melhor e o pior ajuste para a VP-Tree na execução do algoritmo *diversity browsing* foram as configurações `VP_MAX_BAL` e `VP_RND_UBAL`, respectivamente. As Figuras 5.5, 5.6 e 5.7 comparam a distribuição dos cálculos de distância e de tempos de execução de consultas nessas duas configurações de índice para cada intervalo de LID e valores de vizinhança $k = \{5, 10, 15, 20, 25\}$ nos conjuntos `MNIST`, `COPHIR` e `SIFT`, respectivamente. As distribuições foram podadas em 5% para remoção de valores extremos (que não contribuem no entendimento das distribuições dos valores coletados nos experimentos) e cada medida de tempo representa o valor médio coletado após cinco reexecuções das consultas.

Os resultados mostram que há diferenças no desempenho das consultas que variam em função da dimensionalidade intrínseca local. Em todos os casos, o tempo de resposta e a quantidade de cálculos de distância é menor para objetos de consulta de baixa LID do que para objetos de alta LID, como pode ser observado pela separação de massas das distribuição nas Figuras 5.5, 5.6 e 5.7. Mais do que isso, as massas de probabilidade se deslocam da menor para a maior LID em ordem crescente na ampla maioria das consultas

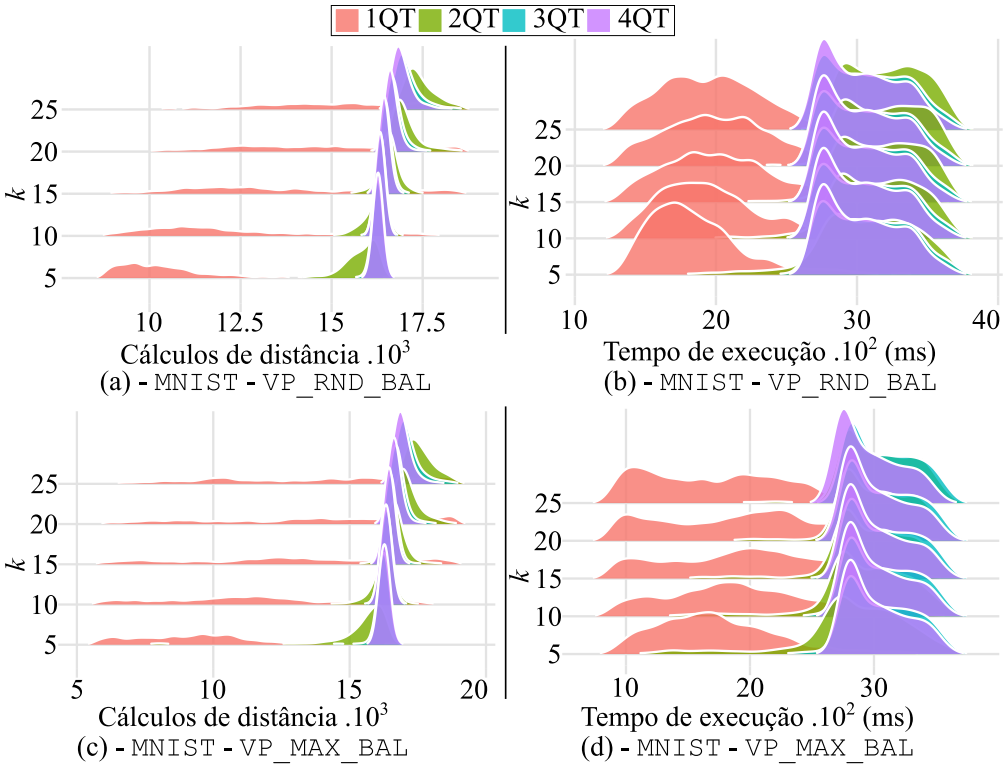


Figura 5.5: Distribuição de cálculos de distância e tempos de execução do conjunto `MNIST`.

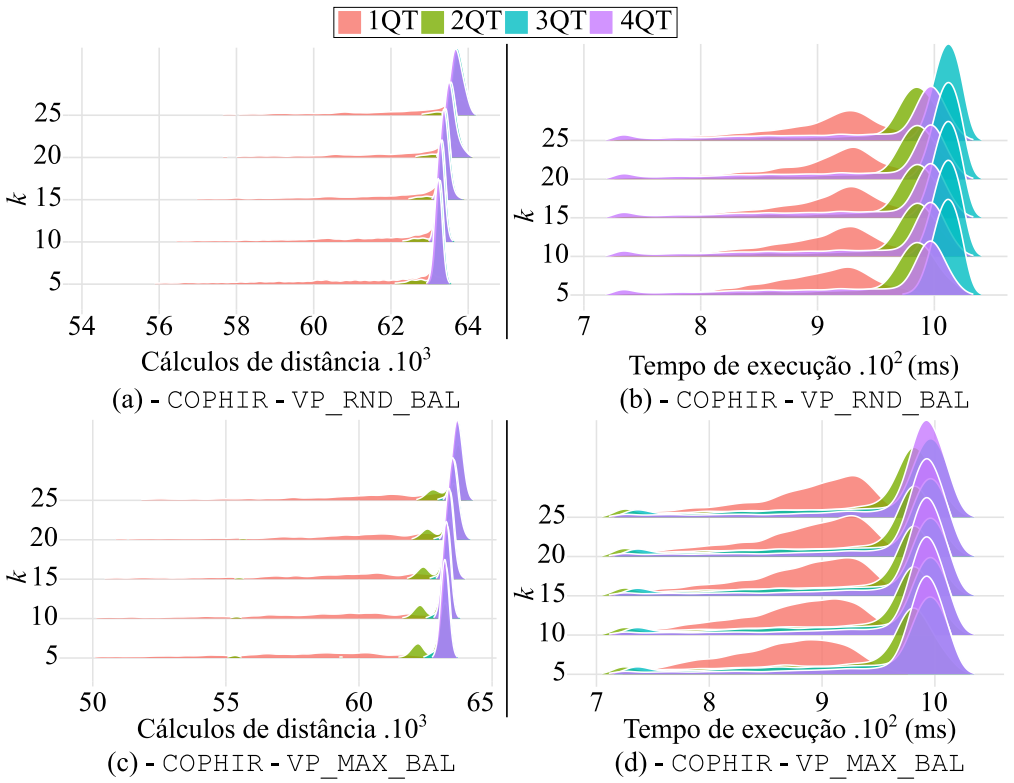


Figura 5.6: Distribuição de cálculos de distância e tempos de execução do conjunto `COPHIR`.

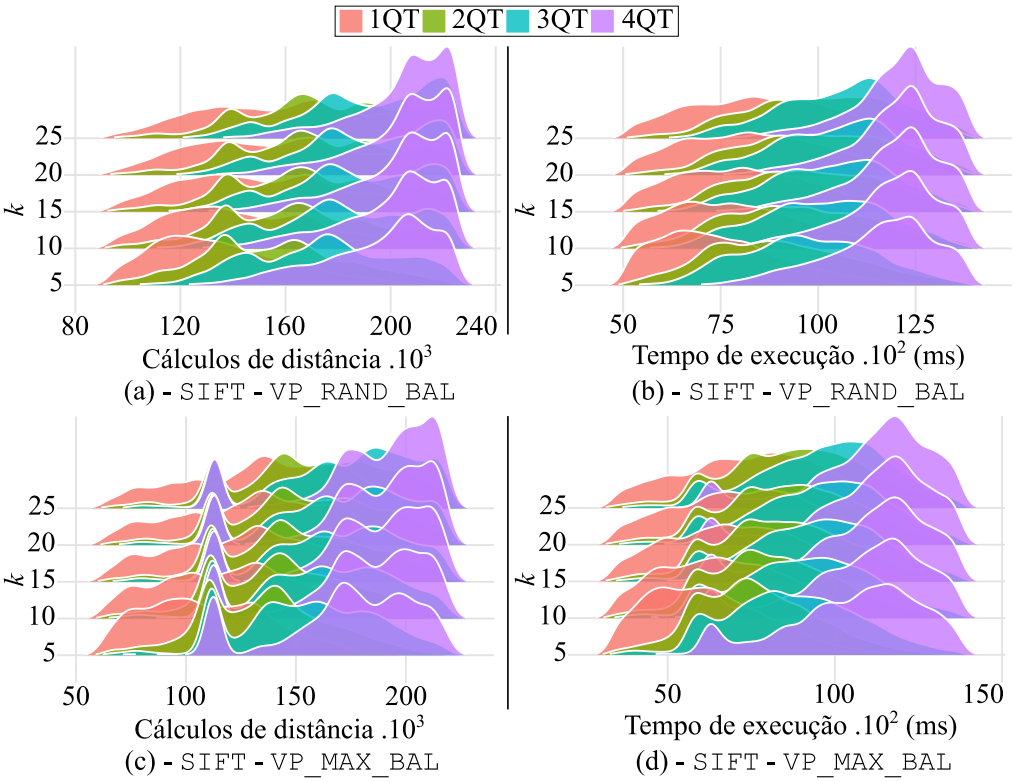


Figura 5.7: Distribuição de cálculos de distância e tempos de execução do conjunto `SIFT`.

realizadas e independentemente do ajuste do índice métrico, o que reflete a forte correlação entre a LID e o custo computacional da consulta de forma análoga ao que já foi reportado

em outros estudos para consultas apenas por similaridade [4,6]. Não obstante, diferenças entre os desempenhos derivados das duas configurações de índice podem ser observadas a partir das distribuições de cálculos de distância e tempos de execução.

Por exemplo, para objetos de consulta "fáceis" (1QT), o *diversity browsing* com o índice `VP_MAX_BAL` foi até 76%, 72% e 67%, mais rápido do que o índice `VP_RND_UBAL` para consultar os conjuntos `MNIST`, `COPHIR` e `SIFT`, respectivamente. Embora esses tenham sido os ganhos observados para o *fold* mais fácil, os ganhos do índice ajustado `VP_MAX_BAL` sobre a configuração `VP_RND_UBAL` também foram consistentes nos demais *folds* analisados, incluindo ganhos de até 74%, 62% e 69% no *fold* 4QT para os mesmos conjuntos de dados.

A Figura 5.8 consolida os ganhos, consulta por consulta, do índice mais ajustado `VP_MAX_BAL` sobre o índice menos ajustado `VP_RND_UBAL`. Nesse gráfico de barras acumuladas, soma-se o valor 1 dividido pelo número total de objetos de consulta para cada busca por vizinhança diversificada onde a configuração `VP_MAX_BAL` foi mais rápida em retornar o conjunto resposta do que o índice menos ajustado. Portanto, o valor máximo de ganho acumulado no eixo "Proporção" é 1, ao passo que uma proporção próxima a 0,5 indica que não há diferença entre usar uma ou outra configuração, já que os dois índices têm a mesma chance de retornar o conjunto resposta com melhor desempenho. As linhas tracejadas nos gráficos correspondem aos pontos onde o desempenho dos índices é equivalente (por *fold*).

Os resultados indicam que a configuração `VP_MAX_BAL` foi superior à versão sem ajuste do índice, tendo maior ganho acumulado em consultas realizadas em todos os conjuntos, intervalos de LID e valores de vizinhança. Os ganhos foram substancialmente maiores para os menores valores de LID, reduzindo-se com o aumento desses intervalos. Para os valores de alta LID, os menores ganhos foram encontrados no conjunto de maior dimensionalidade

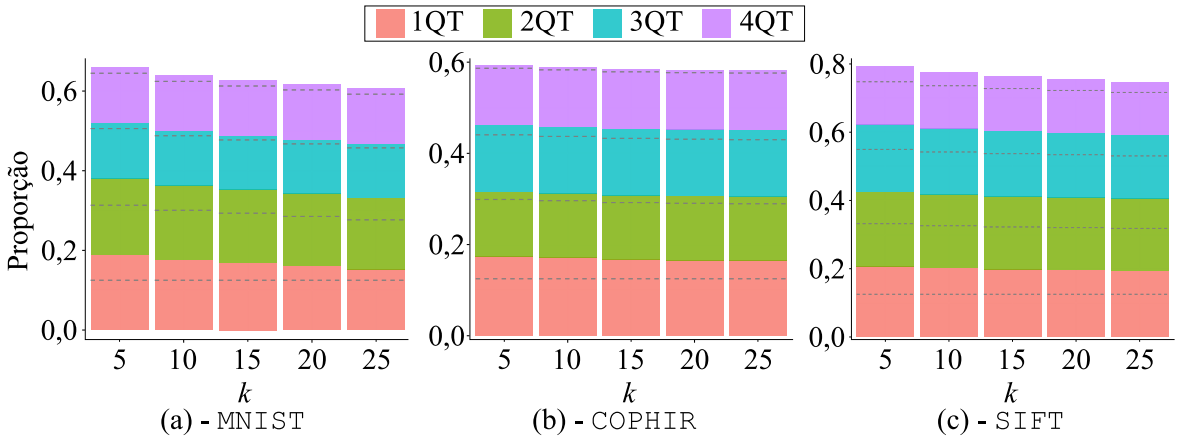


Figura 5.8: Proporção de consultas onde o algoritmo *diversity browsing* com índice `VP_MAX_BAL` foi mais rápido do que com o índice `VP_RND_UBAL`.

intrínseca global COPHIR, onde os desempenhos chegaram próximos da equivalência. Por último, mas não menos importante, os resultados indicam que a medida que se aumenta a quantidade de vizinhos ($k \to \infty$) os ganhos dos ajustes tendem a desaparecer no último intervalo de LID, sendo que essa observação está alinhada com as indicações prévias da literatura para valores crescentes de LID e vizinhança.

Ainda assim, o ajuste fino do método de indexação se mostrou capaz de otimizar a maior parte das consultas para valores crescentes de vizinhança com destaque para os menores valores de LID. Esses resultados mostram um caminho inicial para que otimizadores de consulta possam melhorar ainda mais o desempenho de diversificação de resultados se valendo tanto da LID estimada para o objeto de consulta quanto da abordagem *diversity browsing* para definir um ambiente onde os conjuntos de dados sejam divididos em pedaços que serão consultados por uma rotina distribuída que possa implementar os critérios de vizinhança e Influência. Em resumo, os resultados experimentais indicam que:

- consultas por objetos de consulta de baixa LID são executadas pelo *diversity browsing* de forma muito mais eficiente do que por objetos de alta LID independentemente da estrutura de índice;
- o ajuste fino de estruturas de indexação melhora a execução das consultas para a maioria das LIDs, porém esse aumento de desempenho é mais acentuado em objetos de baixa e média LID; e
- o ajuste fino de estruturas de indexação tende a não compensar para valores crescentes de vizinhança e dimensionalidade intrínseca.

5.4 Visualização de dados por diversidade e LID

Uma contribuição adicional do uso do algoritmo *diversity browsing* para consultar espaços de alta dimensionalidade é que os conceitos de proximidade e Influência podem ser combinados para explorar visualmente o conjunto de dados consultado. A ideia por trás dessa contribuição é que os Conjuntos de Influência Forte podem ser usados para "agrupar" elementos influenciados que serão usados para representar a densidade de elementos similares aos vizinhos mais próximos. Na sequência, o método de redução Análise de Componentes Principais é empregado para representar o objeto de consulta e sua vizinhança na forma de um gráfico de bolhas diversificadas[5].

[5]Analogamente, outro método de redução de dimensionalidade como o Kernel PCA ou FastMap [1,35] pode ser acoplado para permitir a visualização da vizinhança em duas ou três dimensões.

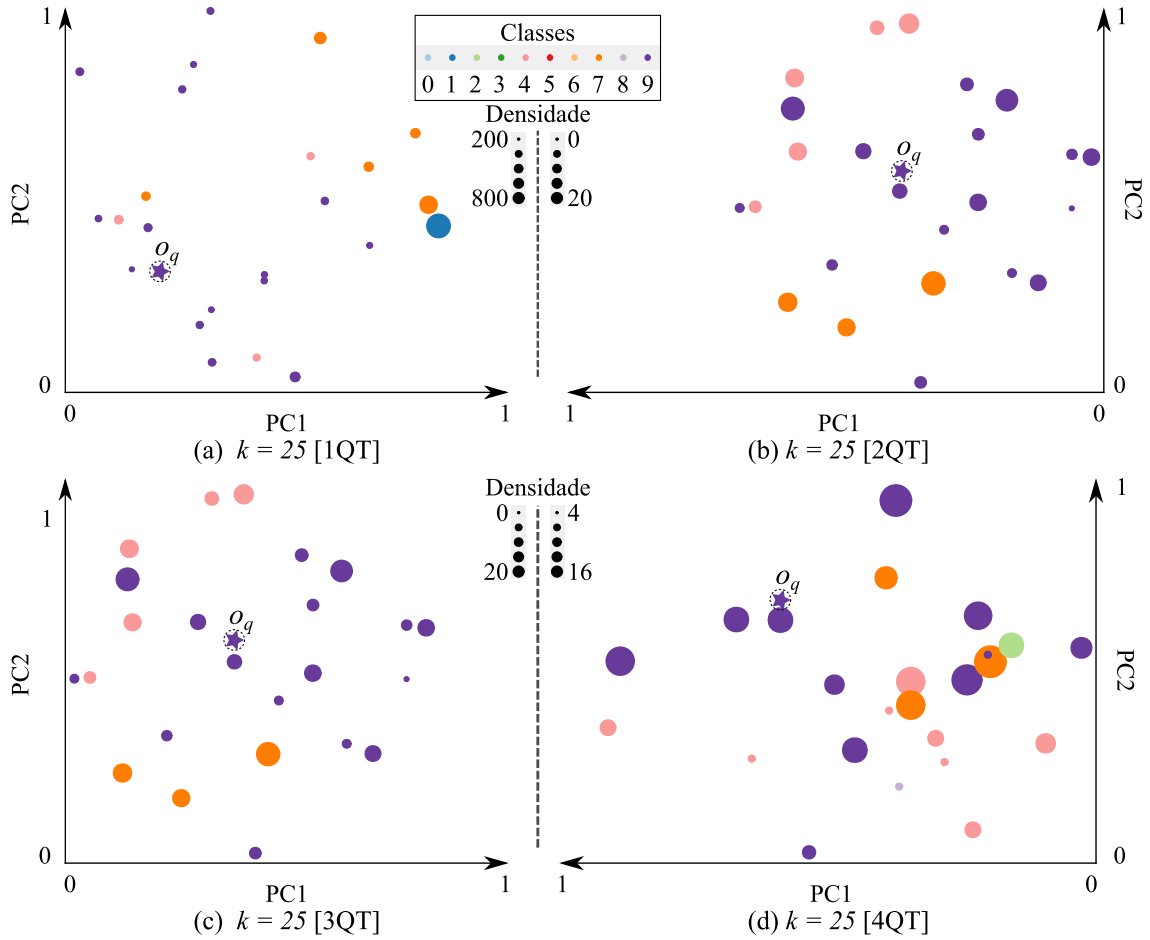


Figura 5.9: Visualização de uma vizinhança diversificada $k = 25$ para um mesmo objeto de consulta sobre quatro *folds* construídos para o conjunto de dados `MNIST`.

A visualização mais detalhada dos dados apresenta as vizinhanças a partir das perspectivas dos quatro intervalos de LID construídos (1QT, 2QT, 3QT e 4QT), pois os experimentos prévios indicaram que o número de vizinhos diversificados e suas densidades variam sensivelmente de acordo com as LIDs. A Figura 5.9(a–d) apresenta o gráfico de bolhas diversificado para um mesmo objeto de consulta o_q que representa o dígito 9 considerando os quatro *folds* construídos para o conjunto de dados `MNIST` e uma vizinhança de $k = 25$. O gráfico apresentado também se beneficia do conjunto `MNIST` ser rotulado e apresenta as densidades em escala dos vizinhos de acordo com uma cor atribuída a cada rótulo e elemento influenciador.

No exemplo da Figura 5.9, a visualização é centrada em o_q e preserva as proximidades semânticas da métrica L_2. As densidades baseadas em Influência no entorno dos vizinhos é representada pelo diâmetro das bolhas e são calculadas como o de elementos que caem dentro do Conjunto de Influência Forte definido pelo vizinho e pelo objeto de consulta. Elementos influenciados por mais de um vizinho são contados apenas uma única vez

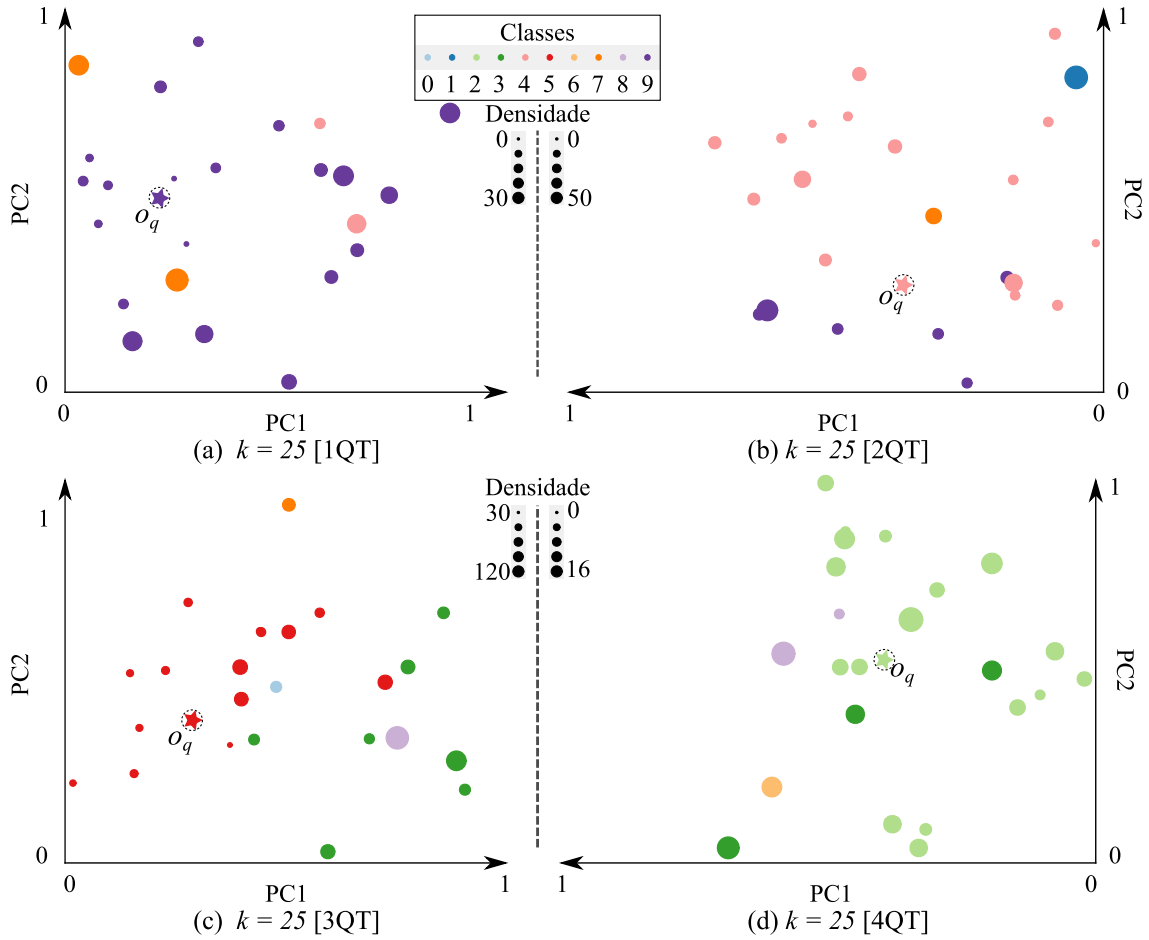
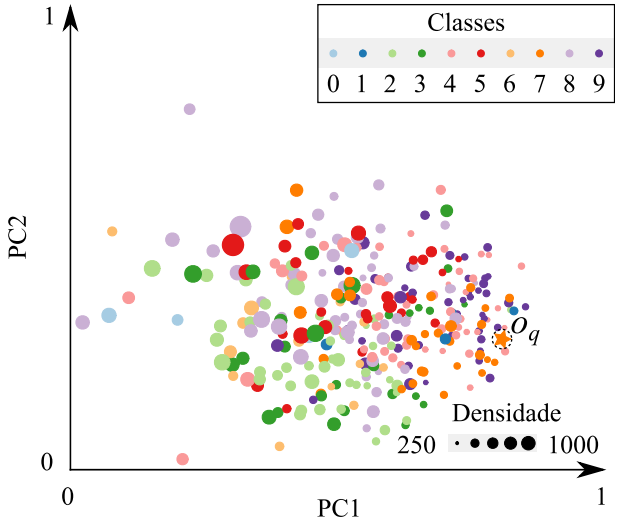


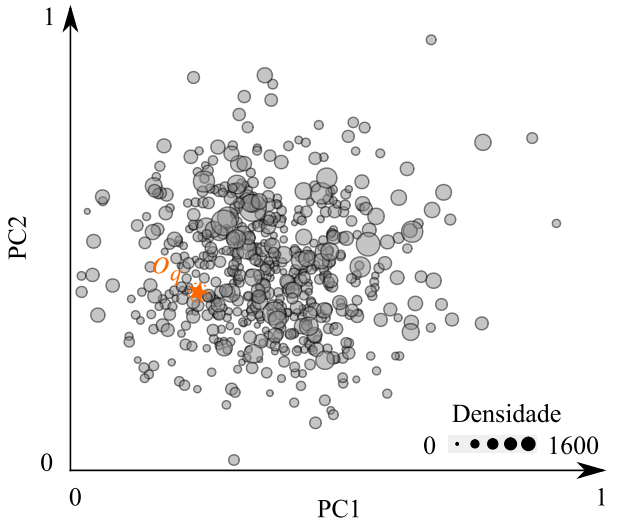
Figura 5.10: Visualização de uma vizinhança diversificada $k = 25$ para diversos objetos de consulta sobre quatro *folds* construídos para o conjunto de dados `MNIST`.

dentro da densidade do vizinho mais próximo com empates quebrados aleatoriamente. A visualização do primeiro *fold* mostra que os rótulos $9, 7$ e 4 são retornados pelo *diversity browsing*, enquanto que os rótulos 9 e 7 sãos os mais frequentes para o *fold* 2QT. As visualizações dos conjuntos de maior LID 3QT e 4QT mostram que uma maior variedade pode ser obtida, mas a semântica de vizinhança ainda pode ser encontrada pois os rótulos mais frequentes continuam sendo 9 e 7 com densidades mais uniformes.

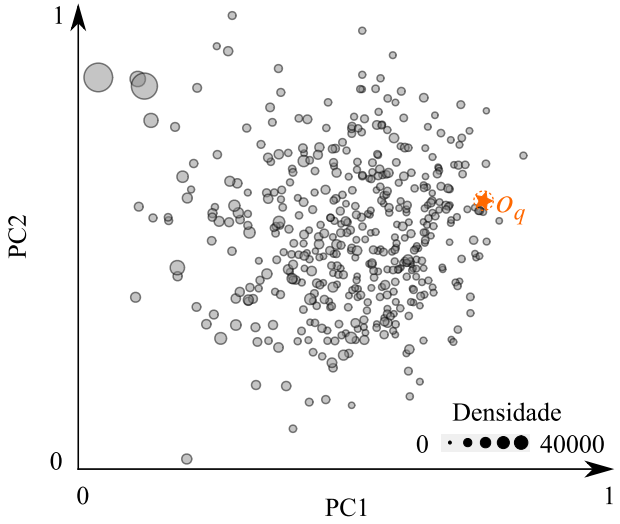
A Figura 5.10(a–d) apresenta outro exemplo do gráfico de bolhas diversificado para diferentes objetos de consulta representando classes distintas em diferentes intervalos de LID para o conjunto de dados `MNIST` e uma vizinhança de $k = 25$. Diferentemente da Figura 5.9, a visualização na Figura 5.10 permite observar as diferentes densidades nos *folds*, bem como permite explorar as classes que se encontram próximas dentro de cada intervalo de LID. Como pode ser observado na figura, os objetos podem ser classificados para além de apenas distância e classes próximas são alcançadas com baixo valor de vizinhança, *e.g.*, $k = 25$, em um conjunto de $70K$ instâncias.



(a) MNIST - $k \to \infty$ [4QT]



(b) COPHIR - $k \to \infty$ [4QT]



(c) SIFT - $k \to \infty$ [4QT]

Figura 5.11: Visualização de uma vizinhança diversificada $k \to \infty$ para um objeto de consulta aleatório junto aos *folds* 4QT dos conjuntos MNIST, COPHIR e SIFT, respectivamente.

Uma generalização da visualização de vizinhança diversificada pode ser obtida para o número de vizinhos $k \to \infty$. A Figura 5.11 mostra o resultado de uma busca desse tipo para objetos de consulta escolhidos aleatoriamente e com relação aos valores mais altos de LID nos conjuntos `MNIST`, `COPHIR` e `SIFT`. Nesse caso, os elementos retornados compõem uma amostra representativa na qual as distâncias para o_q em duas dimensões são discerníveis da perspectiva de separação linear para a distância L_2 do método PCA.

Essa visualização também indica que não apenas a dimensionalidade de conjuntos de alta LID deve ser reduzida para melhor representação semântica de distâncias, mas também que a cardinalidade do conjunto de dados precisa ser reduzida de forma que os objetos influenciados possam ser descartados da análise exploratória possibilitando a preservação do sentido de proximidade na mineração visual dos dados.

5.5 Conclusões parciais

Essa seção discutiu experimentalmente os efeitos da LID sobre o algoritmo *diversity browsing* de acordo com diversas perspectivas. Os resultados experimentais permitiram avançar o estado-da-arte de diversificação de resultados em espaços de alta dimensionalidade, com as seguintes contribuições:

1. apesar de serem potencialmente mais custosas do que consultas apenas por similaridade, consultas por similaridade diversificada executadas com o *diversity browsing* se tornam proporcionalmente mais econômicas para conjuntos de LIDs maiores na comparação com LIDs menores;

2. o algoritmo *diversity browsing*, contra-intuitivamente, é propenso a encontrar mais vizinhos diversificados em espaços com maior LID, dado que os raios de Influência gerados a partir dos elementos incluídos no conjunto resposta crescem de forma mais suave para altas LIDs do que em baixas LIDs;

3. a parametrização adequada de índices métricos é um tópico também relevante para o *diversity browsing*, sendo que os maiores ganhos decorrentes desses ajustes ocorrem em objetos de consulta de baixa e média LID; e

4. o algoritmo *diversity browsing* permite explorar visualmente o espaço de busca de uma nova forma que combina a visualização dos dados de alta dimensionalidade de acordo com diferentes intervalos de LID e densidade de vizinhança.

Os resultados reportados nessa seção corroboram a última parte da Hipótese de Pesquisa dessa dissertação, pois, para além dos ganhos de desempenho, a comparação experimental do *diversity browsing* com a rotina de busca apenas por similaridade indicou potenciais ganhos semânticos oriundos do uso de consultas por similaridade diversificada para buscas em espaços de alta dimensionalidade.

Capítulo 6

Conclusões

Esse trabalho apresentou um estudo sobre como consultas por vizinhança modeladas com a Teoria de Espaços Métricos podem ser estendidas para consultas por vizinhança diversificada, não apenas do ponto de vista de implementação mas de formalização de uma rotina de busca que atualiza o algoritmo de consulta indexado estado-da-arte da literatura. A construção da proposta foi realizada de acordo com as indicações dos trabalhos revisados para a obtenção de uma solução incremental, indexável, e que permitisse manter características semânticas desejáveis de um método de diversificação de resultados, tais como o controle da cardinalidade do conjunto resposta, a ausência de parâmetros externos fornecidos pelo usuário e o balanceamento dinâmico entre similaridade e diversidade.

Para além de apenas propor um método que estende o estado-da-arte em termos de eficiência computacional, as soluções propostas foram avaliadas em cenários limite de consultas sobre espaços de alta dimensionalidade, onde espera-se que métodos indexados tenham um baixo desempenho computacional e semântico. Para essa avaliação foi utilizado o conceito de LID, que permitiu que os testes experimentais fossem equitativamente particionados por nível de dificuldade.

Os resultados experimentais encontrados indicaram que o método proposto suaviza os efeitos da concentração de distância sobre consultas por similaridade, o que permite uma execução mais eficiente e a recuperação de elementos próximos com maior estabilidade. Alguns desses resultados, aparentemente contra-intuitivos, mostraram como o método induziu essas suavizações, incluindo o aumento da diversidade em conjuntos de dados com alta LID e o aumento da densidade de vizinhança em espaços de baixa LID.

Ao final do estudo conduzido, foi possível corroborar teórica e experimentalmente a Hipótese de Pesquisa levantada no início dessa dissertação, pois o algoritmo de busca

proposto, acoplado a um índice métrico aumentou expressivamente o desempenho de consultas por similaridade diversificada. O método usa o particionamento de um índice métrico para descartar regiões que não satisfazem simultaneamente os critérios *(i)* de similaridade para o objeto consultado, e *(ii)* dissimilaridade para os elementos no conjunto resposta. Além disso, a comparação experimental do algoritmo proposto contra os algoritmos por similaridade existentes em espaços de alta dimensionalidade deixou em evidência os potenciais ganhos semânticos nesses cenários de busca. De forma geral, a lista de contribuições dessa dissertação inclui os seguintes itens:

1. o projeto do algoritmo *diversity browsing*, que estende o algoritmo estado-da-arte *distance browsing* para consultas vizinhanças diversificadas com o apoio do conceito de Influência;

2. a implementação do algoritmo *diversity browsing* junto ao índice métrico VP-Tree, que é uma estrutura barata de se construir, de fácil ajuste fino e particularmente eficiente para consultas em memória principal;

3. uma comparação experimental do *diversity browsing* contra os métodos representativos da literatura GMC, GNE e BRID$_k$, onde o método proposto foi:

   - mais eficiente em cálculos de distância e tempo de execução do que o competidor BRID$_k$ por uma diferença percentual expressiva; e
   - mais eficiente em cálculos de distância e tempo de execução do que os competidores GMC e GNE por ordens de magnitude;

4. uma ampla avaliação do método *diversity browsing* em espaços de alta dimensionalidade particionados de acordo com intervalos de LID que indicou que:

   - consultas executadas com o *diversity browsing* se tornam proporcionalmente mais econômicas do que consultas com a abordagem *distance browsing* para conjuntos de LIDs maiores na comparação com LIDs menores;
   - há uma correlação entre LID e o número de vizinhos que podem ser encontrados. Por isso, o *diversity browsing* encontra mais vizinhos em espaços de maior LID e os raios de Influência nesses espaços crescem de forma mais suave do que em espaços de baixas LIDs;
   - o ajuste fino de índices métricos é um tópico que continua relevante para consultas por vizinhança diversificada pois permitem obter ganhos importantes em buscas de objetos de baixa e média LID;

- o critério de pivôs se mostrou dominante para o ajuste da estrutura de índice em todos os experimentos realizados, sendo o critério da variância máxima o que apresentou os melhores resultados;

5. a visualização, por meio do algoritmo *diversity browsing*, do espaço de busca de alta dimensionalidade por intervalo de LID e com a perspectiva do objeto de consulta, incluindo seus vizinhos e densidade calculada por Influência.

6.1 Limitações

Ainda que o algoritmo proposto seja eficiente o suficiente para executar consultas por similaridade diversificada com o apoio de índices métricos, a proposta não é capaz de:

- Executar consultas de forma distribuída. O método proposto é fortemente baseado na premissa de ordenação por distância (Lemma 2) e não possui atualmente uma representação algorítmica paralela capaz de lidar com dados distribuídos seguindo uma estratégia Map-Reduce.
- Ainda que muito mais resistente do que o *distance browsing*, o algoritmo *diversity browsing* não é imune à concentração de distâncias e pode perder eficiência e semântica em conjuntos de busca de alta dimensionalidade intrínseca.

6.2 Publicações

Essa seção reporta os artigos publicados ou submetidos relacionados a essa dissertação. Primeiramente, são apresentadas as publicações cujo conteúdo é produto direto dessa pesquisa e, na sequência, são indicadas as publicações decorrentes de pesquisa em colaboração sobre consultas por similaridade diversificada.

Artigos oriundos dessa dissertação

- (Artigo completo em conferência – **Publicado**). Jasbick, D., Santos, L., Oliveira, D., Bedo, M. *Some branches may bear rotten fruits: Diversity browsing vp-trees.* In *Similarity Search and Applications* (SISAP'20). Springer, p. 140–154 [49].
  Remoto – Copenhagen/Dinamarca.

- (Artigo completo em revista – **Em fase de submissão**). Jasbick, D., Santos, L., Oliveira, D., Bedo, M. *Pushing diversity into higher dimensions: The LID effect on diversified similarity searching*, p. 1–20.

Trabalhos em colaboração

- (Artigo completo em conferência – **Publicado**)[1]. Lopes, C., Jasbick, D., Bedo, M., Santos, L. *Quality metrics for diversified similarity searching: What they stand for?* In *Simpósio Brasileiro de Bancos de Dados* (SBBD'20), SBC, p. 1–12 [54]. Remoto – Rio de Janeiro/RJ – Brasil.

- (Artigo completo em revista – **Submetido**). Lopes, C., Santos, L., Jasbick, D., Oliveira, D., Bedo, M. *An empirical assessment of quality metrics for diversified similarity searching.* In *Journal of Information and Data Management.* SBC, p. 1–16.

6.3 Trabalhos Futuros

Os resultados desse trabalho apontam novas direções de investigação para tratar a otimização de consultas por similaridade diversificada. Em particular, foram abertos os seguintes novos espaços para estudos que se pretende investigar no futuro:

- Extensão do *diversity browsing* para ambiente de computação de alto desempenho, com dados sendo parcialmente filtrados localmente e posteriormente integrados por proximidade e Influência.

- Uso da LID para particionar conjuntos de alta dimensionalidade em ambientes de computação de alto desempenho, sendo que diferentes métodos de acesso podem ser vinculados a diferentes porções dos dados por um método otimizador de consulta para balanceamento de carga e processamento.

[1]Esse trabalho recebeu menção honrosa na edição SBBD'20 – Realizado na modalidade remoto.

Referências

[1] Aggarwal, C. *Data mining: The Textbook*. Springer, 2015.

[2] Agrawal, R., Gollapudi, S., Halverson, A., Ieong, S. Diversifying search results. In *Web Search and Data Mining* (2009), ACM, p. 5–14.

[3] Amsaleg, L., Chelly, O., Furon, T., Girard, S., Houle, M., Kawarabayashi, K., Nett, M. Estimating local intrinsic dimensionality. In *Special Interest Group on Knowledge Discovery in Data* (2015), ACM, p. 29–38.

[4] Amsaleg, L., Chelly, O., Houle, M., Kawarabayashi, K., Radovanović, M., Treeratanajaru, W. Intrinsic dimensionality estimation within tight localities. In *International Conference on Data Mining* (2019), SIAM, p. 181–189.

[5] Angel, A., Koudas, N. Efficient diversity-aware search. In *International Conference on Management of Data* (2011), ACM, p. 781–792.

[6] Aumüller, M., Ceccarello, M. The role of local intrinsic dimensionality in benchmarking nearest neighbor search. In *International Conference on Similarity Search and Applications* (2019), Springer, p. 113–127.

[7] Bac, J., Zinovyev, A. Local intrinsic dimensionality estimators based on concentration of measure. In *International Joint Conference on Neural Networks* (2020), IEEE, p. 1–8.

[8] Barbará, D., DuMouchel, W., Faloutsos, C., Haas, P., Hellerstein, J., Ioannidis, Y., Jagadish, H., Johnson, T., Ng, R., Poosala, V. The New Jersey data reduction report. In *Data Engineering Bulletin* (1997), IEEE.

[9] Bayer, R., McCreight, E. Organization and maintenance of large ordered indexes. In *Software Pioneers*. Springer, 2002, p. 245–262.

[10] Beyer, K., Goldstein, J., Ramakrishnan, R., Shaft, U. When is "nearest neighbor" meaningful? In *International Conference on Database Theory* (1999), Springer, p. 217–235.

[11] Beygelzimer, A., Kakade, S., Langford, J. Cover trees for nearest neighbor. In *International Conference on Machine Learning* (2006), ACM, p. 97–104.

[12] Borodin, A., Lee, H., Ye, Y. Max-sum diversification, monotone submodular functions and dynamic updates. In *Symposium on Principles of Database Systems* (2012), ACM, p. 155–166.

[13] Bozkaya, T., Ozsoyoglu, M. Distance-based indexing for high-dimensional metric spaces. In *International Conference on Management of Data* (1997), ACM, p. 357–368.

[14] Capannini, G., Nardini, F., Perego, R., Silvestri, F. Efficient diversification of web search results. In *International Conference on Very Large Data Bases* (2011), vol. 4, ACM, p. 451–459.

[15] Carbonell, J., Goldstein, J. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *International Conference on Research and Development in Information Retrieval* (1998), ACM, p. 335–336.

[16] Casanova, G., Englmeier, E., Houle, M., Kröger, P., Nett, M., Schubert, E., Zimek, A. Dimensional testing for reverse k-nearest neighbor search. In *International Conference on Very Large Data Bases* (2017), vol. 10, ACM, p. 769–780.

[17] Catallo, I., Ciceri, E., Fraternali, P., Martinenghi, D., Tagliasacchi, M. Top-k diversity queries over bounded regions. In *Transactions on Database Systems* (2013), vol. 38, ACM.

[18] Chávez, E., Navarro, G., Baeza-Yates, R., Marroquín, J. L. Searching in metric spaces. In *Computing Surveys* (2001), vol. 33, ACM, p. 273–321.

[19] Chen, L., Gao, Y., Li, X., Jensen, C., Chen, G. Efficient metric indexing for similarity search. In *International Conference on Data Engineering* (2015), IEEE, p. 591–602.

[20] Chen, L., Gao, Y., Li, X., Jensen, C., Chen, G. Efficient metric indexing for similarity search and similarity joins. In *Transactions on Knowledge and Data Engineering* (2017), vol. 29, IEEE, p. 556–571.

[21] Chen, L., Gao, Y., Song, X., Li, Z., Miao, X., Jensen, C. Indexing metric spaces for exact similarity search. *arXiv preprint arXiv:2005.03468* (2020).

[22] Chen, L., Gao, Y., Zheng, B., Jensen, C., Yang, H., Yang, K. Pivot-based metric indexing. In *International Conference on Very Large Data Bases* (2017), vol. 10, ACM, p. 1058–1069.

[23] Dang, V., Croft, B. Term level search result diversification. In *International Conference on Research and Development in Information Retrieval* (2013), ACM, p. 603–612.

[24] Deng, L. The mnist database of handwritten digit images for machine learning research [best of the web]. In *Signal Processing Magazine* (2012), vol. 29, IEEE, p. 141–142.

[25] Deng, T., Fan, W. On the complexity of query result diversification. In *International Conference on Very Large Data Bases* (2013), vol. 6, ACM, p. 577–588.

[26] Deza, M., Deza, E. Encyclopedia of distances. In *Encyclopedia of distances*. Springer, 2009, p. 1–583.

[27] Dou, Z., Yang, X., Li, D., Wen, J., Sakai, T. Low-cost, bottom-up measures for evaluating search result diversification. In *Information Retrieval Journal* (2020), vol. 23, Springer, p. 86–113.

[28] Drosou, M., Jagadish, H., Pitoura, E., Stoyanovich, J. Diversity in big data: A review. In *Big data* (2017), vol. 5, Mary Ann Liebert, Inc., p. 73–84.

[29] Drosou, M., Pitoura, E. Search result diversification. In *SIGMOD Record* (2010), vol. 39, ACM, p. 41–47.

[30] Drosou, M., Pitoura, E. Disc diversity: result diversification based on dissimilarity and coverage. *arXiv preprint arXiv:1208.3533* (2012).

[31] Drosou, M., Pitoura, E. Dynamic diversification of continuous data. In *International Conference on Extending Database Technology* (2012), ACM, p. 216–227.

[32] Drosou, M., Pitoura, E. Poikilo: a tool for evaluating the results of diversification models and algorithms. In *International Conference on Very Large Data Bases* (2013), vol. 6, ACM, p. 1246–1249.

[33] Drosou, M., Pitoura, E. Multiple radii disc diversity: Result diversification based on dissimilarity and coverage. In *Transactions on Database Systems* (2015), vol. 40, ACM, p. 1–43.

[34] Erba, V., Gherardi, M., Rotondo, P. Intrinsic dimension estimation for locally undersampled data. In *Scientific Reports* (2019), vol. 9, Nature, p. 1–9.

[35] Faloutsos, C., Lin, K. Fastmap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. In *International Conference on Management of Data* (1995), ACM, p. 163–174.

[36] Francois, D., Wertz, V., Verleysen, M. The concentration of fractional distances. In *Transactions on Knowledge and Data Engineering* (2007), vol. 19, IEEE, p. 873–886.

[37] Fu, A., Chan, P., Cheung, Y., Moon, Y. Dynamic VP-Tree indexing for n-nearest neighbor search given pairwise distances. In *International Conference on Very Large Data Bases* (2000), vol. 9, Springer, p. 154–173.

[38] Gao, R., Shah, C. Toward creating a fairer ranking in search engine results. In *Information Processing and Management* (2020), vol. 57, Elsevier, p. 102138.

[39] Garcia-Molina, H. *Database Systems: The Complete Book*. Pearson, 2008.

[40] Ge, X., Chrysanthis, P., Pelechrinis, K., Zeinalipour-Yazti, D., Sharaf, M. Serendipity-based points-of-interest navigation. In *Transactions on Internet Technology* (2020), vol. 20, ACM.

[41] Gollapudi, S., Sharma, A. An axiomatic approach for result diversification. In *International Conference on World Wide Web* (2009), ACM, p. 381–390.

[42] Hetland, M. The basic principles of metric indexing. In *Swarm intelligence for multi-objective problems in data mining*. Springer, 2009, p. 199–232.

[43] Hetland, M. Metrics and ambits and sprawls, oh my. In *Similarity Search and Applications* (2020), Springer, p. 126–139.

[44] Hjaltason, G., Samet, H. Index-driven similarity search in metric spaces (Survey Article). In *Transactions on Database Systems* (2003), vol. 28, ACM, p. 517–580.

[45] Houle, M. Dimensionality, discriminability, density and distance distributions. In *International Conference on Data Mining* (2013), IEEE, p. 468–473.

[46] Hu, S., Dou, Z., Wang, X., Sakai, T., Wen, J. Search result diversification based on hierarchical intents. In *International Conference on Information and Knowledge Management* (2015), ACM, p. 63–72.

[47] Jain, A., Sarda, P., Haritsa, J. Providing diversity in k-nearest neighbor query results. In *Advances in Knowledge Discovery and Data Mining* (2004), Springer, p. 404–413.

[48] Jasbick, D. Revisitando vp-trees: Estruturas de complexidade de busca sub-linear em espaços métricos. Relatório Técnico, Santo Antônio de Pádua, 2018.

[49] Jasbick, D., Santos, L., de Oliveira, D., Bedo, M. Some branches may bear rotten fruits: Diversity browsing vp-trees. In *Similarity Search and Applications* (2020), Springer, p. 140–154.

[50] Jiang, Z., Dou, Z., Zhao, W. X., Nie, J.-Y., Yue, M., Wen, J.-R. Supervised search result diversification via subtopic attention. In *Transactions on Knowledge and Data Engineering* (2018), vol. 30, IEEE, p. 1971–1984.

[51] Kaster, D., Bugatti, P., Traina, A., Traina Jr, C. Fmi-sir: A flexible and efficient module for similarity searching on oracle database. In *Journal of Information and Data Management* (2010), vol. 1, SBC, p. 229–229.

[52] Liu, J., Dou, Z., Wang, X., Lu, S., Wen, J. Dvgan: A minimax game for search result diversification combining explicit and implicit features. In *International Conference on Research and Development in Information Retrieval* (2020), ACM, p. 479–488.

[53] Liu, S., Wei, Y. Fast nearest neighbor searching based on improved vp-tree. In *Pattern Recognition Letters* (2015), vol. 60, Elsevier, p. 8–15.

[54] Lopes, C., Jasbick, D., Bedo, M., Santos, L. F. D. Quality metrics for diversified similarity searching: What they stand for? In *Simpósio Brasileiro de Bancos de Dados* (2020), SBC, p. 1–12.

[55] Lu, W., Hou, J., Yan, Y., Zhang, M., Du, X., Moscibroda, T. Msql: efficient similarity search in metric spaces using sql. In *International Conference on Very Large Data Bases* (2017), vol. 26, Springer, p. 829–854.

[56] Mei, Q., Guo, J., Radev, D. Divrank: The interplay of prestige and diversity in information networks. In *International Conference on Knowledge Discovery and Data Mining* (2010), ACM, p. 1009–1018.

[57] Micó, L., Oncina, J., Carrasco, R. C. A fast branch and bound nearest neighbour classifier in metric spaces. In *Pattern Recognition Letters* (1996), vol. 17, Elsevier, p. 731–739.

[58] Nargesian, F., Samulowitz, H., Khurana, U., Khalil, E., Turaga, D. Learning feature engineering for classification. In *International Joint Conference on Artificial Intelligence* (2017), p. 2529–2535.

[59] Nargesian, F., Zhu, E., Miller, R. J., Pu, K. Q., Arocena, P. C. Data lake management: Challenges and opportunities. In *International Conference on Very Large Data Bases* (2019), vol. 12, ACM, p. 1986–1989.

[60] Navarro, G., Paredes, R., Reyes, N., Bustos, C. An empirical evaluation of intrinsic dimension estimators. In *Information Systems* (2017), vol. 64, Elsevier, p. 206–218.

[61] Padmanabhan, D., Deshpande, P. *Operators for Similarity Search - Semantics, Techniques and Usage Scenarios*. Springer, 2015.

[62] Pearson, K. On lines and planes of closest fit to systems of points in space. In *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* (1901), vol. 2, Taylor and Francis, p. 559–572.

[63] Pestov, V. An axiomatic approach to intrinsic dimension of a dataset. In *Neural Networks* (2008), vol. 21, Elsevier, p. 204–213.

[64] Pestov, V. Is the k-nn classifier in high dimensions affected by the curse of dimensionality? In *Computers and Mathematics with Applications* (2013), vol. 65, Elsevier, p. 1427–1437.

[65] Pestov, V. Lower bounds on performance of metric tree indexing schemes for exact similarity search in high dimensions. In *Algorithmica* (2013), vol. 66, Springer, p. 310–328.

[66] Pestov, V. Elementos da teoria de aprendizagem de máaquina supervisionada. *arXiv preprint arXiv:1910.06820* (2019).

[67] Pickett, S., Luttmann, C., Guerin, V., Laoui, A., James, E. Divsel and complib - strategies for the design and comparison of combinatorial libraries using pharmacophoric descriptors. In *Journal of Chemical Information and Computer Sciences* (1998), vol. 38, American Chemical Society, p. 144–150.

[68] Pisinger, D. Upper bounds and exact algorithms for p-dispersion problems. In *Computers and Operations Research* (2006), vol. 33, Elsevier, p. 1380–1398.

[69] Pitoura, E., Tsaparas, P., Flouris, G., Fundulaki, I., Papadakos, P., Abiteboul, S., Weikum, G. On measuring bias in online information. In *SIGMOD Record* (2018), vol. 46, ACM, p. 16–21.

[70] Polinsky, A., Feinstein, R., Shi, S., Kuki, A. Librain: Software for automated design of exploratory and targeted combinatorial libraries. In *Molecular diversity and combinatorial chemistry: Libraries and drug discovery* (1996), vol. 996, American Chemical Society, p. 219–232.

[71] Qin, L., Yu, J. X., Chang, L. Diversifying top-k results. *arXiv preprint arXiv:1208.0076* (2012).

[72] Rachkovskij, D. Distance-based index structures for fast similarity search. In *Cybernetics and Systems Analysis* (2017), vol. 53, Kluwer Academic Publishers Norwell, p. 636–658.

[73] Radlinski, F., Dumais, S. Improving personalized web search using result diversification. In *International Conference on Research and Development in Information Retrieval* (2006), ACM, p. 691–692.

[74] Resende, M., Ribeiro, C. *Optimization by GRASP: Greedy Randomized Adaptive Search Procedures*. Springer, 2016.

[75] Ruiz, E. An algorithm for finding nearest neighbours in (approximately) constant average time. In *Pattern Recognition Letters* (1986), vol. 4, Elsevier, p. 145–157.

[76] Santos, L., Blanco, G., Oliveira, D., Traina, A., Traina Jr, C., Bedo, M. Exploring Diversified Similarity with Kundaha. In *International Conference on Information and Knowledge Management* (2018), ACM, p. 1903–1906.

[77] Santos, L., Carvalho, L., Bedo, M., Traina, A., Traina Jr, C. When similarity is not enough, ask for diversity: Grouping elements based on influence. In *International Symposium on Multimedia* (2016), IEEE, p. 26–29.

[78] Santos, L., Dias, R., Ferreira, M., Ribeiro, M., Traina, A., Traina Jr, C. Have you met viks?: A novel framework for visual diversity search analysis. In *Demos do Simpósio Brasileiro de Bancos de Dados* (2014), SBC, p. 209–214.

[79] Santos, L., Oliveira, W., Ferreira, M., Traina, A., Traina Jr, C. Parameter-free and domain-independent similarity search with diversity. In *International Conference on Scientific and Statistical Database Management* (2013), ACM, p. 1–12.

[80] Santos, R., Peng, J., Macdonald, C., Ounis, I. Explicit search result diversification through sub-queries. In *European Conference on Information Retrieval* (2010), Springer, p. 87–99.

[81] Semaan, G., Brito, J. A., Coelho, I., Silva, E., Fadel, A., Ochi, L., Maculan, N. A brief history of heuristics: from bounded rationality to intractability. In *Latin America Transactions* (2020), vol. 18, IEEE, p. 1975–1986.

[82] Skopal, T., Pokornỳ, J., Snasel, V. Pm-tree: Pivoting metric tree for similarity search in multimedia databases. In *Advances in Databases and Information Systems* (2004), Springer.

[83] Stewart, J. *Calculus: Concepts and contexts*. Cengage Learning, 2009.

[84] Stoyanovich, J., Amer-Yahia, S., Milo, T. Making interval-based clustering rank-aware. In *International Conference on Extending Database Technology* (2011), p. 437–448.

[85] Tao, Y., Zhang, J., Papadias, D., Mamoulis, N. An efficient cost model for optimization of nearest neighbor search in low and medium dimensional spaces. In *Transactions on Knowledge and Data Engineering* (2004), vol. 16, IEEE, p. 1169–1184.

[86] Tasan, M., Ozsoyoglu, Z. Improvements in distance-based indexing. In *International Conference on Scientific and Statistical Database Management* (2004), IEEE, p. 161–170.

[87] Thordsen, E., Schubert, E. Abid: Angle based intrinsic dimensionality. In *International Conference on Similarity Search and Applications* (2020), Springer, p. 218–232.

[88] Traina Jr., C., Santos, R., Traina, A., Vieira, M., Faloutsos, C. The Omni-family of all-purpose access methods: A simple and effective way to make similarity search more efficient. In *International Conference on Very Large Data Bases* (2007), vol. 16, Springer, p. 483–505.

[89] Turner, C., Fuggetta, A., Lavazza, L., Wolf, A. A conceptual basis for feature engineering. In *Journal of Systems and Software* (1999), vol. 49, Elsevier, p. 3–15.

[90] Van Leuken, R., Garcia, L., Olivares, X., van Zwol, R. Visual diversification of image search results. In *International Conference on World Wide Web* (2009), ACM, p. 341–350.

[91] Vieira, M., Razente, H., Barioni, M., Hadjieleftheriou, M., Srivastava, D., Traina Jr., C., Tsotras, V. On query result diversification. In *International Conference on Data Engineering* (2011), IEEE, p. 1163–1174.

[92] Vieira, M., Traina Jr., C., Traina, A., Arantes, A., Faloutsos, C. Boosting k-nearest neighbor queries estimating suitable query radii. In *International Conference on Scientific and Statistical Database Management* (2007), p. 10–10.

[93] Volnyansky, I., Pestov, V. Curse of dimensionality in pivot based indexes. In *International Conference on Similarity Search and Applications* (2009), IEEE, p. 39–46.

[94] Wang, Y., Meliou, A., Miklau, G. Rc-index: Diversifying answers to range queries. In *International Conference on Very Large Data Bases* (2018), vol. 11, ACM, p. 773–786.

[95] Xia, L., Xu, J., Lan, Y., Guo, J., Cheng, X. Learning maximal marginal relevance model via directly optimizing diversity evaluation measures. In *International Conference on Research and Development in Information Retrieval* (2015), ACM, p. 113–122.

[96] Xia, L., Xu, J., Lan, Y., Guo, J., Cheng, X. Modeling document novelty with neural tensor network for search result diversification. In *International Conference on Research and Development in Information Retrieval* (2016), ACM, p. 395–404.

[97] Yianilos, P. Data structures and algorithms for nearest neighbor search in general metric spaces. In *Symposium on Discrete Algorithms* (1993), ACM-SIAM, p. 311–321.

[98] Yianilos, P. Excluded middle vantage point forests for nearest neighbor search. Relatório Técnico, NEC Research Institute, 1998.

[99] Yu, C., Lakshmanan, L., Amer-Yahia, S. It takes variety to make a world: Diversification in recommender systems. In *International Conference on Extending Database Technology: Advances in Database Technology* (2009), ACM, p. 368–378.

[100] Yu, H., Ren, F. Search result diversification via filling up multiple knapsacks. In *International Conference on Information and Knowledge Management* (2014), ACM, p. 609–618.

[101] Yue, Y., Joachims, T. Predicting diverse subsets using structural svms. In *International Conference on Machine learning* (2008), p. 1224–1231.

[102] Zezula, P., Amato, G., Dohnal, V., Batko, M. *Similarity Search: The Metric Space Approach*, 1st ed. Springer, 2010.

[103] Zhang, M., Wang, H., Li, J., Gao, H. Diversification on big data in query processing. In *Clinical Orthopaedics and Related Research* (2018).

[104] Zheng, K., Wang, H., Qi, Z., Li, J., Gao, H. A survey of query result diversification. In *Knowledge and Information Systems* (2017), vol. 51, Springer, p. 1–36.

[105] Zhu, X., Goldberg, A., Van Gael, J., Andrzejewski, D. Improving diversity in ranking using absorbing random walks. In *Human Language Technologies* (2007), ACM, p. 97–104.

[106] Zhu, Y., Lan, Y., Guo, J., Cheng, X., Niu, S. Learning for search result diversification. In *International Conference on Research and Development in Information Retrieval* (2014), ACM, p. 293–302.

[107] Ziegler, C., McNee, S., Konstan, J., Lausen, G. Improving recommendation lists through topic diversification. In *International Conference on World Wide Web* (2005), ACM, p. 22–32.