Universidade Estadual de Feira de Santana

Programa de Pós-Graduação em Computação Aplicada

Consulta Espacial Preferencial por Palavra-chave

João Paulo Dias de Almeida

Feira de Santana

2016

Universidade Estadual de Feira de Santana

Programa de Pós-Graduação em Computação Aplicada

João Paulo Dias de Almeida

Consulta Espacial Preferencial por Palavra-chave

Dissertação apresentada à Universidade Estadual de Feira de Santana como parte dos requisitos para a obtenção do título de Mestre em Computação Aplicada.

Orientador: João B. Rocha-Junior

Feira de Santana

2016

Está página deverá ser substituída por uma folha contendo as assinaturas dos membros da banca.

Está página deverá ser substituída por uma folha contendo a ficha catalográfica.

Abstract

With the popularity of devices that are able to annotate data with spatial information (latitude and longitude), the processing of spatial queries has received a lot of attention from the research community recently. In this dissertation, we study a new query type named Top-k Spatial Keyword Preference Query that selects objects of interest based on the textual relevance of other spatio-textual objects in their spatial neighborhood. This work introduces this new query type, presents three algorithms for processing the query efficiently and performs an experimental evaluation using real databases to study the performance of the proposed algorithms.

Keywords: query processing, spatial databases, hybrid indexes, preference queries, information systems, information retrieval

Resumo

Com a popularidade de dispositivos capazes de anotar dados com coordenadas espaciais (latitude e longitude), o processamento de consultas espaciais tem recebido bastante atenção da comunidade científica recentemente. Esta dissertação apresenta uma nova consulta, chamada Consulta Espacial Preferencial por Palavra-chave, que seleciona objetos de interesse de acordo com a relevância textual de outros objetos espaço-textuais presentes na sua vizinhança espacial. Este trabalho introduz esta nova consulta, apresenta três algoritmos para processá-la de forma eficiente e avalia o desempenho dos algoritmos propostos através de um estudo experimental, utilizando bases de dados reais.

Palavras-chave: processamento de consultas, bases de dados espaciais, índices híbridos, consultas preferenciais, sistemas de informação, recuperação de informação

Prefácio

Esta dissertação de mestrado foi submetida a Universidade Estadual de Feira de Santana (UEFS) como requisito parcial para obtenção do grau de Mestre em Computação Aplicada.

A dissertação foi desenvolvida dentro do Programa de Pós-Graduação em Computação Aplicada (PGCA) tendo como orientador o Dr. **João B. Rocha-Junior**.

Esta pesquisa foi financiada pela Capes[1] e pela FAPESB[2].

[1]Programa de Demanda Social (DS): http://www.capes.gov.br/bolsas/bolsas-no-pais/ds-e-proap

[2]Financiamento de equipamentos do projeto, como computadores e notebooks

Agradecimentos

Em primeiro lugar, agradeço a minha mãe Eliete Dias de Almeida pelo apoio e incentivo que foram fundamentais para a conclusão deste trabalho. Você é um exemplo de luta e determinação no qual eu me inspiro todos os dias para alcançar meus objetivos.

Agradeço a Andreza Camilo, pelo companheirismo, carinho e dedicação direcionados a mim durante todo o período da confecção deste trabalho, seu apoio foi renovador e inspirador em vários momentos difíceis. Obrigado por ser essa pessoa especial que trouxe mais alegria aos meus dias.

Ao meu orientador, Prof. João B. Rocha-Junior, agradeço pela paciência e compreensão. Agradeço pelas aulas, reuniões e todo conhecimento que me foi passado. Sem o seu auxílio, este trabalho não seria possível.

Agradeço a todos do grupo de pesquisa ADaM (*Advanced Data Management Research Group*) pelo desenvolvimento do programa *Travel Assistant*, necessário para a organização das bases de dados, e por formarem um espaço de trabalho amigável e enriquecedor.

À minha família e amigos, os meus sinceros agradecimentos pelo carinho e atenção dedicados todo esse tempo.

Por fim, agradeço a CAPES pela bolsa de apoio à pós-graduação. Este apoio foi fundamental para que eu tenha sido capaz de me dedicar exclusivamente a este projeto durante 2 anos.

Sumário

Lista de Publicações

- Consulta Espacial Preferencial por Palavra-chave. João Paulo Dias de Almeida, João B. Rocha-Junior. (2014). *In WPOS/ERBASE*.

- Consulta Espacial Preferencial por Palavra-chave. João Paulo Dias de Almeida, João B. Rocha-Junior. (2015). *In SBBD*.

Lista de Tabelas

Lista de Figuras

Lista de Abreviações

Abreviação	**Descrição**
EPPC	Consulta Espacial Preferencial por Palavra-chave
IF	Arquivo Invertido (*Inverted File*)
S2I	*Spatial Inverted Index*
MBR	*Minimum Bounding Rectangle*
aR-Tree	*Aggregated* R-Tree
rng	Seleção Espacial (*range*)
inf	Influência
nn	Vizinho mais próximo
SK	*Top-k Spatial Keyword Query*
IFA	*Inverted File Based Algorithm*
SIA	*Spatially Indexed Algorithm*
SIA^+	*Optimized Spatially Indexed Algorithm*
$q.l$	Localização da consulta
$Q.D$	Conjunto de palavras-chave de uma consulta Q
$Q.k$	Quantidade de objetos a serem retornados pela consulta Q
$f.D$	Descrição textual de um objeto de referência f
$dist(x, y)$	Distância entre o objeto x e y
F	Conjunto de objetos de referência
P	Conjunto de objetos de interesse

Lista de Símbolos

Símbolo	Descrição
ψ	Definição de vizinhança espacial de interesse
τ^{ψ}	Escore do objeto de interesse de acordo com a vizinhança espacial
θ	Função de cálculo da relevância textual entre dois conjuntos de caracteres

Capítulo 1

Introdução

"O sucesso é ir de fracasso em fracasso sem perder o entusiasmo."
– Winston Churchill

Um dado espacial é a informação geográfica referente a um objeto físico. Esta informação pode ser representada utilizando valores numéricos em um sistema de coordenadas geográficas, como por exemplo, latitude e longitude. Um dado espacial pode conter outras informações além da informação geográfica que representam o objeto físico, a exemplo de informações textuais que representem características do objeto.

O volume de dados espaciais tem crescido significativamente nos últimos anos, o que explica o interesse por novas técnicas capazes de processar esses dados de forma eficiente [Cao et al. 2012]. Facebook[1], Google Maps[2], Twitter[3] e Waze[4] são exemplos de aplicações que processam informações espaciais para fornecer serviços úteis para o usuário.

A maioria desses dados espaciais estão associados a uma informação textual. Por exemplo, algumas mensagens do Twitter (texto) enviados a partir de *smartphones* possuem uma coordenada espacial (latitude e longitude). Uma grande parcela dos mais de 4 bilhões de objetos espaciais armazenados no OpenStreetMap (www.osm.org) estão associados a um texto descritivo. Estes objetos que possuem informações espaciais e textuais são comumente referenciados como objetos espaço-textuais [Vaid et al. 2005].

[1] www.facebook.com
[2] www.google.com.br/maps/
[3] www.twitter.com/
[4] www.waze.com

Para processar consultas de forma eficiente em grandes bases de dados compostas por objetos espaço-textuais, faz-se necessário utilizar índices híbridos [Chen et al. 2013, Cong et al. 2009, Rocha-Junior et al. 2011]. Estes índices combinam métodos de acesso espaciais, como R*-tree [Beckmann et al. 1990]; e textuais, como os Arquivos Invertidos [Zobel e Moffat 2006], para processar consultas de forma eficiente.

Nesta pesquisa é apresentada uma nova consulta para extrair informações de bases de dados espaço-textuais, a consulta Espacial Preferencial por Palavra-chave (EPPC). Diferente da consulta espaço-textual tradicional [Cong et al. 2009, Rocha-Junior et al. 2011] que retorna objetos próximos a uma dada localização; a consulta EPPC retorna objetos de interesse com base em outros objetos espaço-textuais presentes em sua vizinhança espacial.

Este capítulo é organizado da seguinte forma: a Seção 1.1 contém a motivação desta pesquisa, seguida pela Seção 1.3 que apresenta as questões de pesquisa. A Seção 1.2 descreve possíveis aplicações para a consulta proposta, enquanto a Seção 1.4 expõe o método de pesquisa utilizado. Por fim, a Seção 1.5 lista os artigos publicados durante o mestrado e a Seção 1.6 faz um esboço da estrutura desta dissertação.

1.1 Motivação

Uma das principais consultas utilizadas para extrair informações de bases de dados espaço-textuais é a consulta Espacial Preferencial Tradicional proposta por Yiu et al [Rocha-Junior et al. 2010, Yiu et al. 2007]. Dado um conjunto D de objetos de interesse para o usuário, a consulta Espacial Preferencial Tradicional retorna os k objetos existentes em D com os maiores escores. O escore de um objeto de interesse é definido pela qualidade dos objetos de referência (ex: cafés, restaurantes, hospitais) existentes na sua vizinhança espacial. A qualidade de um objeto de referência (*feature*) é dada a priori, por exemplo, através de uma agência de *rating*.

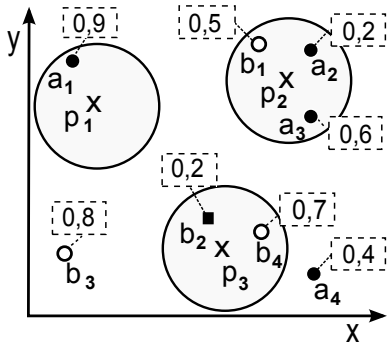


Figura 1.1: Objetos de interesse (p) e objetos de referência (a e b) com um escore associado.

A Figura 1.1 apresenta um espaço contendo objetos espaciais de interesse p (ex: hotéis) e objetos espaciais de referência a (ex: cafés) e b (ex: bares). Os objetos

de referência estão associados a um escore pré-definido. O círculo em volta de cada objeto de interesse representa o critério de vizinhança espacial. Assim, um usuário interessado em um hotel próximo a um bom café, terá como retorno os objetos p_1 e p_2, sendo p_1 melhor que p_2, porque o objeto de referência a_1 presente na vizinhança espacial de p_1 tem um escore maior que a_3 (melhor escore entre os restaurantes na vizinhança de $p2$).

Na consulta Espacial Preferencial Tradicional, a escolha dos objetos espaciais de referência é feita a partir de uma lista pré-definida de opções. Sendo assim, um usuário que deseja buscar por um hotel (objeto de interesse) próximo a um restaurante (objeto de referência), precisa selecionar a opção "restaurante" nesta lista pré-definida. Desta forma, o usuário é limitado pelas opções existentes na lista, não sendo capaz de selecionar outras opções, ou escolher um restaurante específico.

Exemplo. Dada uma lista de opções pré-definida $O = \{restaurante, bar, hospital\}$*, o usuário deseja encontrar um hotel próximo a um restaurante que sirva massas ou carnes. Como não existe esta opção na lista* O*, o usuário seleciona "restaurante", e a consulta Espacial Preferencial Tradicional retorna hotéis que possuem qualquer tipo de restaurante em sua vizinhança espacial.*

Para processar uma consulta Espacial Preferencial Tradicional é necessário que cada objeto espacial de referência possua uma categoria (ex: restaurante). Estas categorias são pré-definidas antes da consulta ser executada. Isto impede que esta consulta seja executada em bases espaço-textuais como o *Twitter*, visto que os *tweets* não possuem uma categoria pré-estabelecida.

Visando remover essas limitações e fornecer mais liberdade ao usuário, propõe-se nesta pesquisa uma nova consulta denominada Consulta Espacial Preferencial por Palavra-chave (EPPC). Nesta nova consulta, o usuário é capaz de descrever os objetos de referência desejados a partir de um conjunto de palavras-chave, formado por qualquer quantidade de termos que o usuário desejar. Para isto, é necessário que o objeto de referência possua uma descrição textual, ou seja, que o objeto de referência seja um objeto espaço-textual de referência.

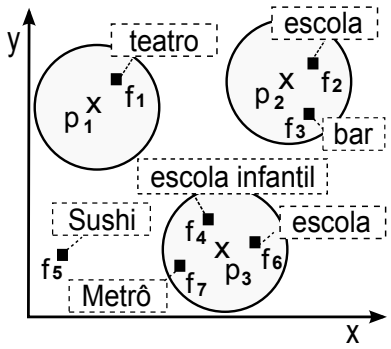


Figura 1.2: Objetos (p) e objetos de referência (f) com um texto associado.

Exemplo. A Figura 1.2 apresenta um espaço contendo objetos espaciais de interesse

p (ex: apartamentos) e objetos espaço-textuais de referência f (ex: estabelecimentos diversos). O círculo em volta de cada objeto de interesse representa o critério de vizinhança espacial. Assim, um usuário interessado em alugar um apartamento próximo a um objeto relevante para as palavras-chave "escola" e "infantil", teria como retorno os objetos p_2 e p_3, sendo p_3 o objeto mais relevante (top-1), pois ele possui na sua vizinhança espacial, um objeto (f_4) mais relevante para as palavras-chave de busca que p_2.

Entretanto, processar uma consulta Espacial Preferencial Tradicional é custoso [Rocha-Junior et al. 2010, Yiu et al. 2007]. A nova consulta proposta (EPPC) possui um custo de processamento ainda maior do que a consulta Espacial Preferencial Tradicional, por permitir que o usuário descreva quais são os objetos de referência desejados a partir de um conjunto de palavras-chave.

Sabe-se que o processamento eficiente de consultas Espaciais Preferenciais, e consequentemente o curto tempo de resposta, é crucial para conquistar a aceitação dos usuários [Balke et al. 2002, Kießling e Köstler 2002, Rocha-Junior 2012]. Sistemas *Web* são comumente utilizados por usuários que não possuem um conhecimento preciso do conteúdo da base de dados [Georgiadis et al. 2008]. Como estas bases geralmente são muito grandes, é comum que o usuário tente várias consultas até encontrar o resultado esperado. O usuário aprende a interagir com a aplicação a partir dos resultados obtidos de várias consultas [Rocha-Junior 2012]. Desta forma, o processamento eficiente da consulta permite aumentar a satisfação do usuário com o sistema, uma vez que permite o usuário realizar mais consultas à base de dados, no mesmo período de tempo.

Uma forma trivial para processar a consulta EPPC requer selecionar todos os objetos espaço-textuais presentes na vizinhança espacial de cada objeto de interesse, computar a relevância textual de cada um destes objetos para as palavras-chave de busca, atribuir o escore ao objeto de interesse, e repetir este processo para todos os objetos de interesse até obter os k melhores. Em bases de dados grandes, este método é bastante custoso. As principais contribuições deste trabalho são: 1) propor a consulta EPPC; 2) apresentar algoritmos para processar esta consulta de forma eficiente e 3) avaliar os algoritmos propostos através de experimentos em bases de dados reais.

1.2 Aplicações

Várias aplicações podem se beneficiar da consulta EPPC para prover serviços aos usuários. A seguir são apresentados alguns exemplos.

Segurança pública

Uma fração substancial das consultas realizadas no Google e no Bing tem o propósito de encontrar locais, ou descrições textuais de localidades [Cao et al. 2012]. Portanto, os usuários estão acostumados a utilizar consultas textuais para obter informações relacionadas a objetos espaciais. Muitos aplicativos populares atualmente podem ser utilizados em conjunto com a consulta Espacial Preferencial por Palavra-chave, como por exemplo, o *Twitter*.

Muitas das mensagens do *Twitter* (*tweets*), enviadas de *smartphones* que possuem um GPS (*Global Posiotining System*) integrado, possuem uma localização espacial (latitude e longitude) e texto (existente no corpo do *tweet*). Uma base de dados formada por *tweets* pode ser utilizada por um aplicativo de segurança pública para localizar ruas, ou áreas públicas, cuja vizinhança espacial possui algum *tweet* contendo os termos 'violência' e 'assalto' em seu conteúdo.

Desta forma, é possível utilizar o *tweet* como indicador da existência de criminalidade naquela rua, ou área pública, permitindo que o poder público se organize, e planeje as medidas de segurança necessárias. A seguir é detalhado cada um dos objetos da consulta para esta aplicação e é descrito um exemplo.

Objetos de interesse. Nesta aplicação, o conjunto de objetos de interesse é formado pelas ruas de uma cidade.

Objetos espaço-textuais de referência. O conjunto de objetos espaço-textuais de referência é formado por *tweets*.

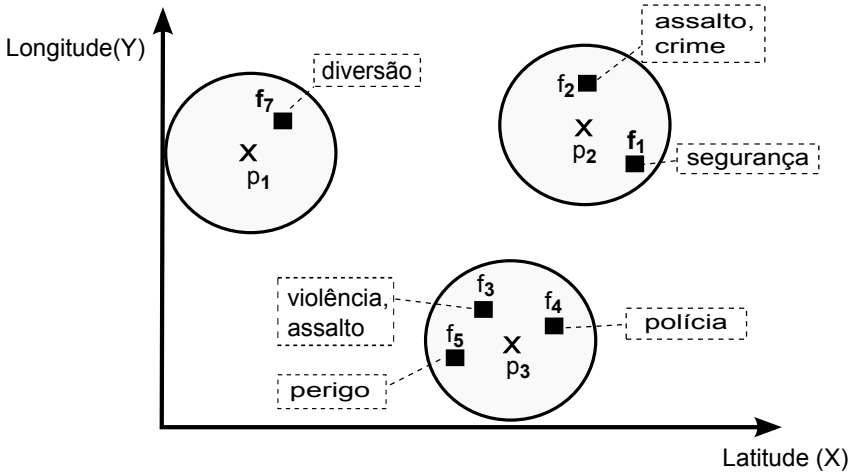


Figura 1.3: Objetos (p) e objetos de referência (f) com um texto associado.

Exemplo. A Figura 1.3 apresenta um espaço contendo objetos espaciais de interesse p (ruas[5]) e objetos espaço-textuais de referência f (tweets). O círculo em volta de cada objeto de interesse representa o critério de vizinhança espacial. Assim, um

[5]Neste caso, as ruas são representadas por um ponto no espaço.

aplicativo de segurança pública pode indicar ruas que possuem algum tweet relevante para as palavras-chave "violência" e "assalto" em suas vizinhanças espaciais. Neste contexto, a consulta retorna os objetos p_2 e p_3, sendo p_3 o objeto mais relevante (top-1), visto que ele possui na sua vizinhança espacial, um objeto (f_3) mais relevante para as palavras-chave de busca que p_2.

Turismo

O turismo se tornou um empreendimento altamente competitivo por todo o mundo, e a sua competitividade é impulsionada cada vez mais pelo avanço dos Sistemas de Informação [Yueh et al. 2007]. Atualmente, a Internet é a fonte principal de informação para turistas que desejam informações sobre o seu destino [Deri 2015, Yueh et al. 2007].

Suponha um turista visitando Paris pela primeira vez. Este turista não conhece a cidade, porém possui interesse em conhecer os museus existentes nela. Sendo assim, um aplicativo *Web* pode utilizar a consulta EPPC para retornar as k estações de metrô próximas a estabelecimentos relevantes para um conjunto de palavras-chave que definem o interesse do usuário (ex: "museu de arte"), facilitando a locomoção do usuário pela cidade. A seguir é detalhado cada um dos objetos da consulta para esta aplicação e é descrito um exemplo.

Objetos de interesse. O conjunto de objetos de interesse é formado pelas estações de metrô.

Objetos espaço-textuais de referência. O conjunto de objetos espaço-textuais de referência é formado por estabelecimentos diversos que possuem uma localização espacial e uma descrição textual.

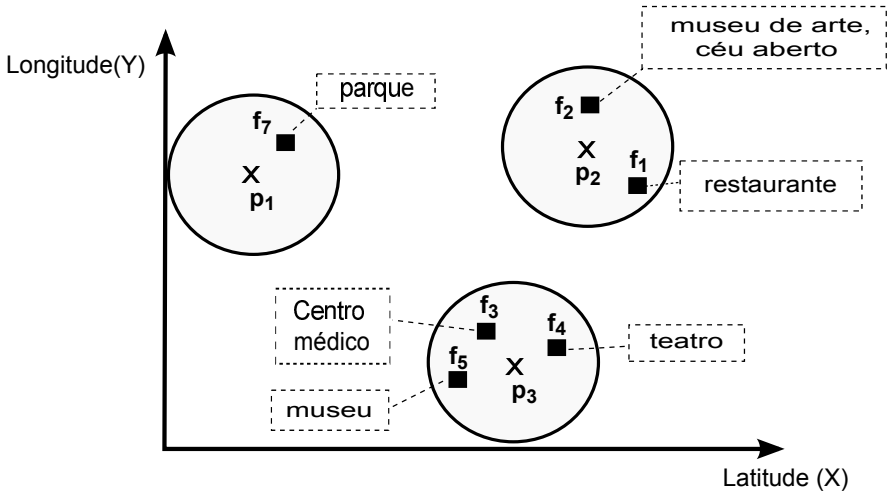


Figura 1.4: Objetos (p) e objetos de referência (f) com um texto associado.

Exemplo. A Figura 1.4 apresenta um espaço contendo objetos espaciais de interesse p (metrôs) e objetos espaço-textuais de referência f (estabelecimentos diversos). Um aplicativo Web pode indicar uma estação de metrô próxima a um objeto de referência relevante para as palavras-chave "museu de arte". Sendo assim, a consulta retorna os objetos p_2 e p_3, sendo p_2 o objeto mais relevante (top-1), visto que ele possui na sua vizinhança espacial, um objeto (f_2) mais relevante para as palavras-chave de busca que p_3.

Observatório astronômico

Quando astrônomos analisam imagens de telescópio, eles verificam se o objeto recém observado está listado em um catálogo de objetos conhecidos, como o *Guide Star Catalog II* que contém quase um bilhão de objetos [Lasker et al. 2008]. Devido a atmosfera e a distorções óticas, a posição de um objeto celeste em uma imagem de telescópio pode variar levemente entre várias observações do mesmo objeto. Assim, utilizar a posição do objeto (dada pela imagem do telescópio) para verificar se o objeto existe no catálogo de objetos conhecidos é inadequado [Fu et al. 2012].

Considerando uma base formada por imagens de um telescópio (objetos espaço-textuais de referência), onde cada imagem possui a posição do corpo celeste observado e informações textuais referentes a observação (ex: nome do observador, características do corpo celeste, informações da atmosfera durante a observação), e uma outra base formada pelo catálogo de objetos celestes conhecidos (objetos de interesse). A consulta EPPC pode ser utilizada para identificar quais objetos do catalogo foram observados por um astrônomo específico, ou encontrar objetos que possuem uma determinada característica visível pelo telescópio, por exemplo.

A seguir é detalhado cada um dos objetos da consulta para esta aplicação.

Objetos de interesse. Nesta aplicação, o conjunto de objetos de interesse é formado por todos os objetos existentes no catalogo de objetos conhecidos. Cada objeto possui a sua respectiva localização espacial.

Objetos espaço-textuais de referência. O conjunto de objetos espaço-textuais de referência é formado por uma base de dados que contém todas as imagens geradas por um telescópio. Cada imagem esta associada às coordenadas espaciais do corpo celeste observado e a várias informações textuais, como o nome do observador, características do corpo celeste, e informações da atmosfera durante a observação.

Sendo assim, um usuário pode utilizar o conjunto de palavras-chave da consulta EPPC para identificar características dos objetos de interesse. Segue o exemplo da Figura.

Exemplo. Suponha um astrônomo que deseje encontrar objetos no catalogo de objetos conhecidos que possuam a cor verde. Dada a Figura 1.5, que representa uma área espacial composta por objetos de interesse e objetos de referência, ao inserir o termo "verde" no conjunto de palavras-chave da consulta EPPC, é realizada uma busca em todas as imagens obtidas do telescópio (objetos espaço-textuais de referência) para encontrar as imagens próximas a um objeto de interesse que possuem o termo "verde" em sua descrição textual. Desta forma, a consulta retorna o objeto p_2 como o objeto mais relevante (top-1), pois ele é o único que possui na sua vizinhança espacial um objeto (f_2) relevante para a palavra-chave de busca.

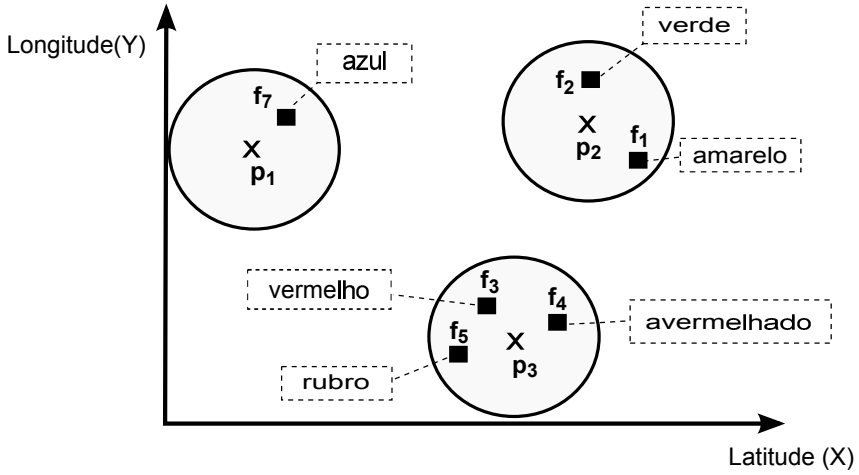


Figura 1.5: Objetos (p) e objetos de referência (f) com um texto associado.

1.3 Questões de pesquisa

A questão de pesquisa geral desta dissertação é – Como processar esta nova consulta espaço-textual (EPPC)? Esta questão conduz a questões mais específicas. As questões de pesquisa abordadas por esta dissertação são apresentadas a seguir:

Q 1. Como processar a consulta EPPC de forma eficiente?

Q 2. É possível utilizar índices híbridos para melhorar o desempenho da consulta EPPC?

Q 3. É possível desenvolver novas técnicas, além dos índices híbridos, para otimizar ainda mais o desempenho da consulta EPPC?

1.4 Método de Pesquisa

Nesta seção são descritos os passos utilizados para realizar esta pesquisa; são eles o levantamento bibliográfico, formalização da consulta, publicação dos resultados preliminares em congressos, implementação de técnicas avançadas para o processamento da consulta, levantamento de bases de dados para validação da consulta, realização de experimentos para avaliar os algoritmos propostos para processar a consulta, e por fim, a escrita da dissertação.

Durante o levantamento bibliográfico foi realizada a seleção de artigos relacionados ao tema. Entre estes artigos, foram selecionados artigos clássicos e artigos com forte base teórica no tema pesquisado. Por fim, elaborou-se um resumo sistematizando as ideias principais dos artigos levantados e verificou-se a existência de patentes ou registro de direitos autorais relacionados ao tema de pesquisa.

A partir dos artigos selecionados durante a revisão bibliográfica, encontrou-se técnicas que processam consultas similares à consulta proposta neste trabalho. Inspirando-se nas técnicas encontradas, chegou-se a três algoritmos para processar a consulta proposta.

Para obter um *feedback* da comunidade científica, foram submetidos artigos em congressos da área. Foi submetido um artigo para o *Workshop* de Trabalhos de Pós-Graduação (WPOS) no ERBASE 2014 e outro para o Simpósio Brasileiro de Banco de Dados (SBBD) 2015. O artigo enviado ao SBBD 2015 foi convidado para ser enviado ao *Journal of Information and Data Management* (JIDM).

A consulta foi avaliada utilizando bases de dados reais, como por exemplo, o OpenStreetMap[6] (OSM). Destas bases de dados são extraídos os objetos de interesse e os objetos de referência utilizados na consulta EPPC. Um pequeno programa é responsável pela extração dos objetos espaciais do arquivo fornecido pela base de dados.

Nesta dissertação é empregado um método de pesquisa quantitativo, baseado em avaliações experimentais. Em cada experimento, é selecionada uma variável independente e é realizado um estudo sobre o efeito desta variável nas variáveis dependentes. Para responder às questões de pesquisa, foram conduzidos experimentos, descritos no Capítulo 5. O ciclo de vida da pesquisa é listado a seguir:

- **Definir problema.** Um novo problema é definido após identificar fragilidades nas abordagens atuais, ou identificando oportunidades para melhorias.
- **Propor solução.** Uma nova consulta e novos algoritmos são propostos para resolver o problema.
- **Avaliar solução.** As soluções propostas são comparadas com abordagens consolidadas na comunidade científica, ou com um *baseline* factível.
- **Divulgar tecnologia.** Os resultados obtidos são publicados em eventos ou simpósios.

A seguir é feita uma breve descrição das variáveis utilizadas na avaliação experimental e como as bases de dados são preparadas para os experimentos. As variáveis e as bases de dados utilizadas são descritas detalhadamente no Capítulo 5.

Variáveis. Durante a avaliação experimental, uma variável independente é selecionada para determinar o seu efeito sobre outras variáveis dependentes. As principais variáveis independentes utilizadas na avaliação experimental são 1) *cardinalidade*, ex: o número de objetos (tuplas) armazenadas na base de dados; 2) *número de resultados*, ex: o número (k) de resultados esperado pela consulta; 3) *tamanho do conjunto de palavras-chave*, ex: quantidade de palavras-chave utilizadas na consulta. O efeito de manipular variáveis independentes é estudado em variáveis dependentes como: *I/O* (ex: a quantidade de páginas de disco acessadas) e *tempo de resposta* (ex: a janela temporal entre realizar a consulta e obter o conjunto resposta).

[6] www.openstreetmap.org

Preparação da base de dados. Na avaliação experimental são empregadas bases de dados reais, provenientes de aplicativos reais. Todas as bases de dados são provenientes do OpenStreetMap[7], ou de terceiros que utilizam a base do OpenStreetMap para gerar bases menores.

1.5 Publicações

Nesta seção são listadas todas as publicações originadas desta pesquisa de mestrado, acompanhadas por uma breve descrição.

- Consulta Espacial Preferencial por Palavra-chave. João Paulo Dias de Almeida, João B. Rocha-Junior. *In WPOS/ERBASE*, Bahia, Feira de Santana, Maio 2014.

  Artigo apresentado no *Workshop* de pós-graduação da Escola Regional de Computação Bahia-Alagoas-Sergipe (ERBASE). Neste artigo foi apresentado apenas a definição da consulta Espacial Preferencial por palavra-chave e algumas informações relacionadas a proposta da pesquisa de mestrado. Durante o WPOS, o artigo foi analisado por uma banca de doutores que ofereceram sugestões para melhorias do trabalho.

- Consulta Espacial Preferencial por Palavra-chave. João Paulo Dias de Almeida, João B. Rocha-Junior. *In Simpósio Brasileiro de Banco de Dados (SBBD)*, Rio de Janeiro, Petrópolis, Outubro 2015.

  O artigo apresentado no SBBD apresenta uma especificação da consulta EPPC, três algoritmos para processar a consulta de forma eficiente, e uma discussão abrangendo os experimentos realizados na consulta utilizando os algoritmos propostos. Foram conduzidos experimentos para analisar o tempo de resposta e a quantidade de páginas lida pela consulta. Neste artigo, todos os experimentos e algoritmos foram apresentados utilizando apenas o critério de vizinhança seleção espacial (*range*), não apresentando os algoritmos para o critério de vizinhança vizinho mais próximo e influência.

1.6 Esboço da Dissertação

Nesta seção é descrita a organização desta dissertação.

Capítulo 1: Introdução. Este capítulo oferece a introdução desta dissertação. Também estão incluídos neste capítulo a motivação, as questões de pesquisa, as possíveis aplicações deste trabalho e as publicações realizadas.

[7] http://www.openstreetmap.org/

Capítulo 2: Fundamentação Teórica. Neste capítulo são abordados os temas indispensáveis para o bom entendimento da pesquisa apresentada nesta dissertação.

Capítulo 3: Preliminares. Neste capítulo é feita a definição da consulta Espacial Preferencial por Palavra-chave e é apresentado o Arquivo Invertido Adaptado para processar a consulta EPPC.

Capítulo 4: Algoritmos. Este capítulo descreve em detalhes os algoritmos propostos para processar a consulta EPPC.

Capítulo 5: Avaliação Experimental. Neste capítulo é realizada a avaliação experimental dos algoritmos propostos para processar a consulta EPPC, discutindo vantagens e desvantagens de cada abordagem.

Capítulo 6: Consideração Finais. Este capítulo conclui a dissertação, apresentando as principais contribuições e os trabalhos futuros.

Capítulo 2

Fundamentação Teórica

"Todas as verdades são fáceis de entender uma vez que são descobertas; o ponto é descobri-las."
– Galileu Galilei

Neste capítulo são apresentados os conceitos fundamentais para compreender o contexto discutido nesta dissertação. Inicialmente são apresentadas as consultas Textuais. Posteriormente, são apresentados os Bancos de Dados Espaciais e as consultas Espaciais. Em seguida, é definido o conceito de consultas Preferenciais. Finalmente, o capítulo é encerrado com a apresentação das consultas Espaciais Preferenciais e das consultas Espaço-Textuais.

2.1 Consultas Textuais

Consulta textual é uma tecnologia chave para engenhos de busca [Zobel e Moffat 2006]. Em uma consulta textual (*text query*) o usuário pode digitar uma ou mais palavras-chave na consulta para descrever qual tipo de documento ele deseja obter [Manning et al. 2008]. Esta consulta é utilizada para localizar e recuperar informações de uma determinada coleção textual, retornando ao usuário os documentos relevantes para o conjunto de palavras-chave da consulta [Salminen e Tompa 1994]. Engenhos de busca *Web* e sistemas de busca em *desktop* são exemplos cotidianos de aplicações práticas das consultas Textuais.

Uma base de dados textual é uma coleção de dados textuais, como: páginas *Web*, enciclopédias, publicações acadêmicas, ou *e-mails*. Cada elemento de uma coleção de dados textual é denominado "Documento". Segundo Manning et al. [Manning et al. 2008], Documento é qualquer unidade sobre a qual se decida construir um Sistema de Recuperação de Informação. Em um Sistema de Recuperação

de Informação Textual típico, o usuário descreve o Documento que deseja obter através de um conjunto de palavras-chave (*bag of words*) [Zobel e Moffat 2006].

Exemplo. A partir da base de dados textual representada na Figura 2.1, um usuário pode identificar um documento de seu interesse através de consultas textuais. Neste caso, cada linha do texto é considerada um Documento. Portanto, quando um usuário submete um conjunto de palavras-chave em uma consulta textual, esta consulta retorna todos os documentos existentes na base de dados textual que são relevantes para as palavras-chave especificadas. Como exemplo, caso o usuário submeta a palavra-chave "big", a consulta textual retorna os Documentos 2 e 3 representados na Figura 2.1.

Para processar as consultas textuais de forma eficiente, é comum utilizar uma estrutura de dados denominada de Arquivo Invertido (*Inverted File* – IF) [Zobel e Moffat 2006]. Para criar um IF, é necessário extrair os termos de cada documento da base de dados textual. Para este fim, é realizada uma etapa de pré-processamento denominada *parsing*.

1 The old night keeper keeps the keep in the town
2 In the big old house in the big old gown.
3 The house in the town had the big old keep
4 Where the old night keeper never did sleep.
5 The night keeper keeps the keep in the night
6 And keeps in the dark and sleeps in the light.

Figura 2.1: Base de dados textual *Keeper*. Cada linha do texto representa um documento. Fonte: Zobel e Moffat [Zobel e Moffat 2006].

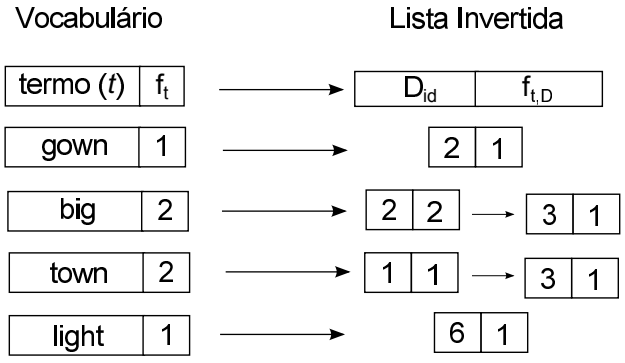


Figura 2.2: Exemplo de Arquivo Invertido utilizando os termos existentes na base textual *keeper*.

O *parsing* [Zobel e Moffat 2006] pode ser realizado em duas etapas: o *casefold* e a remoção das *stop words*. O *casefold* converte todas as letras contidas no documento, em letras minúsculas. Aplicando o *casefold* no Documento 1, obtém-se "*the old night keeper keeps the keep in the town*".

Em todo idioma existem palavras que aparacem constantemente nos textos produzidos para este idioma, ou palavras cuja função é apenas de identificar uma relação gramatical. Estas palavras são conhecidas como *stop words*. Removendo as *stop words* do Documento 1, e após ser aplicado o *casefold*, são obtidos os seguintes termos do Documento 1: "*old night keeper keeps keep town*" [Zobel e Moffat 2006]. Observa-se que a aplicação do *parsing* também reduz o tamanho do documento consideravelmente, facilitando o armazenamento e organização destes documentos.

O IF é formado por um vocabulário (também conhecido como dicionário de termos) e por um conjunto de listas invertidas (ou *postings list*). Cada termo t existente na coleção possui uma lista invertida correspondente. Em cada lista invertida existe um identificador (D_{id}) para cada documento (D[1]) que possui o termo t em sua descrição textual. Cada D_{id} é acompanhado de um valor que representa a frequência $f_{t,D}$ com que o termo t aparece na descrição textual deste Documento. O vocabulário pode armazenar a quantidade f_t de documentos existentes na coleção que possuem o termo t, e um ponteiro para a lista invertida do termo t correspondente [Zobel e Moffat 2006].

Exemplo. A Figura 2.2 representa parte de um Arquivo Invertido gerado a partir da base de dados textual keeper (Figura 2.1). Este IF possui os termos gown, big, town e light. Estão armazenados no vocabulário os termos, a quantidade de documentos que possuem estes respectivos termos e o ponteiro apontando para a lista invertida (representado pela seta unidirecional). Para cada documento que possui um termo t, a lista invertida armazena uma tupla composta pelo identificador deste documento (D_{id}) e a frequência ($f_{t,D}$) com que o termo t aparece em D.

Em um Sistema de Recuperação de Informação Textual que computa a relevância textual de um documento em relação a consulta, utiliza-se um *ranking* para identificar os possíveis documentos a serem retornados como resposta para o usuário. Para criar o *ranking*, é aplicado uma medida de similaridade ou heurística capaz de indicar a proximidade existente entre um documento e as palavras-chave existentes na consulta do usuário [Zobel e Moffat 2006].

A similaridade cosseno é amplamente utilizada pela comunidade da Recuperação de Informação para uma formulação efetiva da similaridade entre o documento e o conjunto de palavras-chave fornecido pelo usuário [Cohen et al. 2003, Zhu et al. 2011, Zobel e Moffat 2006]. Dada uma consulta textual T, composta por um conjunto de termos t ($t \in T$), a similaridade cosseno $\theta(T, D)$ calcula o ângulo cosseno, em um espaço n-dimensional, entre o vetor formado pelo peso dos termos t em T e o texto associado ao documento D. Sendo assim, um documento D é considerado como uma possível resposta a consulta apenas quando existir pelo menos um termo $t \in T$ que também esteja presente em D ($\exists t \in T : t \in D$).

A seguir são apresentadas algumas medidas utilizadas para calcular o cosseno entre um documento e uma consulta, como proposto por Zobel e Mofat

[1]Na base textual apresentada na Figura 2.1, cada linha de texto corresponde a um documento D e a descrição textual deste documento corresponde ao texto existente nesta linha

[Zobel e Moffat 2006]:

- a frequência $f_{t,D}$ do termo t na descrição textual de D
- a frequência $f_{t,T}$ do termo t na consulta T
- a quantidade f_t de documentos que contém o termo t
- o total N de documentos existentes na coleção

Existem muitas variações da formulação da similaridade cosseno [Manning et al. 2008, Rocha-Junior 2012]. Nesta dissertação é utilizada a formulação apresentada por Zobel e Mofat [Zobel e Moffat 2006], apresentada na Equação 2.1 que emprega as medidas descritas anteriormente.

$$\theta(T, D) = \frac{\sum_{t \in T} w_{t,D} \cdot w_{t,T}}{\sqrt{\sum_{t \in D} (w_{t,D})^2 \cdot \sum_{t \in T} (w_{t,T})^2}} \tag{2.1}$$

O peso do termo t no documento D ($w_{t,D}$) é computado por $w_{t,D} = 1 + ln f_{t,D}$, enquanto o peso $w_{t,T}$ do termo t na consulta Q é $w_{t,T} = ln\Big(1 + \frac{N}{f_t}\Big)$. Quanto maior o valor de $\theta(T, D)$, maior é a relevância textual entre o documento D e a consulta T. Por isto, $\theta(T, D)$ também é denominado como o escore textual de D em relação a consulta T.

Além disto, $w_{t,T}$ representa uma propriedade usualmente descrita como a frequência inversa do documento (*inverse document frequency* – IDF), enquanto $w_{t,D}$ captura a propriedade da frequência do termo (*term frequency* – TF). Devido a isto, a formulação descrita na Equação 2.1 é conhecida na literatura como TFxIDF [Zobel e Moffat 2006].

Ao processar a consulta textual pode ser criado um *ranking* com os documentos a serem apresentados ao usuário como resposta. A Figura 2.3 exemplifica o processamento de uma consulta textual. Neste exemplo, os termos são processados um a um. Inicialmente, cada documento possui um escore textual igual a zero. Para isto, um vetor acumulador A, de tamanho N, pode ser criado para armazenar o escore textual parcial de cada documento. Em cada posição A_D do vetor, é armazenado o escore textual parcial de um documento D.

Sendo assim, para cada termo $t \in T$ é calculado a contribuição $w_{t,D} \cdot w_{t,T}$ (Equação 2.1) do termo t para a similaridade entre o documento D e a consulta T. O valor obtido deste somatório é armazenado na posição A_D do vetor acumulador (representado pelo "Acumulador" na Figura 2.3).

O escore textual de cada documento D é calculado pela divisão do seu respectivo valor em A_D pelo peso dos documentos (W_D), $W_D = \sqrt{\sum_{t \in D} (w_{t,D})^2 \cdot \sum_{t \in T} (w_{t,T})^2}$ (Equação 2.1). Por fim, os documentos são ordenados pelos seus respectivos valores de escores textual e são apresentados ao usuário de forma ordenada.

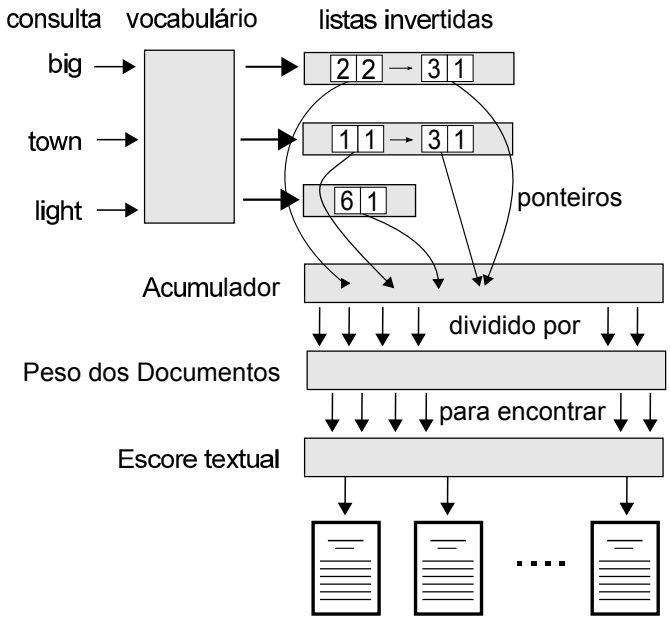


Figura 2.3: Exemplo de execução de uma consulta textual utilizando o Arquivo Invertido (IF). Fonte: Adaptado de Zobel e Mofat [Zobel e Moffat 2006].

2.2 Consultas Espaciais

Os bancos de dados precisaram se adaptar para armazenar e organizar diferentes tipos de dados de forma mais eficiente. O aumento na produção de dados espaciais em conjunto com o avanço da tecnologia, propiciou um contexto onde dados espaciais são o centro de muitas aplicações [Rigaux et al. 2001]. Atualmente, qualquer indivíduo com um *smartphone* é um potencial produtor de dados espaciais, devido a grande popularização do *Global Positioning System* (GPS).

Imagens de satélite, equipamentos médicos, *Geographic Information System* (GIS) são outras fontes que fornecem um grande volume de dados espaciais. Devido este grande volume, a tarefa de examinar estes dados espaciais detalhadamente é um processo caro e impraticável para usuários que não possuem ferramentas computacionais adequadas [Gao e Xia 2006].

Um sistema de banco de dados espacial oferece suporte para objetos espaciais, como pontos, linhas e polígonos [Güting 1994, Rigaux et al. 2001]. Este sistema fornece suporte adicional para a modelagem dos dados espaciais e para a descrição da consulta espacial. Índices espaciais são utilizados para processar as consultas espaciais eficientemente [Güting 1994].

Objetos espaciais. A Figura 2.4 apresenta alguns dos objetos espaciais básicos: pontos (objetos), linhas, e regiões (polígonos). Um ponto (Figura 2.4(a)) representa um objeto cuja a área não é relevante, apenas a sua localização espacial. Por exemplo, a localização de um objeto de referência (ex: restaurante) ou de uma pessoa

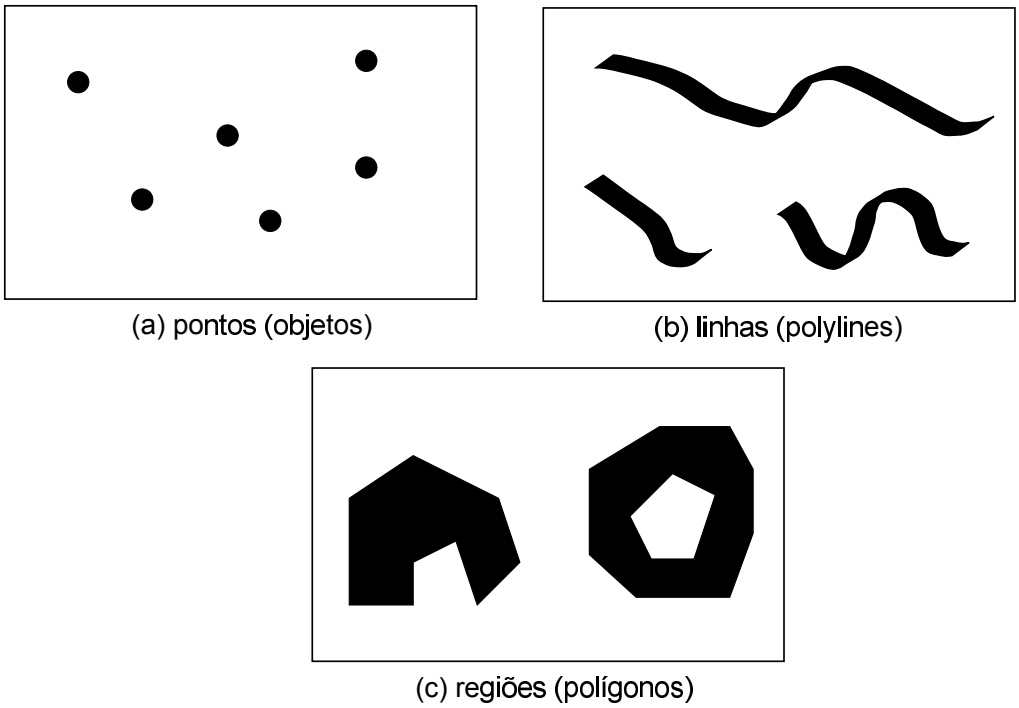


Figura 2.4: Objetos espaciais básicos. Fonte: Adaptado de Rocha-Junior [Rocha-Junior 2012].

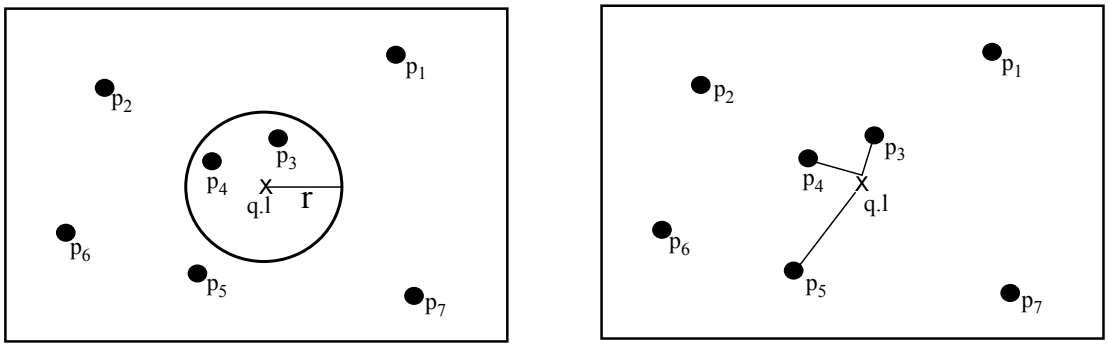
podem ser representados como um ponto. Uma linha (Figura 2.4(b)) pode ser utilizada para representar um rio, uma estrada, ou linhas de abastecimento elétrico. Linhas podem intersectar outras linhas. Por fim, uma região (Figura 2.4(c)) usualmente é modelada como um polígono e pode representar objetos espaciais cuja a área espacial é relevante, como uma fazenda, ou uma floresta. Regiões são disjuntas, entretanto, elas podem apresentar buracos ou podem ser compostos por vários pedaços disjuntos [Güting 1994, Rocha-Junior 2012].

Com o aumento do volume de dados espaciais disponíveis para busca, houve um aumento no uso de consultas Espaciais. Um dos tipos mais importantes de consultas espaciais suportadas por um banco de dados espacial são as seleções espaciais baseadas em predicados (*spatial selection*) [Güting 1994, Rocha-Junior 2012]. Dado um conjunto de dados, uma seleção espacial retorna o conjunto de objetos que satisfazem um predicado. Este predicado pode ser representado utilizando uma ou mais relações espaciais. As relações espaciais são as mais importantes operações oferecidas pela álgebra espacial [Güting 1994]. Entre estas relações, podem-se citar a relação espacial topológica (ex: adjacência, disjunção), a direcional (ex: acima, abaixo, a esquerda), e a métrica (ex: distância). Como exemplo de uma seleção espacial, pode-se citar "encontre todos os restaurantes em um raio de 100m a partir da minha localização atual"

Nesta dissertação, o foco é direcionado para duas *spatial selections* em particular: *range* e vizinho mais próximo.

Range. Dada a localização de uma consulta $q.l$ e uma distância $dist(p, q.l)$ (distância euclidiana entre $q.l$ e um objeto de interesse p), a consulta espacial *Range* retorna todos os p objetos de interesse cuja as distâncias são menores ou iguais a r, $dist(p, q.l) \leq r$ [Rocha-Junior 2012, Yiu et al. 2007]. Portanto, r define a vizinhança espacial da consulta $q.l$.

Vizinho mais próximo (k-NN). Dado um conjunto de objetos de interesse P ($p \in P$), a localização da consulta $q.l$ e um valor inteiro k, esta consulta espacial retorna os k objetos de interesse mais próximos espacialmente de $q.l$ (menor distância $dist(p, q.l)$ entre qualquer p e $q.l$, ou seja, $\forall p' : dist(p, q.l) < dist(p', q.l) | p, p' \in P, p \neq p'$) [Rocha-Junior 2012, Yiu et al. 2007].



(a) Range. (b) Vizinho mais próximo (3-NN).

Figura 2.5: Exemplos de *spatial selections*. Fonte: Rocha-Junior [Rocha-Junior 2012].

A Figura 2.5 apresenta exemplos da consulta *range* (a) e *Nearest Neighbor* (b). Na Figura 2.5(a), $q.l$ representa o local onde a consulta está sendo realizada, enquanto r representa o raio de interesse. Aplicando esta consulta na área espacial apresentada na Figura 2.5 (a), os pontos p_3 e p_4 são retornados como resposta. Na Figura 2.5(b), $q.l$ também representa o local da consulta. Nesta consulta, o usuário está interessado nos 3 objetos mais próximos ($3-NN$) da sua localização $q.l$. Sendo assim, os objetos p_3, p_4 e p_5 são retornados, nesta ordem.

2.2.1 Índices Espaciais

Um sistema de banco de dados espaciais necessita de um mecanismo que o ajude a retornar os objetos espaciais de acordo com as suas localizações no espaço de forma rápida e eficiente [Guttman 1984]. A fim de auxiliar nesta tarefa, muitos índices espaciais foram propostos por diversos pesquisadores [Beckmann et al. 1990, Guttman 1984, Papadias et al. 2001, Samet 1984]. Nesta subseção são apresentados alguns destes índices espaciais.

A R-tree é uma árvore balanceada, similar a uma B-tree [Baruffolo 1999, Bayer e McCreight 1970, Comer 1979] cuja as folhas possuem ponteiros para obje-

tos espaço-textuais. A R-tree é dinâmica, ou seja, inserção e remoção de elementos podem ser realizados em conjunto com consultas sem precisar reorganizar a árvore periodicamente [Guttman 1984]. Além disto, os nós da R-tree, geralmente, possuem o tamanho de uma página de disco, e sua estrutura é planejada para que a busca percorra apenas uma pequena quantidade de nós. Sendo assim, cada nó da R-tree possui um número mínimo e máximo de entradas [Guttman 1984, Rocha-Junior 2012].

Existem dois tipos de nós em uma R-tree: nós intermediários e nós folha. O nó intermediário contém ponteiros para os nós descendentes, enquanto os nós folhas possuem ponteiros para os objetos indexados. As entradas de uma R-tree são formadas por (MBR, *id*). A *Minimum Bounding Rectangle* (MBR) é um retângulo n-dimensional que envolve o objeto indexado, e o *id* é um número que identifica a entrada. O *id* de um nó intermediário é um ponteiro (endereço) para outro nó na árvore (nó descendente), enquanto a MBR de uma entrada intermediária envolve as MBR's de todas as entradas no nó filho. Na entrada de um nó folha, o *id* é a identificação do objeto na base de dados e a MBR é menor retângulo n-dimensional possível capaz de envolver o objeto espacial [Rocha-Junior 2012].

A Figura 2.6(a) é a representação de uma área espacial onde os objetos (p) estão indexados em uma R-tree. Logo, $q.l$ é o local da consulta, r define a vizinhança espacial de $q.l$, e o $m_1, m_2, m_3 eraiz$ são as MBR's. Ao lado, na Figura 2.6(b), a raiz é um nó intermediário que possui três entradas intermediárias m_1, m_2, m_3 que apontam para os nós folha n_1, n_2, n_3, respectivamente. A entrada intermediária $(m_1, *n_1)$ possui uma MBR m_1 que envolve todos os objetos armazenados no nó n_1, e um ponteiro $*n_1$ apontando para o nó que a MBR desta entrada envolve, ou seja o nó n_1. O nó folha n_1 contém duas entradas folha: $(m_{p1}, *p_1)$ e $(m_{p3}, *p_3)$, onde m_{p1} é a MBR que envolve o objeto espacial p_1 e $*p_1$ é o ponteiro (identificador) do objeto p_1 na base de dados.

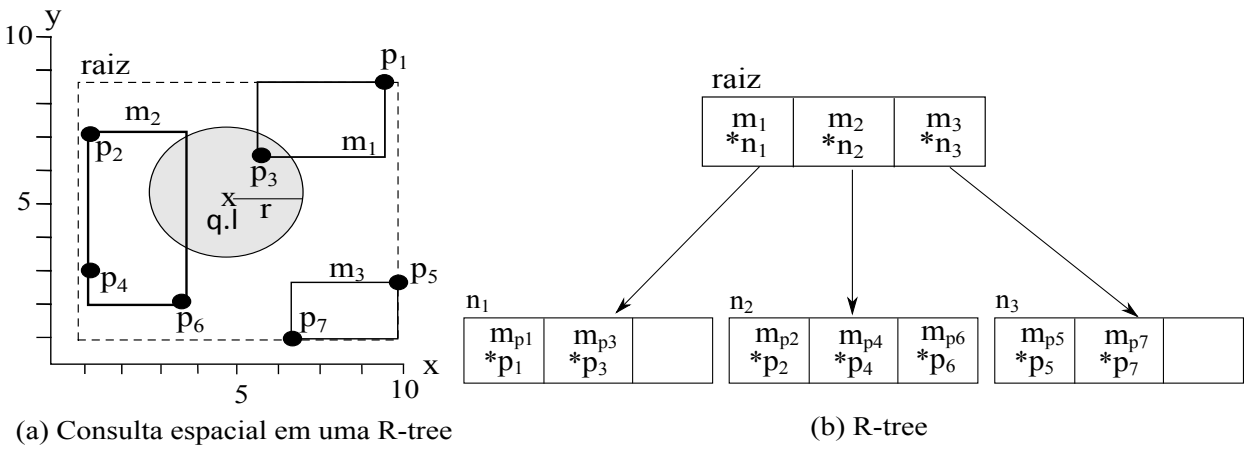


Figura 2.6: Exemplos de R-tree. Adaptado de Rocha-Junior [Rocha-Junior 2012].

Exemplo. A Figura 2.6(a) apresenta um exemplo de consulta range na R-tree. A consulta busca por objetos espaciais dentro da vizinhança espacial definida por r,

ou seja, busca por objetos que estejam dentro da circunferência que tem o 'x' como centro e o r como raio. A consulta range é iniciada na raiz e busca por todas as entradas cuja a distância mínima entre a sua MBR e q.l sejam menores do que r. Sendo p o ponto mais próximo de q.l em uma MBR, a distância mínima de uma MBR é definida por dist(p, q.l), onde dist(p, q.l) é a distância entre p e q.l. Duas entradas satisfazem esta condição: $(m_1, *n_1)$ e $(m_2, *n_2)$. Assim, os nós folha n_1 e n_2 são acessados em busca das entradas folha cuja a MBR[2] esteja dentro da vizinhança espacial definida por r, resultando no objeto p_3 como resposta.

A R-tree é baseada em uma otimização heurística que consiste em minimizar a área da MBR de cada nó intermediário. Porém este critério demonstrou não ser o melhor possível [Beckmann et al. 1990]. Uma das variações mais conhecidas da R-tree é a R*-tree [Chen et al. 2013, Hariharan et al. 2007, Wu et al. 2012b, Zhou et al. 2005]. A R*-tree é superior à R-tree em desempenho do processamento da consulta e no algoritmo que define a MBR dos nós [Beckmann et al. 1990].

A R*-tree reduz a área de cobertura das MBR's dos nós intermediários. Assim, menos ramos da árvore são utilizados durante o processamento da consulta, resultando em um menor acesso à páginas de disco. Além disto, a R*-tree reduz a sobreposição entre MBR's, reduzindo a probabilidade de haver mais de uma MBR abrangendo a mesma área e aumentando a eficiência da consulta, por consequência [Rocha-Junior 2012].

Outra variação muito utilizada da R-tree é a *aggreagate* R-tree (aR-tree), proposta por [Papadias et al. 2001]. A principal característica da aR-tree é utilizar dados não espaciais pré-agregados para otimizar o processamento da consulta. Em outras palavras, cada nó de um aR-tree possui um dado não espacial (ex: um valor numérico) agregado.

Exemplo. Assuma que cada objeto p existente na Figura 2.6 possua um escore (valor numérico) não espacial. Neste contexto, pode ser realizada uma consulta em busca dos objetos que estejam na vizinhança espacial definida por r e que possuam um escore maior do que 0.7. Em uma R-tree tradicional, esta consulta necessita ser realizada em duas etapas. Inicialmente, todos os objetos que estejam na vizinhança espacial de q.l são selecionados. Em seguida, o escore de cada objeto selecionado é verificado, e apenas aqueles que possuem um escore maior que 0.7 são retornados [Rocha-Junior 2012, Papadias et al. 2001].

A fim de otimizar esse processo, cada nó intermediário da aR-tree armazena um valor que é obtido através de uma função aggregated aplicada nas entradas dos nós filhos. Para isto, uma Max aR-tree, que utiliza a função aggregated max(), pode ser utilizada. Assim é agregado a cada nó intermediário o valor máximo de escore existente nos seus nós filho [Rocha-Junior 2012, Papadias et al. 2001].

[2]Neste caso, as entradas folha são pontos bidimensionais. Portanto, o vértice superior direito é idêntico ao vértice inferior esquerdo.

A Figura 2.7 representa uma Max aR-tree onde foi aplicada a função aggregated max(). Por isto, observa-se que o escore armazenado na entrada intermediária $(m_1, 0, 9, *n_1)$ *é 0,9 porque este é o maior valor de escore entre as entradas do nó* n_1*:* $(m_{p1}, 0, 5, *p_1)$ *e* $(m_{p3}, 0, 9, *p_3)$*. A estrutura e a forma como a consulta da aR-tree são executadas são semelhantes a da R-tree. Porém, apenas as entradas que satisfazem as condições espaciais e não-espaciais são visitadas. Por exemplo, para encontrar os objetos que estão na vizinhança espacial de q.l e possuem um escore maior do que 0.7, a raiz é acessada em busca da entrada que satisfaça estas duas condições (critério de vizinhança e escore). Assim, apenas a entrada* $(m_1, 0, 9, *n_1)$ *é visitada, e o objeto* p_3 *é retornado pois é o único que possui um escore maior que 0.7 e possui uma distância para q.l menor do que r.*

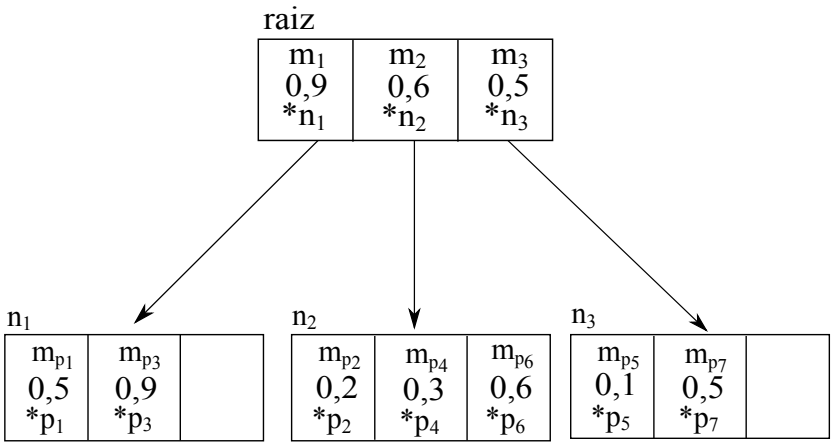


Figura 2.7: Exemplo de aR-tree.

2.3 Consultas Preferenciais

Consultas tradicionais realizadas em bancos de dados fornecem uma maneira rígida para definir as características dos dados que necessitam ser recuperados [Lacroix e Lavency 1987]; resultando em um conjunto resposta muito grande, ou muito pequeno, de dados recuperados. Por isto, o tratamento das preferências do usuário tem se tornado uma ferramenta importante nos Sistemas de Informação atuais [Chomicki 2003]. Estas preferências são utilizadas para filtrar a informação, reduzindo o volume de dados apresentado ao usuário [Chomicki 2003].

Exemplo. A Tabela 2.1 representa um conjunto de dados H que contém informações sobre hotéis, os seus respectivos valores de diárias, e a distância destes hotéis para a praia. Assuma um usuário que não conheça o conjunto de dados e deseje localizar um hotel barato. Utilizando consultas tradicionais, o usuário pode solicitar a lista de hotéis com valores de diária inferiores a 60. Neste caso, nenhum hotel será retornado. Em contrapartida, caso o usuário solicite a lista de hotéis com diárias

superiores a 60, toda base será retornada. Desta forma, o usuário terá que percorrer toda base até encontrar o hotel que deseja, dificultando o processo de localizar o hotel mais barato.

Tabela 2.1: Exemplo de conjunto de dados contendo os valores das diárias dos hotéis e a distância destes hotéis para a praia. Fonte: Tabela adaptada de Rocha-Junior [Rocha-Junior 2012].

Hotel	Diária (R$)	Distância (m)
h_1	300	50
h_2	100	100
h_3	500	100
h_4	90	300
h_5	250	500

As Consultas Preferenciais [Lacroix e Lavency 1987] permitem que o usuário possa expressar as suas preferências de forma mais clara e precisa[3]. É possível solucionar o problema descrito anteriormente definindo uma consulta da seguinte forma: "selecione os hotéis com os menores valores de diária, pare depois de k" [Rocha-Junior 2012]. Assim, considerando $k = 3$ e utilizando os dados da Tabela 2.1, esta consulta Preferencial retorna os hotéis h_2, h_4, h_5.

As consultas Preferenciais podem ser classificadas pela forma como elas expressam a necessidade por informação de um usuário. A consulta Preferencial qualitativa especifica a preferência do usuário diretamente entre pares de objetos (tuplas) existentes na base de dados, utilizando para isto, uma fórmula preferencial $f(h, q)$. Dado dois objetos h e q em um conjunto de dados H, a fórmula preferencial $f(h, q)$ determina se um objeto atende a necessidade do usuário. A formula preferencial $f(h, q)$ é uma operação binária entre os objetos h e q. Sendo assim, quando o resultado desta fórmula é verdadeiro (*true*), entende-se que o objeto h atende melhor às necessidades do usuário do que o objeto q. A fórmula preferencial é definida utilizando operadores lógicos [Chomicki 2003, Kießling 2002].

Exemplo. Considere a base de dados representada pela Tabela 2.1 e um usuário interessado pelo hotel mais barato e mais próximo da praia. Este interesse do usuário pode ser descrito através da fórmula preferencial $f_1(h, q) = [(h[diária] \leq q[diária]) \wedge (h[distância] \leq q[distância]$. *O objeto* h_2 *satisfaz melhor a necessidade do usuário do que o objeto* h_3. *Ambos os objetos possuem a mesma distância para a praia, porém o hotel* h_2 *é mais barato. Neste contexto, é dito que* h_3 *é dominado por* h_2 *pois* $f_1(h_2, h_3) = true$. *Todos os objetos que não são dominados são respostas válidas para a consulta descrita pela fórmula* f_1.

[3]Borzsony, Kossmann e Stocker [Borzsony et al. 2001] demonstram como uma consulta preferencial pode ser implementada utilizando SQL (sem realizar modificações no sistema de banco de dados), e os motivos pelos quais tal implementação apresenta baixo desempenho quando comparado a uma consulta preferencial (implementada a partir da extensão de um sistema de banco de dados com um novo operador lógico, que representa a consulta preferencial).

Por outro lado, a consulta Preferencial quantitativa especifica as preferências indiretamente para cada objeto existente no conjunto de dados. Uma função escore avalia os atributos de um objeto, e produz um valor numérico (escore) que representa a importância deste objeto para as necessidades do usuário. Consultas quantitativas são frequentemente denominadas de consultas top-k. Neste tipo de consulta é especificado a função para calcular o escore dos objetos e a quantidade (k) de objetos que são recuperados da base de dados [Rocha-Junior 2012].

Exemplo. Na base de dados H apresentada na Tabela 2.1, o hotel h_1 *pode ser representado por* $h_1 = \{300, 50\}$*, sendo que o valor 300 está posicionado na coluna (dimensão) 1 da tabela e o valor 50 na coluna 2. Pode-se utilizar a consulta Preferencial quantitativa para encontrar os 3 hotéis mais baratos e mais próximos da praia. Supondo uma função escore* $f(h) = 0,5 * h[diária] + 0,5 * h[distância]$*, onde* $h \in H$*, os objetos com menores escores são os que se aproximam mais da necessidade do usuário. Sendo assim, a função escore retorna os valores de escore* $f(h_2) = 100$*,* $f(h_1)$*=175 e* $f(h_3)$*=195 para os os objetos* h_2*,* h_1 *e* h_3*, respectivamente. Portanto, a consulta Preferencial quantitativa retorna os objetos* h_2*,* h_1 *e* h_3 *como resposta. Esta função escore seria considerada monótona se para todo objeto* $h_x, h_y \in H$*,* $f(h_x) \leq f(h_y)$ *quando* $h_x[i] \leq h_y[i]$*. Como* $f(h_2) \leq f(h_1)$ *mas* $h_2[2] > h_1[2]$ *(100 > 50), esta função de escore não é considerada monótona.*

A maioria das técnicas de processamento de consultas top-k utilizam funções escore denominadas de funções *ranking* monótonas, pois estas funções possuem propriedades especiais que permitem o processamento eficiente da consulta top-k [Ilyas et al. 2008]. Sendo um objeto $h \in H$ representado por $h = h[1], ..., h[n]$, onde $h[i]$ é um valor numérico na dimensão i. Uma função f_h definida sobre os atributos (dimensões) de um objeto h é monótona, se para todos os objetos $h, q \in H$, $f_h \leq f_q$ quando $h[i] \leq q[i]$ para todo i [Rocha-Junior 2012].

2.4 Consultas Espaciais Preferenciais Tradicionais

Os bancos de dados espaciais gerenciam grandes coleções de entidades geográficas. Cada entidade tem como característica principal as coordenadas geográficas, que indicam a posição do objeto no espaço. Porém, em conjunto com o atributo espacial é comum existir informações não-espaciais como descrição textual, nome do objeto, tamanho ou preço deste objeto [Yiu et al. 2007].

Consultas espaciais top-k retornam um conjunto de objetos espaciais que podem atender a necessidade do usuário. Entretanto, cada consulta define seu próprio conjunto de parâmetros para representar a preferência do usuário. Yiu et al. [Yiu et al. 2007] apresenta um novo tipo de consulta Top-k, a consulta Espacial Preferencial Tradicional. Nesta consulta, os k melhores objetos de interesse para

um usuário são definidos através da qualidade dos objetos de referência[4] (*features*) existentes na vizinhança espacial de cada objeto de interesse.

Desta forma, dado um conjunto P de objetos de interesse, a consulta Espacial Preferencial Tradicional retorna os k objetos existentes em P com os maiores escores. O escore de um objeto de interesse é definido pela qualidade dos objetos de referência (ex: cafés, restaurantes, hospitais) existentes na sua vizinhança. Nesta consulta, a qualidade de um objeto de referência (*feature*) pode ser obtida através de um sistema de classificação online, como o Booking[5] ou o Foursquare[6], onde os usuários avaliam diversos tipos de objetos de referência [Yiu et al. 2007].

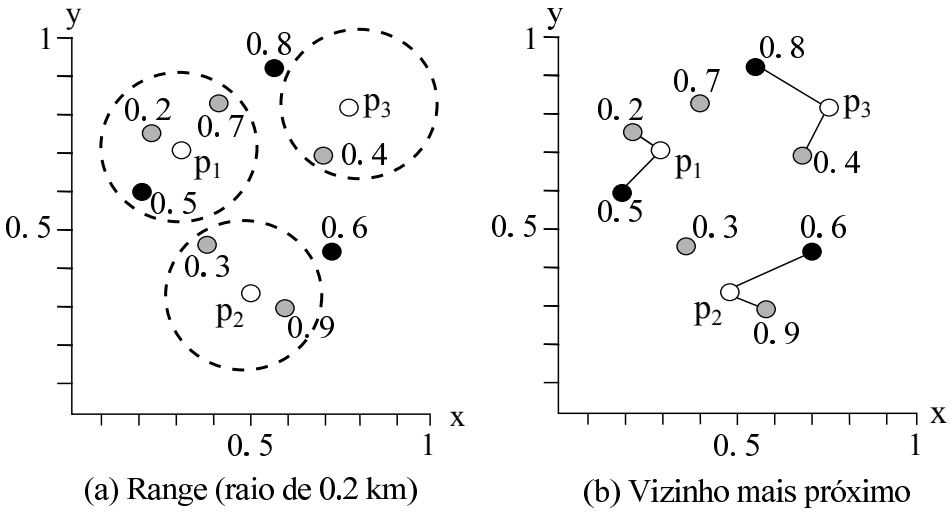


Figura 2.8: Exemplos de consultas Espaciais Preferenciais utilizando diferentes maneiras de definir a vizinhança espacial do objeto de interesse. Fonte: Yiu et al. [Yiu et al. 2007].

Exemplo. Os pontos brancos p na Figura 2.8 representam objetos espaciais de interesse. Além destes, os pontos cinza representam restaurantes, enquanto os pontos pretos representam cafeterias. Cada objeto existente no conjunto de restaurantes e cafeterias (pontos pretos e cinza) possuem um valor de escore pré-definido, representado pelo número real posicionado ao redor de cada um destes pontos.

Assumindo que o usuário deseja obter os melhores hotéis em termos de cafeterias e restaurantes, a consulta Espacial Preferencial Tradicional retorna os objetos de interesse (hotéis) com os maiores escores. Neste caso, o escore do objeto de interesse é computado pela soma do escore do melhor restaurante existente na vizinhança do objeto de interesse, com o valor da melhor cafeteria existente nesta mesma vizinhança.

[4]Considere "objeto de referência" como uma classe de objetos em um mapa espacial, como uma instalação específica ou um serviço [Yiu et al. 2007]. Cada objeto de referência é associado a um escore que é pré-definido por um sistema de classificação.

[5]www.booking.com

[6]www.foursquare.com

Sendo assim, os valores dos objetos de interesse para a consulta do tipo range são $\tau(p_1) = 0,7 + 0,5 = 1,2$ *,* $\tau(p_2) = 0,9 + 0 = 0,9$ *e* $\tau(p_3) = 0,4 + 0 = 0,4$*. Objetos de interesse que não possuem cafeterias ou restaurantes na sua vizinhança recebem o valor zero como escore deste objeto de referência ausente, como aconteceu nos casos dos objetos de interesse* p_2 *e* p_3*. Através dos resultados extraídos da Figura 2.8, obtemos o objeto* p_1 *como o resultado top-1 da consulta Espacial Preferencial Tradicional do tipo range [Yiu et al. 2007].*

De forma semelhante, é calculado o escore para a consulta Espacial Preferencial Tradicional do tipo vizinho mais próximo (Figura 2.8 (b)). Portanto, nesta consulta, o escore do objeto de interesse é o escore do objeto de referência mais próximo. Assim, temos $\tau(p_1) = 0,2+0,5 = 0,7$*,* $\tau(p_2) = 0,9+0,6 = 1,5$ *,* $\tau(p_3) = 0,4+0,8 = 1,2$*, resultando em* p_2 *como o melhor hotel [Yiu et al. 2007].*

Deste modo, a consulta Espacial Preferencial Tradicional utiliza duas etapas para selecionar seus objetos de interesse. Primeiro, calcula a distância do objeto de interesse para um determinado objeto de referência (*feature*). Em seguida, ordena os objetos de interesse a partir de sua qualidade [Yiu et al. 2007].

Além desta consulta Top-k, muitas pesquisas estão sendo desenvolvidas nesta área [Liu et al. 2011], [Yiu et al. 2011], [Cao et al. 2012] e [Rocha-Junior et al. 2010], demonstrando a popularidade das consultas Top-k.

2.5 Consultas Espaciais Preferenciais Textuais

Entre as consultas espaciais existem aquelas que utilizam palavras-chave para expressar o que o usuário deseja obter da base de dados. Nesta seção serão descritas algumas consultas que utilizam este modelo para recuperar a informação desejada, além de índices híbridos capazes de indexar dados espaciais e textuais simultaneamente. Estes índices espaço-textuais visam dar suporte a um processamento eficiente de consultas espaço-textuais.

Uma *Top-k Spatial Keyword Query* (SK) [Cao et al. 2012, Chen et al. 2013] utiliza a localização do usuário e um conjunto de palavras-chave, fornecidas por ele, como parâmetros. A consulta identifica objetos que são espacialmente próximos a localização do usuário, e textualmente relevantes às palavras-chave; retornando os k objetos que possuem estas duas características (proximidade do usuário e relevância textual). Uma função escore avalia a proximidade espacial entre um objeto e o usuário, além da relevância textual da descrição do objeto considerando o conjunto de palavras-chave. A resposta desta consulta é ordenada a partir dos valores de escore gerados para cada objeto pela função escore.

Supondo um usuário que deseja encontrar um bar onde tenha apresentação de samba, utilizando a área espacial descrita pela Figura 2.9. Este usuário forma uma consulta

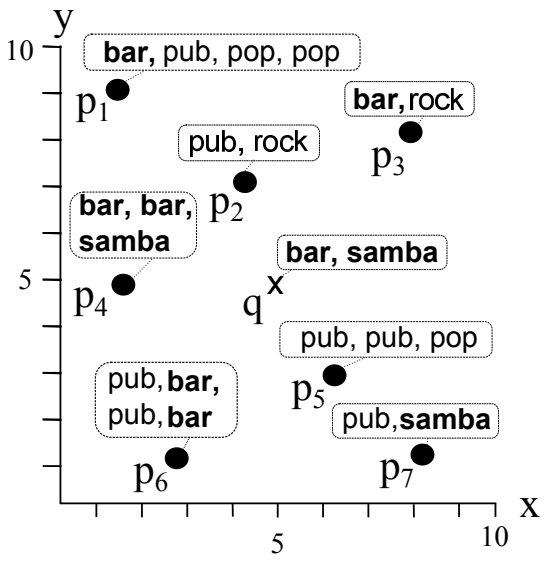


Figura 2.9: Área espacial contendo bares e pubs. Fonte: Rocha-Junior et al. [Rocha-Junior et al. 2011].

q top-3 com as seguintes palavras-chave: "samba" e "bar". A localização do usuário é a localização da consulta q na Figura 2.9.

A *Top-k Spatial Keyword Query* retorna um conjunto resposta formado pelos objetos p_4, p_6, p_7. O objeto p_4 é o resultado top-1 pois a sua descrição textual é semelhante às palavras chaves fornecidas pelo usuário, e é o objeto mais próximo da localização da consulta q. p_6 possui uma relevância maior do que p_7, pois a descrição textual do p_6 é mais relevante e este objeto também fica mais próximo de q do que p_7.

2.5.1 Índices espaço-textuais

Atualmente, muitas aplicações utilizam uma grande quantidade de dados geo-referenciados, como o Twitter[7] e o Flickr[8]. Estas aplicações podem se beneficiar da *Top-k Spatial Keyword Query* (SK) e de outras consultas espaço-textuais, porém o custo do processamento destas consultas é alto [Rocha-Junior et al. 2011]. Por isto, índices espaço-textuais desempenham um papel importante para o processamento destas consultas. Ao indexar dados que contém descrições textuais e geo-referenciamentos, estes índices híbridos possibilitam um processamento eficiente das consultas espaço-textuais [Chen et al. 2013].

O índice espaço-textual Spatial-Keyword Inverted File (SKIF) proposto por Khodaei, Shahabi e Li [Khodaei et al. 2010] é um Arquivo Invertido (IF) capaz de indexar e buscar por dados espaciais e textuais de forma integrada, utilizando apenas

[7] www.twitter.com
[8] www.flickr.com

uma estrutura para gerenciar os dois tipos de dados. Para isto, o espaço é particionado em células (Figura 2.10), onde cada célula é tratada como uma palavra-chave textual.

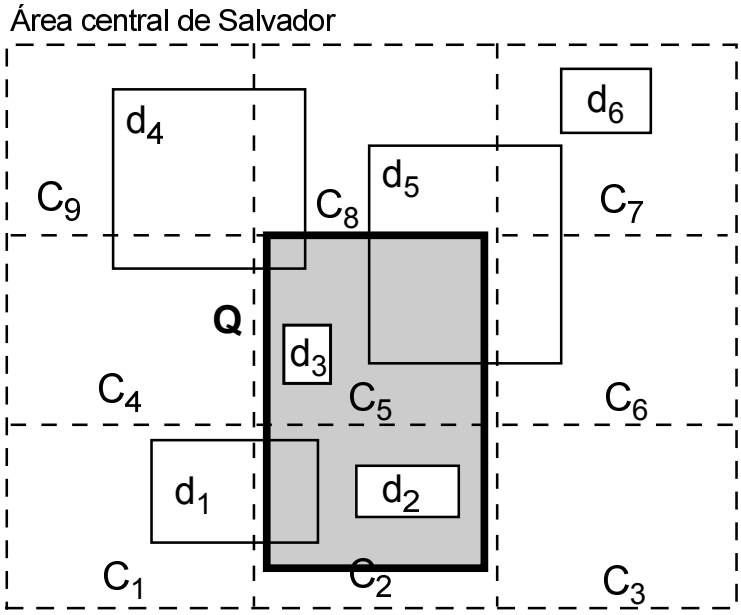


Figura 2.10: A área de busca é restrita pelo usuário (ex: "Área central de Salvador") e dividida em grid (C_i) pelo sistema para utilizar o SKIF. O retângulo escuro corresponde ao local onde a consulta foi realizada. Retângulos claros (d_i) correspondem a documentos próximos a área de interesse do usuário. Fonte: Imagem adaptada de [Khodaei et al. 2010].

De forma semelhante ao IF, o SKIF é composto por um vocabulário e por um conjunto de listas invertidas. O vocabulário contém todos os termos existentes nas descrições textuais dos objetos, e identificadores para as células que formam o grid sobre a área espacial de interesse do usuário. Para cada termo distinto, os seguintes valores são armazenados no vocabulário: o número de documentos f_t que possuem o termo t, um ponteiro para lista invertida correspondente, e o tipo de termo indexado. Cada termo t possui uma lista invertida correspondente, cada uma destas listas armazenam os identificadores dos documentos que possuem o termo t e a frequência normalizada com o que termo t aparece em cada documento D [Khodaei et al. 2010].

O SKIF foi idealizado para processar uma consulta capaz de retornar os k objetos que possuem os maiores escores textuais e espaciais concomitantemente. Para o SKIF, a localização de cada objeto é tratada como uma região ao invés de um ponto. A relevância espacial é expressada pela sobreposição entre a região da consulta (região escura) e a região de um objeto (d_i) [Chen et al. 2013]. Por isto, apesar de ser um índice hibrido, o SKIF não é capaz de processar uma consulta *Top-k Spatial Keyword Query*. Recentemente, Chen et al. [Chen et al. 2013] modificou o SKIF

para processar consultas do tipo *Boolean Range Query* (BRQ).

Assim como o SKIF, muitas outras estruturas baseadas em células já foram propostas para processar objetos espaço-textuais [Bentley e Friedman 1979, Guttman e Stonebraker 1982]. A seguir são discutidos outros índices híbridos capazes de indexar objetos espaço-textuais, estes índices são baseados em árvores como a R-tree proposta por Guttman [Guttman 1984] e a R*-tree proposta por Beckmann et al. [Beckmann et al. 1990].

Cong, Jensen e Wu [Cong et al. 2009] incorporam a similaridade entre documentos para produzir um novo índice espaço-textual denominado DIR-tree (*Document similarity enhanced Inverted file R-tree*). A DIR-tree combina informações espaciais e textuais durante a construção do seu índice, mantendo no mesmo nível da árvore, os objetos próximos espacialmente e maximizando a semelhança textual entre os documentos que fazem parte de uma mesma MBR [Cong et al. 2009, Wu et al. 2012a]. Um parâmetro β é introduzido para determinar o peso entre a característica espacial e textual. Por exemplo, a depender do valor de β os objetos de uma mesma MBR podem ser muito próximos um do outro e possuir pouca relevância textual entre si.

Cada nó de uma DIR-tree é associado a um Arquivo Invertido (IF). Além do IF, cada nó folha da DIR-tree é associado a um sumário de documentos que disponibiliza informações textuais sobre os documentos [Li et al. 2011]. A Figura 2.11 apresenta objetos espaciais O_i e suas respectivas MBRs R_i. A Figura 2.12 exemplifica a organização de uma DIR-tree para indexar os objetos existentes na Figura 2.11.

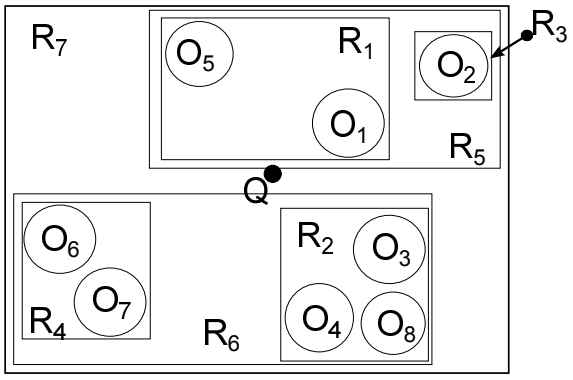


Figura 2.11: Objetos espaciais O_i e suas respectivas MBR's R_i. Fonte: [Cong et al. 2009].

Exemplo. Considere uma consulta espaço-textual que receba um conjunto de palavras-chaves T. Assuma que o objeto O_1 (Figura 2.11) é o mais relevante textualmente para este conjunto de palavras-chave. Sendo assim, para processar esta consulta Q, é necessário percorrer apenas os nós R_7, R_5 e R_1, da árvore ilustrada na Figura 2.12, para chegar ao objeto O_1 e retorná-lo como resposta.

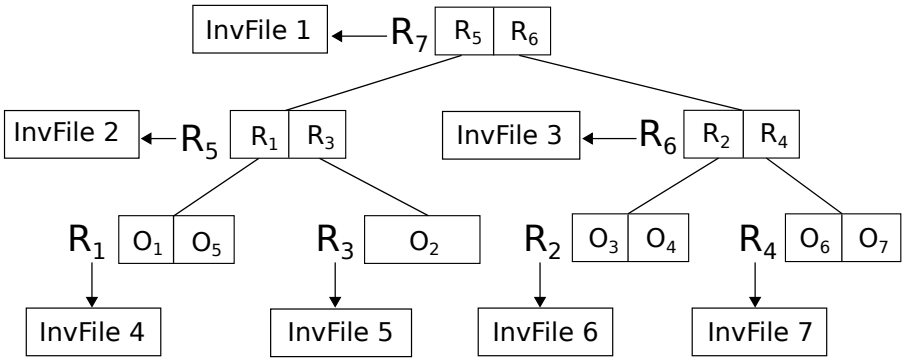


Figura 2.12: Estrutura de uma DIR-tree. Fonte: [Cong et al. 2009].

Em alternativa, Rocha-Junior et al. [Rocha-Junior et al. 2011] propõe uma nova estrutura para indexar os dados espaço-textuais a fim de otimizar o processamento da consulta *Top-k Spatial Keyword Query* (SK). O *Spatial Inverted Index* (S2I) é semelhante ao Arquivo Invertido [Zobel e Moffat 2006, Rocha-Junior et al. 2011], porém o armazenamento dos termos mais frequentes da coleção é feito de forma diferenciada. O S2I mapeia os termos mais frequentes da coleção para uma *aggregated R-tree* (aR-tree) [Papadias et al. 2001], sendo que cada árvore armazena apenas os objetos que possuem o mesmo termo t. Enquanto os termos menos frequentes são armazenados em blocos, dentro de um arquivo; sendo que cada bloco armazena os objetos que possuem o mesmo termo t.

termo	id	df_t	*flag*	armazenamento
escola	t_1	4	árvore	aR^{t_1}
creche	t_2	3	árvore	aR^{t_2}
infantil	t_3	3	árvore	aR^{t_3}
idioma	t_4	1	bloco	(p_5)

Figura 2.13: *Spatial Inverted Index.*

O S2I (exemplificado na Figura 2.13) é composto por vocabulário, blocos de arquivo (p_i) e aR-tree's (aR^{t_i}). O vocabulário armazena cada termo distinto existente na base de dados (ex: "escola", "creche"). Para cada termo t_i, é armazenado a quantidade de documentos em que este termo ocorre df_t, um *flag* indicando em que tipo de estrutura o termo está armazenado (bloco ou árvore) e um ponteiro apontando para o bloco ou árvore que contém o termo (representado pela seta unidirecional na Figura 2.13).

Cada bloco de arquivo armazena um conjunto de objetos. Para cada objeto é armazenado a identificação deste objeto $p.id$, a localização do objeto $p.l$ e a frequência $f_{p,t}$ com que o termo t ocorre na descrição textual deste objeto.

Os nós folha da aR-tree armazenam as mesmas informações de um objeto espacial que o bloco de arquivo: $p.id$, $p.l$ e $f_{p,t}$. Os nós intermediários armazenam um *minimum bounding rectangle* (MBR) que envolve a localização espacial de todos os objetos que estão na sub-árvore. Os nós intermediários de uma aR-tree [Papadias et al. 2001] são capazes de armazenar também um valor não espacial. O valor agregado aos nós intermediários de uma aR-tree é o valor máximo de $f_{p,t}$ que um objeto p pode possuir entre os objetos existentes na sub-árvore deste nó intermediário. Assim, os objetos podem ser acessados decrescentemente pelos valores de impacto do termo t na descrição textual de seu objeto ($f_{p,t}$), e da proximidade espacial [Rocha-Junior et al. 2011].

Segundo Rocha-Junior et al. [Rocha-Junior et al. 2011], os resultados obtidos utilizando o S2I demonstram que a nova estrutura é capaz de otimizar o custo da consulta, assim como o custo para atualizar um termo existente na coleção. Para consultas com apenas um termo, o S2I percorre apenas uma pequena árvore, ou bloco de arquivo. Quando a consulta possui vários termos, é necessário percorrer apenas um conjunto de pequenas arvores, ou blocos de arquivos; dispensando o acesso a um índice invertido externo.

Capítulo 3

Preliminares

"A simplicidade é o último grau de sofisticação"

– Clare Boothe Luce

Neste capítulo são apresentadas definições importantes para compreender a consulta EPPC. Inicialmente é apresentada a definição da consulta Espacial Preferencial por Palavra-chave. Em seguida, é exposto o Arquivo Invertido Adaptado utilizado para processar a consulta EPPC.

3.1 Definição da consulta

A consulta *Espacial Preferencial por Palavra-chave* (EPPC) Q é formada por um conjunto de palavras-chave $Q.D$, pela definição da vizinhança espacial de interesse $Q.\psi$, e pela quantidade $Q.k$ de objetos de interesse que se deseja obter como resposta: $Q = (Q.D, Q.\psi, Q.k)$.

Dado um conjunto de objetos de interesse P, onde cada objeto $p \in P$ possui uma coordenada espacial $p = (p.x, p.y)$; e um conjunto de objetos espaço-textuais de referência F, onde cada objeto $f \in F$ possui uma coordenada espacial $(f.x, f.y)$ e um texto $f.D$, $f = (f.x, f.y, f.D)$, a consulta Q retorna os k objetos de interesse contidos em P que possuem os maiores escores. O escore $\tau^{Q.\psi}(p)$ de um objeto de interesse $p \in P$ é a maior relevância textual (similaridade textual) entre os objetos espaço-textuais de referência f que atendem ao critério de vizinhança espacial $Q.\psi$.

O usuário pode optar entre três formas possíveis de especificar o critério de vizinhança espacial $Q.\psi$: seleção espacial ($Q.\psi = rng$), vizinho mais próximo ($Q.\psi = nn$) ou influência ($Q.\psi = inf$) [Yiu et al. 2007].

- Seleção espacial $\tau^{rng}(p)$, dado um raio r:

$$\tau^{rng}\ (p) = max\ \{\ \theta(f.D, Q.D)\ |\ f \in F : dist(p, f) <= r\ \}$$

- Vizinho mais próximo $\tau^{\ nn}\ (p)$:
  $\tau^{nn}\ (p) = max\ \{\ \theta(f.D, Q.D)\ |\ f \in F, \theta(f.D, Q.D) > 0, \forall\ v \in F : dist(p, f) \leq dist(p, v)\ \}$

- Influência $\tau^{\ inf}\ (p)$, dado um raio r:
  $$\tau^{inf}\ (p) = max\ \{\ \theta(f.D, Q.D) \cdot 2^{\ -dist(p,f)/r}\ |\ f \in F\ \}$$

onde $\theta(f.D, Q.D)$ é a função cosseno que retorna a relevância textual (similaridade textual) entre o texto do objeto de referência $f.D$ e o conjunto de palavras-chave $Q.D$ [Zobel e Moffat 2006, Rocha-Junior et al. 2011], enquanto $dist(p, f)$ é a função que calcula a distância entre um objeto p e um estabelecimento f.

Quando o usuário opta pela *seleção espacial* ($Q.\psi = rng$), o escore de um objeto de interesse $p \in P$ é definido pela maior relevância textual $\theta(f.D, Q.D)$ entre os objetos $f \in F$ que atendem ao critério de vizinhança espacial, ($dist(p, f) \leq r$). Ao optar por *vizinho mais próximo* ($Q.\psi = nn$), o escore é definido pelo objeto de referência mais próximo de p que seja textualmente relevante $\theta(f.D, Q.D) > 0$. Caso existam vários objetos de referência f com a mesma proximidade espacial do objeto de interesse p, o escore de p é a maior relevância textual $\theta(f.D, Q.D)$ entre os objetos f que possuem a mesma menor distância. O critério de vizinhança *influência* ($Q.\psi = inf$) define o escore do objeto de interesse p levando em consideração a relevância textual entre $f.D$ e $Q.D$ e a distância entre p e f, quanto maior a distância menor o escore (influência).

3.2 Arquivo Invertido Adaptado

Em um Arquivo Invertido, a lista invertida geralmente armazena o id de um Documento D_{id}, e a frequência $f_{t,D}$ em que o termo t ocorre no Documento D. Documento é qualquer unidade sobre a qual se decida construir um Sistema de Recuperação de Informação [Manning et al. 2008]. Logo, para a consulta EPPC, um Documento é um objeto espaço-textual f ($f = (f.x, f.y, f.D)$). Assim, para processar a consulta EPPC de forma eficiente, é necessário armazenar também a localização espacial do objeto espaço-textual no Arquivo Invertido. Isto evita o acesso a uma estrutura externa (ex: B-tree [Knuth 1968]) ao Arquivo Invertido para obter as informações espaciais do objeto.

Deste modo, optou-se por armazenar em cada tupla da lista invertida, além do id do objeto espaço-textual (f_{id}) e da frequência $f_{t,f.D}$ em que o termo t aparece na descrição textual de f, as coordenadas espaciais ($f.x, f.y$) deste objeto espaço-textual. A Figura 3.1 apresenta um conjunto de objetos espaço-textuais e a Figura 3.2 apresenta o Arquivo Invertido Adaptado, que é formado por vocabulário e um conjunto de listas invertidas. O vocabulário armazena a quantidade f_t de objetos

espaço-textuais que possuem o termo t e um ponteiro para a sua lista invertida correspondente. Cada lista invertida é formada por um conjunto de tuplas, que são compostas por *id* do objeto espaço-textual, frequência $f_{t,f.D}$ e as coordenadas espaciais do objeto $(f.x, f.y)$.

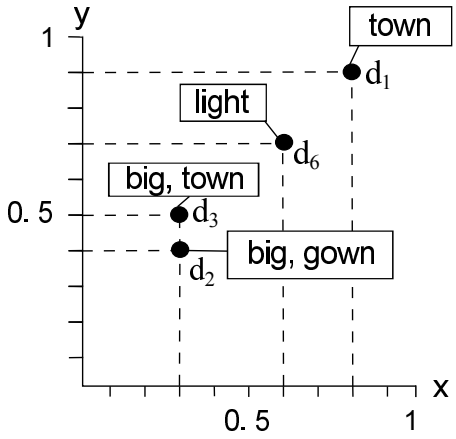


Figura 3.1: Objetos espaço-textuais em uma área espacial

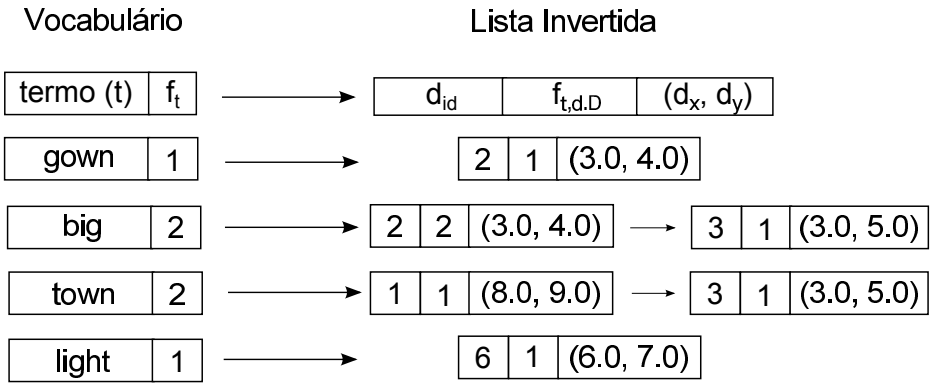


Figura 3.2: Exemplo do Arquivo Invertido adaptado para processar a consulta EPPC.

Exemplo. Dado um conjunto de objetos de interesse e um conjunto de objetos de referência (objetos espaço-textuais), uma consulta EPPC é processada no Arquivo Invertido Adaptado de forma similar a uma consulta textual no Arquivo Invertido, como pode ser visto na Figura 3.3. Os termos ("big", "town", "light") da consulta são enviados ao vocabulário que retorna, para cada termo da consulta, uma lista invertida correspondente. O escore textual parcial de cada objeto de referência é computado no acumulador e dividido pelo peso da descrição textual do objeto $W_{f.D}$ (como discutido no Capítulo 2, Seção 2.1). Em seguida, o filtro verifica quais objetos atendem ao critério de vizinhança espacial do objeto de interesse. O objeto de referência que satisfaz o critério de vizinhança atribui seu escore textual ao respectivo objeto de interesse. Caso outro objeto de referência já tenha submetido seu escore

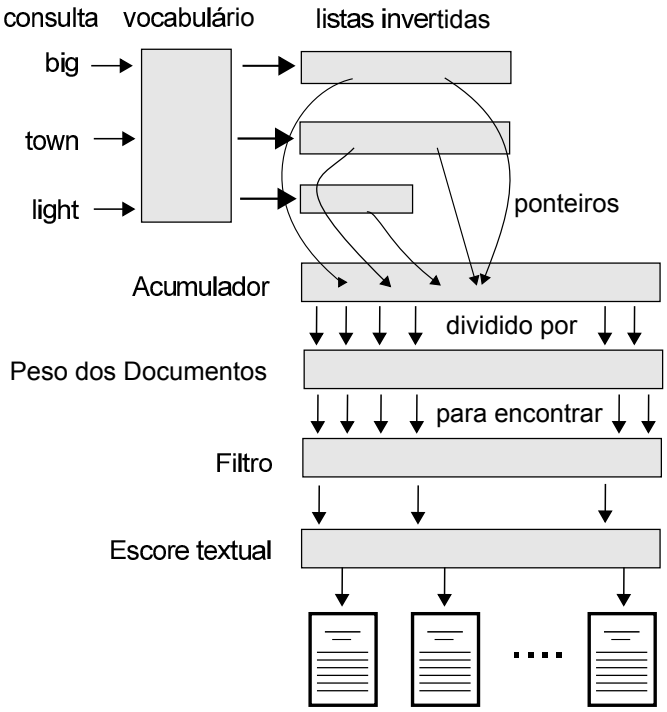


Figura 3.3: Exemplo de execução da consulta EPPC utilizando o Arquivo Invertido Adaptado.

ao objeto de interesse, um novo escore só poderá ser submetido caso este seja maior do que o já existente.

Com esta adaptação é possível processar de forma eficiente a consulta EPPC, pois a latitude e longitude de cada objeto são armazenadas na mesma tupla onde é armazenada a sua informação textual. Desta forma, quando uma tupla é retornada, é possível verificar se o objeto atende ao critério de vizinhança selecionado.

Capítulo 4

Algoritmos Propostos

"Não fique para sempre na via pública. Deixe o caminho batido para trás e mergulhe na floresta"
– Alexander Graham Bell

Nesta seção são apresentados três novos algoritmos para processar a consulta EPPC. Em todos os algoritmos, o conjunto de objetos de interesse P está armazenado em uma R*-tree [Beckmann et al. 1990]. O primeiro algoritmo (IFA) utiliza o Arquivo Invertido Adaptado para indexar o texto dos objetos de referência $f \in F$. Já os outros dois algoritmos (SIA e SIA$^+$), utilizam o índice espaço-textual S2I [Rocha-Junior et al. 2011] para indexar o texto dos objetos de referência $f \in F$. Em cada seção, cada algoritmo é descrito utilizando o critério de vizinhança seleção espacial ($Q.\psi = rng$). Em seguida, são descritas as modificações necessárias para processar a consulta utilizando influência ($Q.\psi = inf$) ou vizinho mais próximo ($Q.\psi = nn$).

4.1 *Inverted File Based Algorithm*

Uma forma simples de processar a consulta EPPC requer acessar cada objeto de referência existente na base de dados e verificar quais destes objetos estão presentes na vizinhança espacial do objeto de interesse. É feito o cômputo da relevância textual de cada um destes objetos de referência para as palavras-chave de busca, e o maior escore obtido é atribuído ao escore do objeto de interesse. Este processo se repete para todos os objetos de interesse e por fim é apresentado os k melhores ao usuário.

O *Inverted File Based Algorithm* (IFA) utiliza o Arquivo Invertido Adaptado para evitar o acesso a todos os objetos de referência da base de dados. Ao invés disso, o

Arquivo Invertido Adaptado retorna apenas os objetos de referência que são textualmente relevantes para as palavras-chave de busca. É verificado qual dos objetos retornados estão presentes na vizinhança espacial do objeto de interesse e é feito o cômputo da relevância textual destes objetos de referência. O maior valor de relevância obtido é atribuído ao escore do objeto de interesse, e os k objetos de interesse com maiores escores são apresentados ao usuário.

Desta forma, o IFA utiliza o Arquivo Invertido Adaptado para acessar apenas os objetos de referência que são textualmente relevantes para as palavras-chave de busca. Otimizando o acesso aos dados espaciais pois evita que muitos objetos de referência desnecessários para a consulta sejam acessados e verificados.

Utilizando o critério de vizinhança seleção espacial, o escore de um objeto de interesse p é o maior escore textual $f.\theta$ entre os objetos de referência $f \in F$ relevantes para as palavras-chave de busca $Q.D$ ($\theta(f.D, Q.D) > 0$), cuja distância para p seja menor ou igual que o raio $Q.r$, $dist(p, f) \leq Q.r$. Para identificar os objetos de referência f relevantes para as palavras-chaves $Q.D$, é necessário recuperar as listas invertidas de cada termo $t \in Q.D$ ($IF.list(t)$). Estas listas são acessadas em ordem crescente de id do objeto de referência f, facilitando o cálculo do escore textual de f.

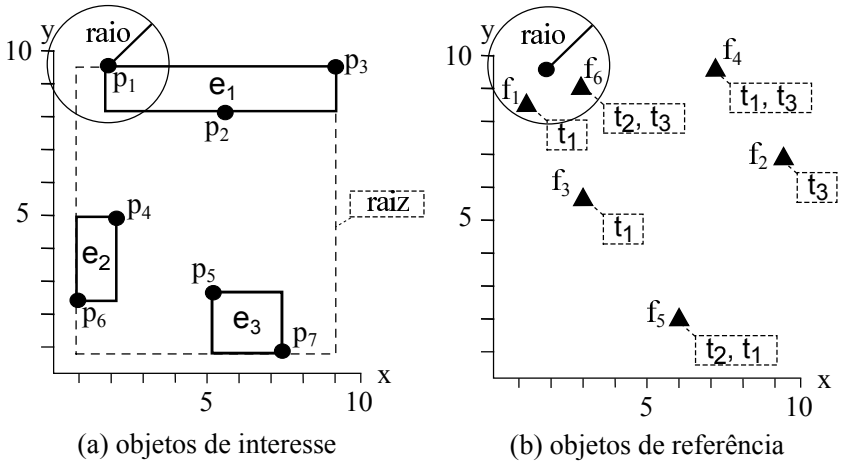


Figura 4.1: Exemplo da seleção do objeto de referência vizinho mais relevante ao objeto de interesse utilizando o IFA e o critério de seleção espacial.

Exemplo. Dada a Figura 4.1, composta por um conjunto de objetos de interesse indexados em uma R-tree (Figura 4.1 (a)) e por um conjunto de objetos de referência (Figura 4.1 (b)); o escore do objeto de interesse p_1 é dado pelo escore textual do objeto de referência que atenda ao critério de vizinhança espacial e possua relevância para o conjunto de palavras-chave $Q.D$. Considerando um conjunto de palavras-chave $Q.D = \{t_1\}$, o escore do objeto de interesse p_1 corresponde ao escore de f_1 pois apenas f_1 possui a palavra-chave de busca t_1 e é vizinho de p_1 (todo objeto que está dentro da circunferência em volta do objeto p_1 é considerado vizinho de p_1).*

Os objetos de referência são retirados da lista invertida $IF.list(t)$ um a um. Cada objeto retirado desta lista possui pelo menos um termo t em sua descrição textual. O

escore textual de cada objeto de referência retornado por $IF.list(t)$ é computado e é verificado qual destes objetos pertence a vizinhança espacial do objeto de interesse. O objeto de referência retornado por $IF.list(t)$ que possui o maior escore e está na vizinhança do objeto de interesse atribui o seu escore textual ao escore do objeto de interesse.

Exemplo. Dada a Figura 4.1, o escore do objeto de interesse p_1 é dado pelo escore textual do objeto de referência que atenda ao critério de vizinhança espacial e possua relevância para o conjunto de palavras-chave $Q.D$. Considerando um conjunto de palavras-chave $Q.D = \{t_1\}$, para calcular o escore do objeto de interesse p_1 é necessário recuperar a lista invertida do termo t_1 ($IF.list(t_1)$). A lista $IF.list(t)$ retorna os objetos f_1, f_3, f_4 e f_5.

Entre todos os objetos de referência retornados pela lista $IF.list(t)$, apenas o objeto f_1 atende ao critério de vizinhança do objeto de interesse p_1, ou seja, entre os objetos retornados, apenas f_1 é vizinho espacial de p_1. Isto ocorre porque a distância de f_1 para p_1 é menor do que o raio $Q.r$ ($dist(p_1, f_1) < Q.r$). Sendo assim, o escore textual de f_1 é atribuído a p_1.

O escore textual de um objeto de referência é a soma dos escores parciais de todos os termos da consulta que estão presentes na descrição textual deste objeto de referência. Ao retirar um objeto de referência f de uma lista invertida para t, o escore textual dele só reflete o escore para o termo t (impacto do termo t em f). Caso este mesmo objeto f, seja localizado em outra lista de um termo $t' \neq t$ e $t' \in Q.D$, o escore textual de f deve ser acrescido do escore parcial de t'. Após computar o escore textual de um objeto de referência f, o algoritmo verifica se f atende ao critério de vizinhança espacial ($dist(p, f) \leq Q.r$) e se o escore atual de p é menor que o escore textual de f ($p.score < f.\theta$), atualizando o escore de p com o escore de f em caso afirmativo.

O Algoritmo 1 apresenta o IFA. O algoritmo computa o escore de cada objeto $p \in P$ (linhas 3-27), inicialmente o escore de p é zero (linha 4). O algoritmo recupera um iterador (linha 6) que tem acesso a todos os objetos da lista invertida do termo t, em ordem crescente de id. Então o algoritmo armazena, em uma *heap* H, uma entrada e composta por dois atributos $e.f$ e $e.l$ ($e = \{e.f, e.l\}$). O atributo $e.f$ é o primeiro objeto da lista t ($iterator.next()$), e o atributo $e.l$ é uma referência para o iterador da lista ($iterator$) (linhas 7-9). O objetivo da *heap* H é armazenar, para cada lista de um termo t, o objeto de referência f com menor id ($e.f$) que ainda não tenha sido visitado e um ponteiro para acessar os demais itens da lista (iterador, $e.l$), permitindo localizar de forma eficiente os objetos de referência que estão armazenadas em mais de uma lista.

O algoritmo continua retirando a entrada da lista cujo o objeto de referência $e.f$ tenha o menor id (linha 11). A função $nextEntry(H)$ retira uma entrada da *heap* ($e \leftarrow heap.pollFirst()$) e armazena na *heap* uma nova entrada $\{e.l.next(), e.l\}$ composta pelo próximo objeto de referência da lista ($e.l.next()$) e pelo ponteiro para o

iterador da lista ($e.l$), caso a lista ainda tenha objetos de referência a serem visitados. Enquanto e for diferente de *null* e enquanto a *heap* H não estiver vazia (linhas 12-20), o algoritmo computa o escore de p. Para computar o escore de p, é preciso computar o escore textual do objeto de referência f, somando os escores parciais de f em cada lista que ele aparece. Como as listas estão ordenadas pelo *id* dos objetos de referência, basta verificar se o objeto de referência está presente no topo de mais de uma lista (linhas 14-17).

Exemplo. Em uma consulta por dois termos t_1 e t_2, um objeto de referência f_1 que tenha t_1 e t_2 na sua descrição textual vai aparecer nas duas listas invertidas: $IF.list(t_1)$ e $IF.list(t_2)$. Como os objetos das listas estão ordenados em ordem crescente de id, f_1 (cujo $id = 1$) vai estar no topo das duas listas. Sendo assim, basta retirar f_1 das duas listas e computar o escore textual de f_1 somando os escores parciais que f_1 tem em cada lista. Caso a descrição textual de f_1 possua apenas o termo t_1, f_1 não estará na lista de t_2 ($IF.list(t_2)$) e o escore textual dele será composto apenas pelo escore parcial obtido através da lista invertida de t_1.

Algoritmo 1: *Inverted File Based Algorithm* (IFA)

```
   Input: Q = (Q.D, Q.r, Q.k) //O critério de vizinhança rng é o raio Q.r.
   Output: Heap contendo os k melhores objetos de interesse.
 1 M ← ∅ //Heap contendo os k melhores objetos de interesse
 2 H ← ∅ //Heap que mantém as entradas e ordenadas pelo id do objeto de referência
 3 for each p ∈ P do
 4 |   p.score ← 0
 5 |   for each t ∈ Q.D do
 6 |   |   iterator ← IF.list(t).iterator()
 7 |   |   if iterator.hasNext() then
 8 |   |   |   H.add({iterator.next(), iterator})
 9 |   |   end
10 |   end
11 |   e ← nextEntry(H)
12 |   while e ≠ null AND H ≠ ∅ do
13 |   |   e' ← nextEntry(H)
14 |   |   while e' ≠ null AND e.f = e'.f do
15 |   |   |   e.f.θ ← e.f.θ + e'.f.θ
16 |   |   |   e' ← nextEntry(H)
17 |   |   end
18 |   |   updateScore(p, e.f)
19 |   |   e ← e'
20 |   end
21 |   updateScore(p, e.f)
22 |   if |M| < k OR p.score > M.peekMin().score then
23 |   |   M.add(p)
24 |   |   if |M| > k then
25 |   |   |   M.removeMin()
26 |   |   end
27 |   end
28 end
29 return M
```

Uma vez que o escore textual do objeto de referência tenha sido computado, utiliza-se a função *updateScore(p, e.f)* para atualizar o escore de p. Esta função atualiza o escore de p se o escore atual de p for menor que o escore textual de f ($p.score < f.\theta$). Após computar o escore de p, o algoritmo atualiza a *heap* M que armazena os melhores objetos (linhas 22-27), adicionando p a M se o tamanho de M ($|M|$) for menor do que k ou o escore de p for maior que o escore do objeto em M com menor escore (linha 22). O algoritmo remove o objeto de M com menor escore caso o tamanho de M supere k (linhas 24-26). O algoritmo repete até que a *heap* H esteja vazia ou $e = null$, retornando os k melhores objetos armazenados em M.

4.1.1 Vizinho mais próximo

Quando o critério de vizinhança é o vizinho mais próximo ($Q.\psi = nn$) é necessário acrescentar antes da linha 4, no Algoritmo 1, a variável *minDistance* que deve ser inicializada com um valor muito alto. Além disto, é necessário substituir *updateScore(p, e.f)* (linhas 18 e 21), por *minDistance =updateScoreNN(p, e.f, minDistance)*. A função *updateScoreNN(p, e.f, minDistance)* atualiza o escore de p quando a distância de p para f for menor do que *minDistance*, ou seja, se $dist(p, f) < minDistance$ então $p.score = f.\theta$. Caso a distância de p para f seja igual a *minDistance* ($dist(p, f) == minDistance$), o escore de p será atualizado apenas se $f.\theta$ for maior do que o escore existente em p (se $f.\theta > p.score$, então $p.score = f.\theta$).

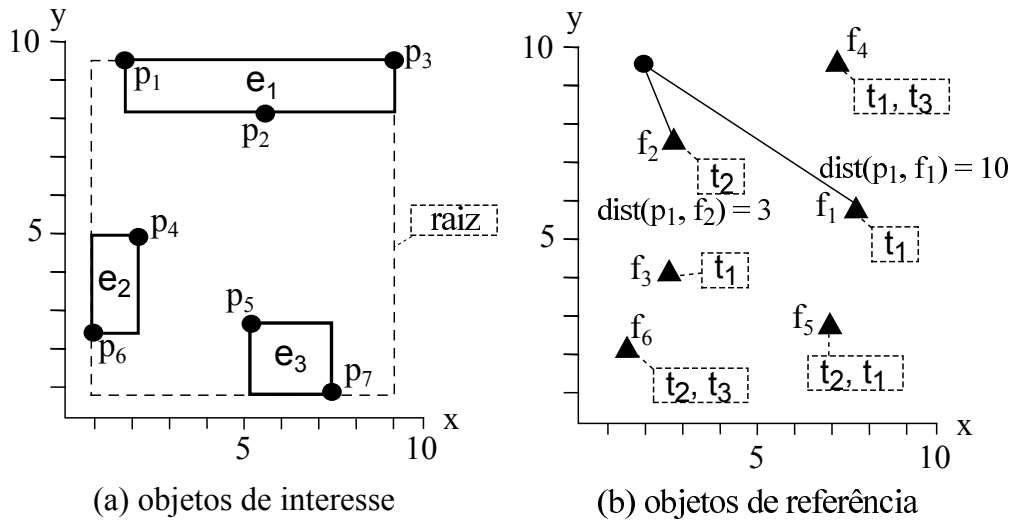


Figura 4.2: Exemplo de seleção do objeto de referência mais relevante para o objeto de interesse utilizando o IFA e o critério de vizinhança vizinho mais próximo.

Exemplo. Dada a Figura 4.2 que apresenta uma R-tree contendo objetos de interesse p (Figura 4.2 (a)) e uma área espacial contendo objetos de referência f (Figura 4.2 (b)), considere dois objetos de referência f_1 e f_2, cuja a distância para o objeto de*

interesse p_1 é 5 e 2, respectivamente. Sabendo que o conjunto de palavras-chave da consulta é formado por t_1 e t_2 ($Q.D = \{t_1, t_2\}$) e que f_1 possui apenas o termo t_1 na sua descrição textual, enquanto f_2 possui apenas o termo t_2. Logo, o objeto f_1 aparece apenas na lista invertida $IF.list(t_1)$ e o objeto f_2 aparece apenas na lista $IF.list(t_2)$. Como ambos objetos possuem apenas um termo, não é necessário somar escores parciais.

A variável minDistance armazena a menor distância de p para f e é inicializada com um valor muito alto ($minDistance = \infty$). Assim, após $IF.list(t_1)$ retornar f_1, este objeto é encaminhado para updateScoreNN(p_1, $e.f_1$, minDistance) que verifica se a distância de p_1 para f_1 é menor do que o valor existente em minDistance. Uma vez que a distância entre p_1 e f_1 é menor do que o valor existente em minDistance, updateScoreNN(p_1, $e.f_1$, minDistance) atualiza o escore de p_1 ($p_1.score = f_1.\theta$) e também atualiza o valor de minDistance ($minDistance = dist(p_1, e.f_1)$, $minDistance = 5$).

Em seguida, $IF.list(t_2)$ retorna o objeto f_2 que é encaminhado para o updateScoreNN(p_1, $e.f_2$, minDistance). O escore de p_1 é atualizado mais uma vez, pois a distância de p_1 para f_2 é menor do que o valor de minDistance ($minDistance = 5$). Logo, minDistance também é atualizado ($minDistance = dist(p_1, e.f_2)$).

4.1.2 Influência

Caso o critério de vizinhança selecionado seja o influência ($Q.\psi = inf$), é necessário modificar o Algoritmo 1. Neste caso, é necessário substituir a função *updateScore(p, e.f)* (linhas 18 e 21) pela função *influenceScore(p, e.f, Q.r)*.

A função *influenceScore(p, e.f, Q.r)* calcula a influência do objeto de referência $e.f$ para o objeto de interesse p. Esta influência é quantificada pela Equação 4.1. Caso o valor de $influenceScore$ seja maior do que o escore do objeto de interesse ($p.score$), o valor de $influenceScore$ é atribuído a $p.score$ ($p.score = influenceScore$).

$$influenceScore = e.f.\theta * 2^{-dist(p,e.f)/Q.r} \tag{4.1}$$

No critério de vizinhança influência, a distância de um objeto referência $e.f$ para o objeto de interesse p é inversamente proporcional ao $influenceScore$ do objeto de referência. Sendo assim, quanto mais distante o objeto de referência estiver do objeto de interesse, menor será o $influenceScore$, como pode ser observado na Equação 4.1. Desta forma, os objetos de referência que possuem relevância textual e estão próximos do objeto de interesse são beneficiados por este critério de vizinhança, sem ignorar objetos de referência que estão mais distantes e também possuem relevância textual.

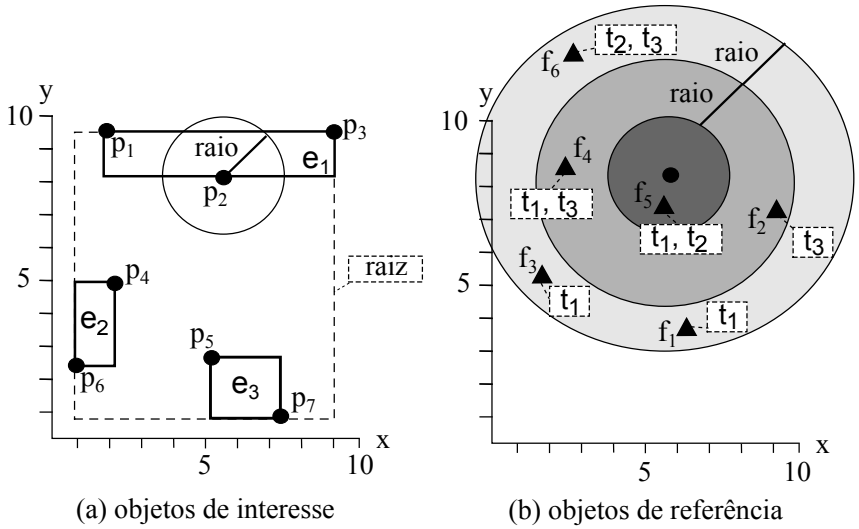


Figura 4.3: Exemplo de seleção do objeto de referência mais relevante na vizinhança espacial do objeto de interesse utilizando o IFA e o critério de vizinhança influência.

Exemplo. Considere uma consulta EPPC que possua dois termos t_1 e t_2 no seu conjunto de palavras-chave ($Q.D = \{t_1, t_2\}$). Dada a Figura 4.3 que apresenta uma R^*-tree onde está indexado o conjunto de objetos de interesse (Figura 4.3 (a)) e outra área espacial contendo objetos de referência f (Figura 4.3 (b)), seleciona-se o objeto de referência f mais relevante para o objeto de interesse p_2.

Sabendo que os objetos existentes nas listas invertidas estão ordenados em ordem crescente de id, quando f_5 (cujo $id = 5$) estiver no topo das duas listas: $IF.list(t_1)$ e $IF.list(t_2)$, basta retirar f_5 das duas listas e computar o escore textual de f_5 somando os escores parciais que f_5 tem em cada lista. Em seguida, o $influenceScore$ de f_5 é calculado na função $influenceScore(p_2, e.f_5, Q.r)$. Caso o valor obtido seja superior ao escore de p_2 ($p_2.score < influenceScore$), $influenceScore(p_2, e.f_5, Q.r)$ atualiza o escore de p_2 ($p_2.score = influenceScore$).

4.2 *Spatially Indexed Algorithm*

O *Spatially Indexed Algorithm* (SIA) utiliza o índice híbrido *Spatial Inverted Index* (S2I) ao invés do Arquivo Invertido Adaptado utilizado no IFA. Portanto, ao invés de obter uma lista invertida com objetos de referência e depois verificar se estes objetos atendem ao critério de vizinhança, o S2I permite selecionar os objetos de referência que são textualmente e espacialmente relevantes em uma única etapa. Ou seja, ao acessar o S2I é retornado um iterador, com o qual são acessados apenas os objetos de referência que atendem ao critério de vizinhança espacial e que possuem relevância textual para o conjunto de palavras-chave. Nesta seção é realizada a descrição do Algoritmo SIA utilizando o critério de vizinhança seleção espacial ($Q.\psi = rng$). Em

seguida são apresentadas as mudanças necessárias para utilizar os demais critérios de vizinhança.

Algoritmo 2: *Spatially Indexed Algorithm* (SIA)

```
   Input: Q = (Q.D, Q.r, Q.k) //O critério de vizinhança Q.τ^rng é o raio Q.r.
   Output: Heap contendo os k melhores objetos de interesse.
 1 M ← ∅ //Heap contendo os k melhores objetos de interesse
 2 H ← ∅ //Heap que mantém as entradas e ordenadas pelo id do objeto de referência
 3 for each p ∈ P do
 4 |   p.score ← 0
 5 |   for each t ∈ Q.D do
 6 |   |   iterator ← S2I.searchRange(t, p.x, p.y, Q.r)
 7 |   |   linhas 7-9 do algoritmo 1 – IFA
 8 |   end
 9 |   linhas 11-27 do algoritmo 1 – IFA
10 end
11 return M
```

Assim como o IFA, o SIA computa o escore de cada objeto de interesse $p \in P$ para obter os k melhores objetos. O Algoritmo 2 começa inicializando o escore de cada objeto p com zero (linha 4). Para cada termo t do conjunto de palavras-chave ($t \in Q.D$), o algoritmo acessa o $S2I$ para recuperar um iterador (linha 6) que acessa objetos de referência relevantes para t, ordenados em ordem crescente de id. Entretanto, ao invés de retornar todos os objetos textualmente relevantes como o IF, o iterador obtido no $S2I$ retorna apenas os objetos f que são textualmente relevantes e cujo a distância para p seja menor ou igual que o raio $Q.r$.

O S2I permite selecionar os objetos f relevantes para os critérios de vizinhança espacial da consulta eficientemente (ex: seleção espacial, $dist(p.x, p.y) \leq Q.r$). O restante do algoritmo SIA é idêntico ao IFA[1].

A Figura 4.4 apresenta uma R*-tree contendo objetos de interesse (Figura 4.4 (a)) e uma aR-tree de um termo t (Figura 4.4 (b)). Todos os objetos de referência representados nesta aR-tree possuem o termo t em sua descrição textual. A seguir é apresentado um exemplo de como identificar um objeto de referência que atende ao critério de vizinhança $Q.\psi = rng$ utilizando essas duas estruturas.

Exemplo. Para identificar qual o objeto de referência que atende ao critério de vizinhança ($Q.\psi = rng$) do objeto de interesse p_1, calcula-se a distância de p_1 para as MBRs de e_1, e_2 e e_3. A MBR de uma entrada é o menor retângulo que envolve os objetos de referência existentes nesta entrada. Caso a distância de p_1 para uma das MBRs seja menor ou igual do que o valor do raio $Q.r$, esta entrada é visitada e a distância entre p_1 e os objetos de referência ($dist(p_1, f)$) existentes nesta entrada são calculadas. Caso esta nova distância também seja menor ou igual do que $Q.r$ ($dist(p_1, f) \leq Q.r$), o objeto de referência é considerado vizinho e textualmente relevante para o objeto de interesse.

[1]No algoritmo SIA o iterador da linha 6 retorna apenas os objetos que atendem ao critério de vizinhança, portanto, o método *updateScore* apenas atualiza o escore do objeto p.

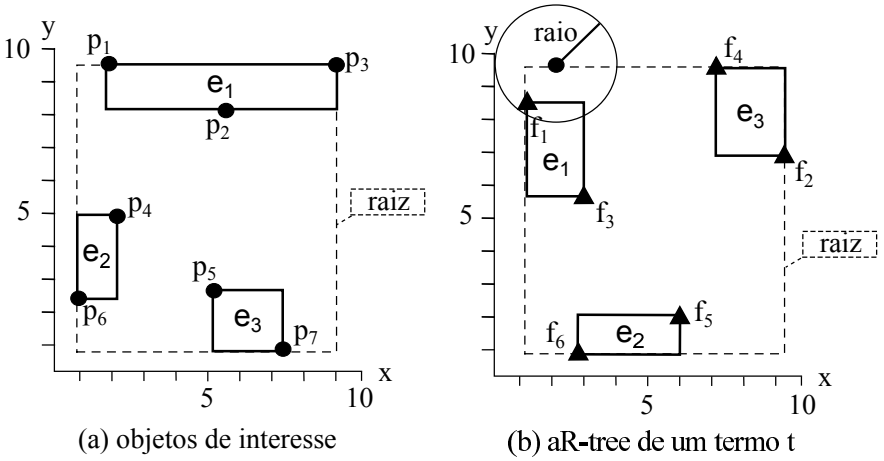


Figura 4.4: Exemplo da seleção dos objetos de referência vizinhos ao objeto de interesse utilizando o SIA e o critério seleção espacial.

Ao computar as distâncias para e_1, e_2 e e_3, conclui-se que apenas a entrada e_1 está próxima o suficiente do objeto p_1 ($dist(p_1, e_1) \leq Q.r$). Isto indica que pelo menos um objeto de referência em e_1 pode ser vizinho de p_1. Para verificar quais são estes objetos, calcula-se a distância de p_1 para f_1, e para f_3. Observa-se na Figura 4.4 (b) que apenas o objeto f_1 é vizinho de p_1, pois este é o único objeto de referência cuja distância para p_1 é menor ou igual do que o raio $Q.r$.

Desta forma, é possível ignorar os objetos existentes nas entradas e_2 e e_3, pois a distância de p_1 para estas entradas é maior do que o raio $Q.r$, evitando o acesso desnecessário a vários objetos de referência.

4.2.1 Vizinho mais próximo

Caso o critério de vizinhança escolhido seja o vizinho mais próximo, é necessário modificar a linha 6 do Algoritmo 2 para $iterator \leftarrow S2I.searchNN(t, p.x, p.y)$. $S2I.searchNN()$ retorna um iterador que acessa o objeto de referência mais próximo espacialmente de p e que possui o termo t. Caso existam vários objetos de referência f com a mesma proximidade espacial do objeto de interesse p, o iterador retorna todos estes objetos.

Também é necessário substituir *updateScore(p, e.f)* por *updateScoreNN(p, e.f)* que atualiza o escore de p quando a distância de p para f for menor do que $minDistance$, entretanto, quando a distância entre p e f for igual a $minDistance$, o escore de p é atualizado se $f.\theta > p.score$.

Exemplo. Dada a Figura 4.5 que apresenta uma R-tree contendo os objetos de interesse p (Figura 4.5 (a)) e uma aR-tree de um termo t ($t \in Q.D$), contendo objetos de referência f que possuem o termo t (Figura 4.5 (b)). Considere que a*

MBR das entradas e_1, e_2 e e_3 da aR-tree possuem uma distância 15, 3 e 10 para o objeto de interesse p_1, respectivamente.

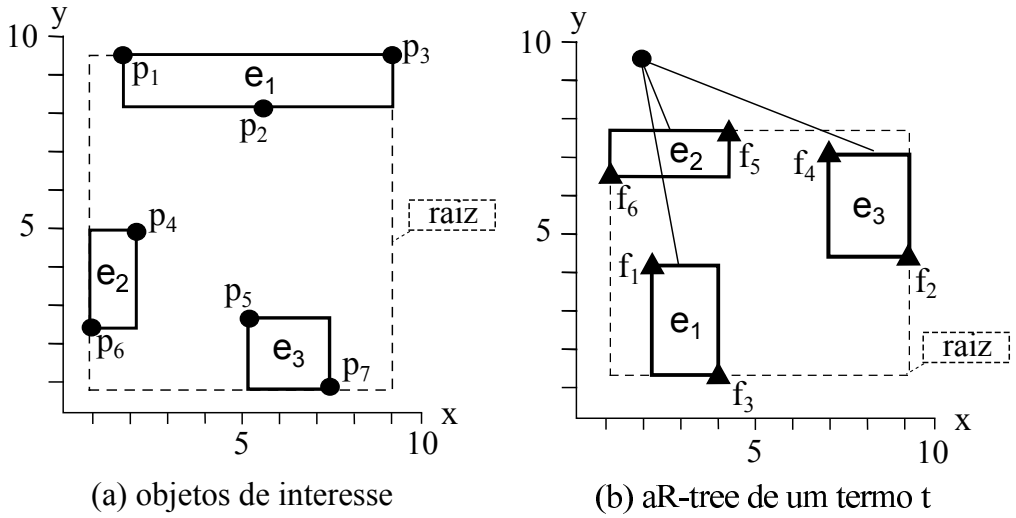


Figura 4.5: Exemplo de seleção do objeto de referência mais próximo do objeto de interesse utilizando o SIA e o critério de vizinhança vizinho mais próximo.

Inicialmente, para encontrar o objeto de referência mais próximo do objeto de interesse p_1, calcula-se a distância da MBR de cada entrada e existente na raiz da aR-tree para p_1 ($dist(e.MBR, p_1)$). Estas entradas são inseridas em uma heap e ordenadas em ordem crescente de distância ($dist(e.MBR, p_1)$) obtidos.

A entrada e_2, que possui a menor distância para p_1, é removida da heap e é acessada. Assim, é calculada a distância de p_1 para cada objeto de referência existente na entrada e_2 (f_5 e f_6). O objeto f_5, que possui a menor distância para p_1, é adicionado a um conjunto C de objetos de referência candidatos a vizinho mais próximo do objeto de interesse p_1. O valor $dist(f_5, p_1)$ é atribuído a uma variável $minDistance$.

Caso existam outros objetos de referência com uma distância $dist(f, p_1)$ igual a de f_5 ($dist(f, p_1) == dist(f_5, p_1)$), estes objetos também são acrescentados ao conjunto C. Caso existisse um objeto de referência f com uma distância para p_1 menor do que a de f_5 ($dist(f_5, p_1) < dist(f, p_1)$), os objetos existentes no conjunto C seriam apagados, e este objeto f seria inserido no conjunto C.

O processo descrito no exemplo anterior é repetido até que não exista nenhuma entrada na *heap*. O iterador (linha 6) retorna os objetos existentes em C ordenados crescentemente pelos seus valores de *id*. O restante do algoritmo é semelhante ao Algoritmo 2 apresentado para o critério seleção espacial.

4.2.2 Influência

Infelizmente, não é vantajoso processar a consulta EPPC utilizando o critério de vizinhança influência e o algoritmo SIA. A otimização do SIA em relação ao IFA

consiste na possibilidade de acessar o S2I e obter os objetos de referência que atendem ao critério de vizinhança espacial e são textualmente relevantes.

Ao acessar uma árvore no S2I, é possível identificar os objetos de referência que não atendem ao critério de vizinhança espacial e não os acessar. Desta forma, o SIA evita acessar muitos dos objetos que são acessados pelo IFA.

Exemplo. No critério de seleção espacial ($Q.\psi = rng$) é possível evitar o acesso a todos os objetos de referência f que possuem uma distância $dist(p, f)$ para o objeto de interesse p maior do que o raio $Q.r$ ($dist(p, f) > Q.r$).

Entretanto, no critério influência não é possível selecionar ou descartar um objeto baseando-se apenas em suas características espaciais. Observa-se na Equação 4.1 que são necessárias informações espaciais e textuais para selecionar um objeto utilizando o critério influência.

Como não é possível fazer esta seleção dos objetos de referência utilizando influência, não faz sentido utilizar o algoritmo SIA, pois para este critério, o SIA não apresenta nenhuma otimização.

4.3 *Optimized Spatially Indexed Algorithm*

O IFA e o SIA computam o escore de cada objeto de interesse e apresentam os k objetos com maiores escores ao usuário. O SIA^+ é uma variação do SIA, que ao invés de computar o escore de cada objeto de interesse separadamente, ele calcula o escore de um conjunto V de objetos de interesse concorrentemente. O objetivo do SIA^+ é reduzir o I/O, percorrendo o S2I apenas uma vez para calcular o escore de todos objetos armazenados em V.

Os objetos de interesse $p \in P$ escolhidos para compor um conjunto V são objetos espacialmente próximos. Por questões de simplicidade, optou-se por computar concorrentemente os objetos p que estão armazenados na mesma folha da R*-tree [Beckmann et al. 1990] que armazena os objetos P[2].

Escolher apenas objetos de interesse próximos entre si para formar o conjunto V facilita a seleção dos objetos de referência que atendem ao critério de vizinhança; pois se um objeto de referência atende ao critério de vizinhança de um objeto de interesse $p \in V$, existe uma grande possibilidade que este mesmo objeto de referência atenda ao critério de vizinhança de outro objeto de interesse em V, uma vez que os objetos em V são próximos espacialmente entre si. Esta abordagem permite que o mesmo objeto de referência seja acessado menos vezes quando comparado ao SIA e ao IFA.

[2]Uma outra solução seria utilizar um algoritmo de *cluster* para agrupar os objetos próximos espacialmente, onde cada V seria composto pelos objetos armazenados no mesmo *cluster*.

O Algoritmo 3 apresenta o *Optimized Spatially Indexed Algorithm* (SIA$^+$). Para cada V, onde V é composto por um conjunto de objetos espacialmente próximos, o algoritmo inicializa todos os escores dos objetos $p \in V$ com zero (linha 4). Este algoritmo acessa o S2I para obter apenas os objetos de referência que podem contribuir com o cálculo do escore de um objeto $p \in V$ (linha 6).

Algoritmo 3: *Optimized Spatially Indexed Algorithm* (SIA+)

```
    Input: Q = (Q.D, Q.r, Q.k) //O critério de vizinhança rng é o raio Q.r.
    Output: Heap contendo os k melhores objetos de interesse.
 1  M ← ∅ //Heap contendo os k melhores objetos de interesse
 2  H ← ∅ //Heap que mantém as entradas e ordenadas pelo id do objeto de referência
 3  for each V ∈ P do
 4  |  ∀p ∈ V, p.score ← 0
 5  |  for each t ∈ Q.D do
 6  |  |  iterator ← S2I.searchRange(t, V.MBR, Q.r)
 7  |  |  linhas 7-9 do algoritmo IFA
 8  |  end
 9  |  e ← nextEntry(H)
10  |  while e ≠ null AND H ≠ ∅ do
11  |  |  e' ← nextEntry(H)
12  |  |  while e' ≠ null AND e.f = e'.f do
13  |  |  |  e.f.θ ← e.f.θ + e'.f.θ
14  |  |  |  e' ← nextEntry(H)
15  |  |  end
16  |  |  updateScore(V, e.f) e ← e'
17  |  end
18  |  updateScore(V, e.f)
19  |  for each p ∈ V do
20  |  |  if |M| < k OR p.score > M.peakMin().score then
21  |  |  |  M.add(p)
22  |  |  |  if |M| > k then
23  |  |  |  |  M.removeMin()
24  |  |  |  end
25  |  |  end
26  |  end
27  end
28  return M
```

Todo objeto de referência f cuja distância para o $V.MBR$[3] seja inferior ou igual a $Q.r$ ($dist(f, V.MBR) \leq Q.r$) pode colaborar com o escore de um objeto $p \in V$. Uma vez que este objeto f pode atender ao critério de vizinhança de pelo menos um $p \in V$.

As linhas 9-18 são idênticas ao IFA e ao SIA, exceto as linhas 16 e 18, que ao invés de computar o escore de um único objeto p, o SIA$^+$ computa o escore de todos os objetos $p \in V$. O mesmo ocorre nas linhas 19-26, onde o SIA$^+$ precisa verificar, para cada $p \in V$, se p pode ser inserido na *heap* M que mantém os k melhores objetos.

[3]Retângulo mínimo que engloba todos os objetos espaciais $p \in V$.

Exemplo. Dada a Figura 4.6, composta por um conjunto de objetos de interesse indexados em uma R-tree (Figura 4.6 (a)) e uma aR-tree de um termo t, contendo os objetos de referência que possuem o termo t (Figura 4.6 (b)). Considere que V é formado pelos objetos da entrada e_1 existente na Figura 4.6 (a).*

Para identificar quais objetos de referência podem contribuir com o escore de pelo menos um objeto $p \in V$ é necessário calcular a distância da MBR que envolve os objetos em V (V.MBR) para as MBRs de todas as entradas existentes na aR-tree da Figura 4.6 (b) (dist(V.MBR, e.MBR).

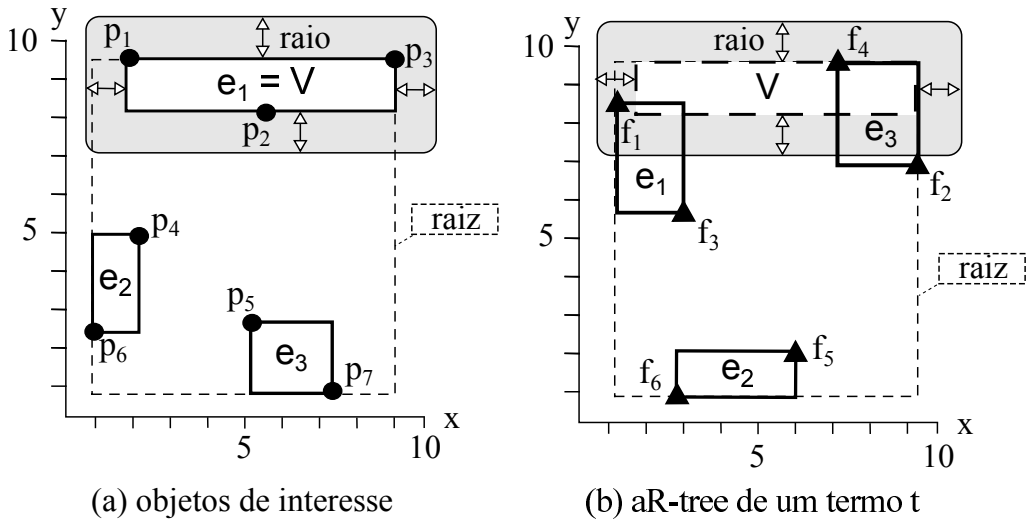


Figura 4.6: Exemplo da seleção dos objetos de referência vizinhos ao objeto de interesse utilizando o SIA$^+$ e o critério de seleção espacial.

Assim, é calculado a distância de V para e_1, e_2 e e_3. É possível observar na Figura 4.6 (b) que apenas as entradas e_1 e e_3 possuem uma distância para V menor ou igual do que o raio Q.r, atendendo ao critério de vizinhança seleção espacial. Portanto, apenas estas entradas são visitadas, evitando o acesso aos objetos de referência existentes na entrada e_2, pois estes objetos não atendem ao critério de vizinhança de nenhum $p \in V$.

As entradas e_1 e e_3 são acessadas, e a distância entre V.MBR e cada objeto de referência existente nestas entradas é calculada. Os objetos de referência que possuem uma distância para V.MBR menor ou igual do que o raio Q.r (f_1 e f_4), são retornados pelo iterador (linha 6) em ordem crescente de id.

4.3.1 Vizinho mais próximo

Quando for selecionado o critério de vizinhança vizinho mais próximo ($Q.\psi = nn$), o S2I é percorrido apenas uma vez e é construído um conjunto C de objetos de referência candidatos a serem os vizinhos mais próximos de pelo menos um $p \in V$.

Para isso, o método $S2I.searchRange$ é substituído por $S2I.searchNN(t, V.MBR)$ (linha 6), que recebe um termo t e a MBR do conjunto V como parâmetros. O método $S2I.searchNN()$ acessa o S2I e retorna o conjunto C.

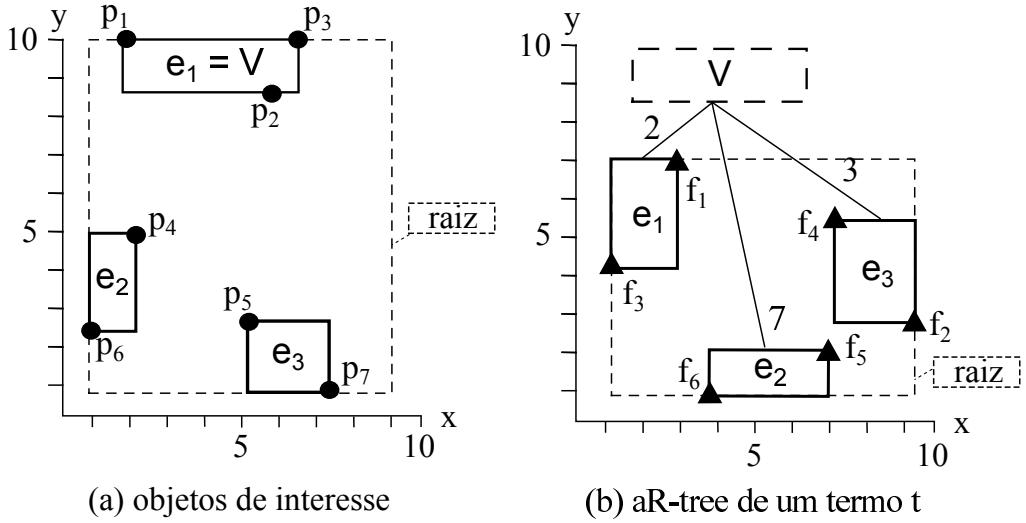


Figura 4.7: Exemplo da seleção dos objetos de referência utilizando o SIA$^+$ e o critério vizinho mais próximo.

Como cada aR-tree corresponde a um termo t, ou seja, cada aR-tree armazena todos os objetos que possuem pelo menos um termo t em sua descrição textual. Para identificar qual é o objeto de referência mais próximo de um objeto de interesse é necessário acessar toda aR-tree que corresponde a um termo da consulta. Pois pode existir um objeto de referência que possui apenas um termo da consulta em sua descrição textual e este ser o mais próximo do objeto de interesse. Por este motivo é que ao acessar uma aR-tree, os objetos retornados são apenas candidatos a vizinho mais próximo.

Além disto, é possível que vários objetos de referência possuam a mesma distância para o objeto de interesse. Se esta distância for a menor a distância encontrada, apenas o objeto de referência que possuir o maior escore textual será considerado o vizinho mais próximo do objeto de interesse.

Inicialmente, para identificar qual é o objeto de referência mais próximo do objeto de interesse é necessário ordenar as entradas existentes na raiz da aR-tree que possui um termo t da consulta. É calculada a distância entre cada entrada existente na raiz para a MBR de V, e estas entradas são ordenadas em ordem crescente de distância.

Exemplo. Dada a Figura 4.7, composta por um conjunto de objetos de interesse indexados em uma R-tree (Figura 4.7 (a)) e uma aR-tree de um termo t, contendo os objetos de referência que possuem o termo t (Figura 4.7 (b)). Para identificar quais objetos de referência podem ser o vizinho mais próximo de um objeto de interesse $p \in V$, é necessário acessar a raiz da aR-tree e calcular a distância entre V e as entradas e_1, e_2 e e_3 (Figura 4.7 (b)). Estas entradas são inseridas em uma*

heap em ordem crescente de distância (dist$(e.MBR, V.MBR)$). Caso estas entradas possuam nós filhos associados, as entradas destes nós filhos também podem ser inseridas na heap.

Em seguida, a entrada mais próxima de V é removida da *heap* e os objetos existentes nesta entrada são acessados. É calculada a distância entre os objetos obtidos desta entrada e os objetos de interesse $p \in V$. O menor valor de distância obtida para cada objeto de interesse é armazenado em uma variável para fazer o controle de quão próximo os objetos de referência desta entrada estão de cada $p \in V$.

Exemplo. A entrada e_1, que possui a menor distância para $V.MBR$, é removida da heap e é acessada. Ao acessar e_1, obtemos os objetos de referência f_1 e f_3. Para cada $p \in V$ existe uma variável $p.minDistance$ que armazena o valor da menor distância $dist(p, f)$ obtida. Cada variável $p.minDistance$ é inicializada com um valor alto[4]*. Sendo assim, é calculada a distância de cada $p \in V$ para cada objeto de referência obtido através da entrada e_1.*

A cada objeto $p \in V$ está associada uma variável $p.mindDistance$ e um conjunto de objetos de referência que possuem a mesma menor distância para p ($p.C'$). Ao acessar os objetos de referência existentes em uma entrada, se uma distância $dist(p, f)$ é menor do que $p.minDistance$, o valor de $p.minDistance$ é atualizado ($p.minDistance = dist(p, f)$), o conjunto $p.C'$ é apagado, e o objeto de referência f é inserido no conjunto $p.C'$. Caso a distância $dist(p, f)$ seja igual ao valor $p.minDistance$, é necessário apenas inserir o objeto de referência em $p.C'$. Isto garante que objetos de referência que possuam a mesma menor distância para o objeto p sejam armazenados em $p.C'$.

Após a distância de cada $p \in V$ para cada objeto existente na entrada for calculada, os valores $p.minDistance$ são ordenados, e a maior distância obtida é atribuída a uma variável $maxDistance$. Esta variável garante que qualquer outra entrada da aR-tree que possua uma distância $dist(V.MBR, e.MBR)$ maior do que $maxDistance$ pode ser ignorada, visto que os objetos de referência existentes nesta entrada não serão vizinho mais próximo de nenhum objeto $p \in V$.

Exemplo. Observa-se na Figura 4.7 (b) que f_1 é o objeto de e_1 mais próximo de V. Então, os valores $p.minDistance$ de cada $p \in V$ é a distancia entre o objeto f_1 e os objetos p_1, p_2 e p_3 que correspondem a 3, 2 e 4, respectivamente. Ordenando os valores $p.minDistance$ de cada $p \in V$ obtém-se o $maxDistance$ equivalente a 4. Sendo assim, o $maxDistance$ equivale a maior distância obtida de f_1 para um objeto de interesse $p \in V$.

É possível utilizar o valor de $maxDistance$ para evitar o acesso a entradas mais distantes do que aquelas já visitadas. Por exemplo, o valor de $maxDistance$ ($maxDistance = 4$) garante que a distância de f_1 para qualquer $p \in V$ é menor do que a distância de e_2 para V ($dist(V.MBR, e_2.MBR)$), pois $maxDistance < dist(V.MBR, e_2.MBR)$. Portanto, e_2 pode ser ignorada, uma vez que os objetos

[4]Esta inicialização é executada uma vez antes de percorrer uma aR-tree pela primeira vez.

de referência existentes nesta entrada (f_5 e f_6) não podem ser mais próximos de nenhum $p \in V$ do que f_1.

Após percorrer todas as entradas da aR-tree e verificar quais objetos de referência podem ser o vizinho mais próximo de cada $p \in V$, os objetos existentes no conjunto $p.C'$ de cada objeto p é inserido no conjunto C. O conjunto C contém todos os objetos de referência candidatos a objeto mais próximo de um objeto $p \in V$, sem repetição.

Os objetos do conjunto C são retornados pelo iterador (linha 6) em ordem crescente de id. O restante do algoritmo é idêntico ao Algoritmo 3, com exceção do $updateScore(V, e.f)$ (linhas 16 e 18) que é substituído por $updateScoreNN(V, e.f)$. A função $updateScoreNN(V, e.f)$ calcula a distância de cada $p \in V$ para $e.f$. Caso a distância obtida seja menor do que $p.minDistance$, o escore de p é atualizado ($p.score = e.f.\theta$) e o $p.minDistance$ também ($p.minDistance = dist(p, e.f)$).

4.3.2 Influência

Assim como demonstrado na Seção 4.2.2, não faz sentido utilizar o SIA$^+$ para processar a consulta EPPC utilizando o critério de vizinhança influência ($Q.\psi = inf$).

Capítulo 5

Avaliação Experimental

"Não é o mais forte que sobrevive, nem o mais inteligente, mas o que melhor se adapta às mudanças."
– Leon C. Megginson

Este capítulo faz uma avaliação dos três novos algoritmos propostos (IFA, SIA e SIA$^+$) para processar a consulta EPPC. Todos os algoritmos foram implementados em Java, utilizando a biblioteca XXL[1]. Todos os experimentos descritos nesta seção foram executados em um computador com processador Intel: 2.0 GHz e 4 GB de memória.

Cada experimento avalia o impacto de uma única variável, enquanto as outras são mantidas fixas. Em cada experimento, o tempo de resposta e o I/O (*page fault*) são coletados. Todos os algoritmos foram implementados para ler páginas de 4MB e não utilizam cache de páginas em memória. Além disto, o tempo de resposta é medido em milissegundos.

Para cada experimento foram realizadas 50 consultas EPPC. Em acréscimo, cada experimento é repetido 10 vezes para garantir a fidelidade dos resultados. Por exemplo, um experimento que varie a quantidade k de objetos de interesse apresentados ao usuário pode realizar 50 consultas com k igual a 1, e depois 50 consultas com k igual a 5. Feito isto, estas 50 consultas com $k = 1$ e $k = 5$ é repetida mais 9 vezes, totalizando 10 execuções. As palavras-chaves das consultas são obtidas coletando termos aleatoriamente e sem repetição entre os 500 termos mais frequentes de cada base de dados. O raio para todas as consultas foi fixado em aproximadamente 200m. A função utilizada para calcular a distância não leva em consideração a inclinação do planeta terra. Esta mesma função foi utilizada em todos os algoritmos, não interferindo nos resultados obtidos.

[1] http://dbs.mathematik.uni-marburg.de/Home/Research/Projects/XXL

Parâmetros	Valores
Número de resultados (k)	1, **5**, 10, 15
Palavras-chave	1, **3**, 5, 7
Conjunto de dados	Veneza, **Londres**, América do Norte

Tabela 5.1: Parâmetros utilizados nos experimentos com os valores padrões destacados em negrito.

A Tabela 5.1 apresenta as variáveis estudadas. Os valores em negrito são os valores utilizados como padrão, quando não explicitamente mencionados. As figuras desta seção são apresentadas em escala logarítmica devido a grande diferença entre os resultados apresentados pelos algoritmos.

Este capítulo é organizado da seguinte forma: a Seção 5.1 apresenta as bases de dados utilizadas nos experimentos. Em seguida são descritos os experimentos realizados para cada critério de vizinhança espacial. Para cada critério, é realizada uma discussão envolvendo o I/O e o tempo de resposta obtidos nos experimentos.

5.1 Bases de dados

Durante a pesquisa, foram identificadas três possíveis bases de dados que podem ser utilizadas durante a avaliação experimental: Wikipedia, OpenStreetMap e o Twitter. A Wikipedia e o OpenStreetMap oferecem suas bases para download[2] gratuitamente. O Twitter possui uma base de dados pública que pode ser acessada através da *Twitter Search API* [Twitter 2014a] ou por uma de suas *streamming API*[Twitter 2014b].

O OpenStreetMap é um mapa colaborativo que contém pontos do mundo inteiro. A OpenStreetMap Foundation disponibiliza[3] todos os seus dados para download e os atualiza diariamente.

As bases de dados espaço-textuais utilizadas nos experimentos apresentados neste capítulo foram obtidas através do Mapzen[4] no dia 22/05/2015 e do GEOFABRIK[5]. Estas organizações mantém *extracts* (recorte da bases de dados total) de diversas regiões do planeta, coletados do OpenStreetMap (http://www.osm.org).

A partir da base de dados global obtida do OpenStreetMap, o Mapzen (www.mapzen.com) constrói *extracts* contendo objetos espaciais de várias cidades. Um *extract* é um conjunto de todos os objetos espaciais existentes em uma cidade

[2]Bases de dados disponíveis para download em http://en.wikipedia.org/wiki/Wikipedia:Database_download e http://planet.openstreetmap.org/

[3]Disponível em: http://wiki.openstreetmap.org/wiki/Planet.osm

[4]http://mapzen.com/metro-extracts

[5]http://download.geofabrik.de/

e que estão registrados no OpenStreetMap. Entre as cidades disponíveis no Mapzen, foram selecionadas as cidades de Veneza e Londres, devido ao tamanho de seus *extracts*. Enquanto do GEOFABRIK foi extraído a base de dados referente ao subcontinente da América do Norte.

O programa *Travel Assistant* desenvolvido pelo projeto PerTA[6] (Personal Travel Assistant) é utilizado para organizar as bases de dados obtidas. Após a execução do programa, são preservados da base de dados original as seguintes informações para cada objeto espacial: a latitude e longitude do objeto espacial, a categoria e subcategoria de objetos na qual este objeto foi classificado, e a sua descrição textual, como representado na Figura 5.1.

A Figura 5.1 exemplifica como um objeto é representado na base de dados OSM (formato OSM XML), e como ele é representado após a processamento realizado pelo *Travel Assistant*. Informações como *node id, timestamp e version* não são utilizadas pela consulta EPPC e por isso são descartadas. Ao final do processo obtém-se uma linha de texto que representa um objeto espacial na base de dados.

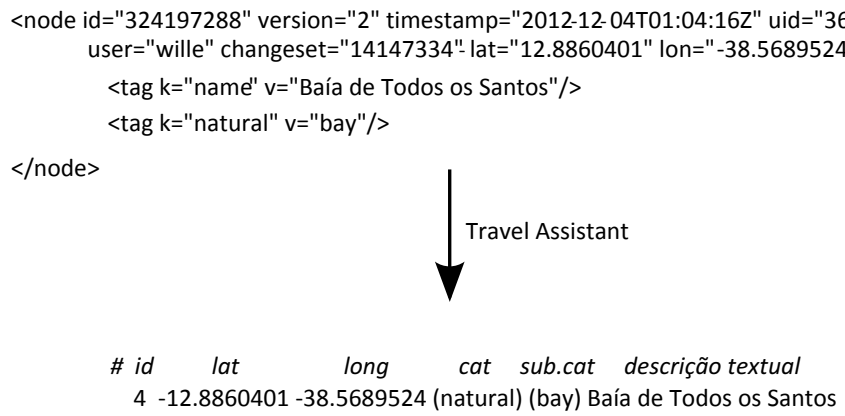


Figura 5.1: Exemplo de objeto espaço-textual após ser processado pelo *parser*.

Após o processamento realizado pelo *Travel Assistant*, cada uma das bases teve seus dados separados em um conjunto de objetos de interesse P e em um conjunto de objetos de referência F. Os objetos $p \in P$ são todos os objetos espaço-textuais cujo a categoria no OpenStreetMap é *hotel*. O conjunto de objetos espaço-textuais de referência $f \in F$ é formado por todos objetos, da base de dados escolhida (ex: São Paulo), que não pertencem à categoria *hotel*. A classificação de um objeto como *hotel* é feita pelos usuários do OpenStreetMap ao adicionarem um novo objeto no mapa. Os usuários indicam a que categoria um objeto pertence.

Além de *hotel*, existem diversas outras categorias que poderiam ser selecionadas (ex: bar, hospital, park). Entretanto, apenas *hotel* apresentou um tamanho médio em todas as bases de dados selecionadas. Além disto, esta categoria de objetos é um foco de diversas aplicações práticas, como o Booking (http://www.booking.com)

[6] http://sites.ecomp.uefs.br/perta

Tabela 5.2: Características das bases de dados.

Bases de dados	$\|P\|$	$\|F\|$	**#termos únicos**	**#total de termos**
Veneza	504	167958	14678	408747
Londres	1341	463066	56569	1198649
América do Norte	9132	2521344	187179	8881870

A Tabela 5.2 apresenta as características de cada uma das bases de dados: número de objetos de interesse $|P|$; número de objetos de referência, $|F|$; número de termos únicos (#termos únicos) que aparecem no vocabulário (coleção) e o número total de termos (#número total de termos). Estas bases de dados podem ser acessadas em: https://goo.gl/zHExTn.

5.2 Seleção espacial

Nesta seção são apresentados os resultados obtidos nos experimentos cujo critério de vizinhança utilizado foi o seleção espacial ($Q.\psi = rng$). Inicialmente é apresentado experimento que varia a quantidade k de objetos retornados pela consulta. Em seguida é apresentado o experimento que varia a quantidade de palavras-chave da consulta e o experimento que varia o tamanho da base de dados. Por fim, é feita uma variação no tamanho do grupo de objetos de interesse que o SIA$^+$ ranqueia concorrentemente.

5.2.1 Variando k

Neste experimento pretende-se estudar o impacto da variação do valor de k nos três algoritmos propostos. Para avaliar este impacto são coletados o tempo de resposta e o I/O de cada experimento. Nas Tabelas 5.3 e 5.4 são apresentados o número de páginas lidas e os tempos de resposta obtidos enquanto foi realizada a variação da quantidade (k) de objetos retornados pela consulta. A primeira coluna de cada uma destas tabelas listam os valores utilizados para k, e as demais colunas apresentam a quantidade de páginas lidas (Tabela 5.3) ou o tempo de resposta, em milissegundos, (Tabela 5.4) para cada um dos algoritmos propostos.

Observa-se a quantidade de páginas lidas na Tabela 5.3, na qual o SIA e o SIA$^+$ acessam menos páginas do que o IFA. Isto demonstra a importância em utilizar um índice híbrido para diminuir o acesso a objetos que não atendem ao critério de vizinhança espacial. O SIA$^+$ apresenta os melhores resultados, visto que ele acessa o S2I uma única vez para computar o escore de um conjunto de objetos de interesse concorrentemente.

Tabela 5.3: Quantidade de páginas lidas ao variar a quantidade de resultados (k) utilizando seleção espacial ($Q.\psi = rng$).

Resultados (k)	**IFA**	**SIA**	**SIA$^+$**
1	84 262,356	12 741,754	283,308
5	84 262,356	12 741,754	283,308
10	84 262,356	12 741,754	283,308
15	84 262,356	12 741,754	283,308

Tabela 5.4: Tempo de resposta (ms) ao variar a quantidade de resultados (k) utilizando seleção espacial ($Q.\psi = rng$).

Resultados (k)	**IFA**	**SIA**	**SIA$^+$**
1	2 579,874	203,975	74,991
5	2 580,884	204,902	75,438
10	2 579,988	205,081	75,458
15	2 577,640	205,176	76,596

É possível observar o tempo de resposta obtido ao variar o valor de k na Tabela 5.4. Neste experimento, o SIA$^+$ obteve um tempo de resposta melhor do que o IFA e o SIA. A diferença entre o SIA o SIA$^+$ que ocorre em número de páginas lidas não se traduz plenamente no tempo de resposta. A principal razão é que o SIA$^+$ requer maior processamento para processar dados concorrentemente, visto que, neste algoritmo, o S2I retorna uma maior quantidade de objetos de referência, fazendo com que mais cálculos de distância sejam executados.

Uma vez que para qualquer valor de k, o escore de cada objeto de interesse em P precisa ser calculado, é esperado que a quantidade de páginas lidas e o tempo de resposta se mantenham estáveis independentemente do valor de k escolhido.

Pode-se observar nas Tabelas 5.3 e 5.4 que existe pouco custo em tempo de resposta ou I/O para exibir mais objetos de interesse ao usuário. Ao aumentar o valor de k, são exibidos mais objetos ao usuário; e para isto é necessário apenas ter uma *heap* maior e fazer mais verificações quando for inserir um novo elemento nesta *heap* (como demonstrado no Capítulo 1).

Examinando a Tabela 5.3, é possível concluir que o desvio padrão populacional das amostras obtidas para cada um dos três algoritmos é zero. Pois não é necessário acessar mais páginas de disco quando o valor de k é variado. Por outro lado, o tempo de resposta (Tabela 5.4) apresenta uma pequena variação decorrente das comparações realizadas na *heap*.

A Tabela 5.5 apresenta o desvio padrão populacional de cada um dos três algoritmos propostos. Em cada coluna é apresentado o desvio padrão referente à variação do tempo de resposta ao variar a quantidade de resultados (k) apresentados para o usuário em um determinado algoritmo. Nota-se que o tempo de resposta varia pouco

Tabela 5.5: Desvio padrão do tempo de resposta obtido ao variar a quantidade de resultados (k) utilizando os três algoritmos propostos.

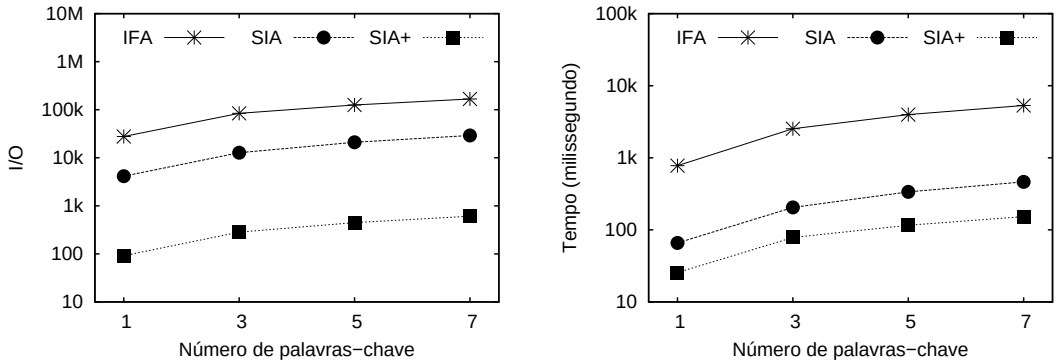
IFA	**SIA**	**SIA$^+$**
1,195	0,477	0,593

nos três algoritmos propostos. Sendo assim, para apresentar um objeto ou 15 objetos de interesse como resposta para o usuário implica em um custo de processamento pequeno, pois a variação no tempo de resposta da consulta é pequeno; com destaque para o SIA e o SIA$^+$ que obtiveram um desvio padrão menor do que o IFA.

5.2.2 Variando o número de palavras-chave

Neste experimento pretende-se estudar o impacto da quantidade de palavras-chave em uma consulta EPPC utilizando os três algoritmos propostos. Para avaliar este impacto são coletados o tempo de resposta e o I/O de cada experimento. A Figura 5.2 apresenta o número de páginas lidas e o tempo de resposta obtidos ao variar o número de palavras-chave da consulta.

Todos os algoritmos são impactados por essa variável, visto que aumentar um termo significa acessar mais dados no momento da consulta. Entretanto, pode-se observar na Figura 5.2(a) que existe uma diferença clara em favor do SIA$^+$, que acessa 2 ordens de magnitude menos páginas que o IFA.



(a) Páginas lidas.

(b) Tempo de resposta.

Figura 5.2: Quantidade de páginas lidas e tempo de resposta ao variar a quantidade de palavras-chave utilizando seleção espacial ($Q.\psi = rng$).

Como pode ser visto na Figura 5.2(b), o SIA e o SIA$^+$ apresentam tempos de resposta próximos; enquanto o IFA é uma ordem de magnitude mais lento que os dois algoritmos. A diferença na quantidade de páginas lidas entre o SIA e o SIA$^+$ foi suficiente para o SIA$^+$ obter um tempo de resposta um pouco melhor. O que demonstra a importância em adotar técnicas para reduzir também o processamento e não apenas o I/O caso o objetivo principal seja otimizar o tempo de resposta.

As Tabelas 5.6(a) e 5.6(b) apresentam o desvio padrão populacional do tempo de resposta e da quantidade de páginas lidas ao variar a quantidade de palavras-chave. Observa-se na Tabela 5.6(a) que o algoritmo SIA$^+$ além de acessar menos páginas que os demais algoritmos, também exige que menos páginas sejam acessadas caso o número de palavras-chave seja incrementado.

Tabela 5.6: Desvio padrão do tempo de resposta e da quantidade de páginas lidas ao variar a quantidade de palavras-chave utilizando seleção espacial ($Q.\psi = rng$).

(a) Páginas lidas.

IFA	SIA	SIA$^+$
51 761,750	9 317,990	191,861

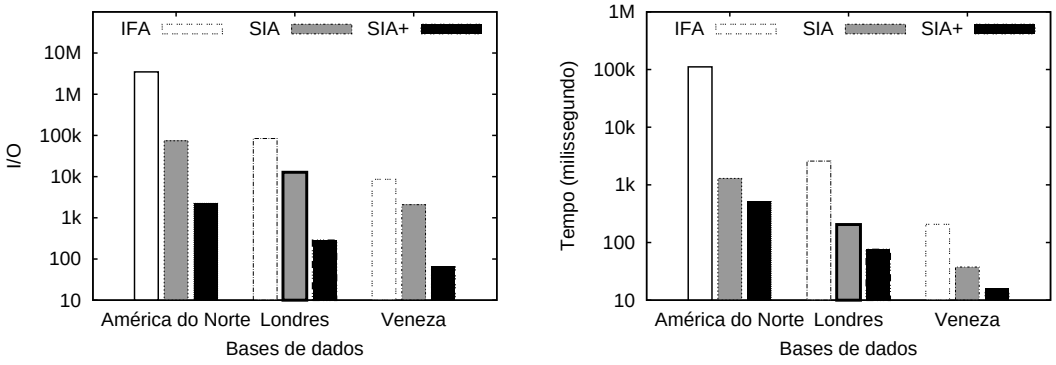
(b) Tempo de resposta.

IFA	SIA	SIA$^+$
1 686,856	148,023	47,075

Os valores apresentados na Tabela 5.6(b) mostram que o SIA$^+$ é algoritmo que possui menor variação no tempo de resposta entre os algoritmos propostos. Aumentar a quantidade de palavras-chave aumenta o custo para processar a consulta EPPC. O desvio padrão apresentado na Tabela 5.6(b) indica que, entre os algoritmos propostos, o algoritmo SIA$^+$ é o que exige menos custo computacional para processar a consulta utilizando uma quantidade maior de palavras-chave.

5.2.3 Variando o tamanho da base de dados

Neste outro experimento, a Figura 5.3 apresenta o tempo de resposta e a quantidade de páginas lidas durante o processamento da consulta em três bases de dados de tamanhos diferentes.



(a) Páginas lidas. (b) Tempo de resposta.

Figura 5.3: Desvio padrão ao variar o tamanho da base de dados utilizando seleção espacial ($Q.\psi = rng$).

A Figura 5.3(a) apresenta o I/O para as três bases de dados. Comprovando que os resultados são consistentes mesmo em bases de dados diferentes e de diferentes

tamanhos. O gráfico apresentado na Figura 5.3(b) reforça a eficiência do SIA e SIA$^+$ em reduzir o tempo de resposta. Demonstrando que quanto maior a base, maior o ganho destas abordagens.

As Tabelas 5.7(a) e 5.7(b) apresentam o desvio padrão populacional do tempo de resposta e da quantidade de páginas lidas ao variar o tamanho da base de dados. Aumentar o tamanho da base de dados implica em acessar mais páginas de disco para processar a consulta EPPC. Observa-se na Tabela 5.7(a) que quando o tamanho da base de dados aumenta o algoritmo IFA necessita acessar mais páginas de disco do que os outros algoritmos propostos. O desvio padrão mostra também que, entre os algoritmos propostos, o SIA$^+$ é o algoritmo que a quantidade de páginas lidas varia menos ao aumentar o tamanho da base de dados.

Tabela 5.7: Desvio padrão da quantidade de páginas lidas e do tempo de resposta ao variar o tamanho da base de dados (k) utilizando seleção espacial ($Q.\psi = rng$).

(a) Páginas lidas.

IFA	**SIA**	**SIA$^+$**
1 624 498,921	31 756,560	977,827

(b) Tempo de resposta.

IFA	**SIA**	**SIA$^+$**
51 774,799	556,313	221,558

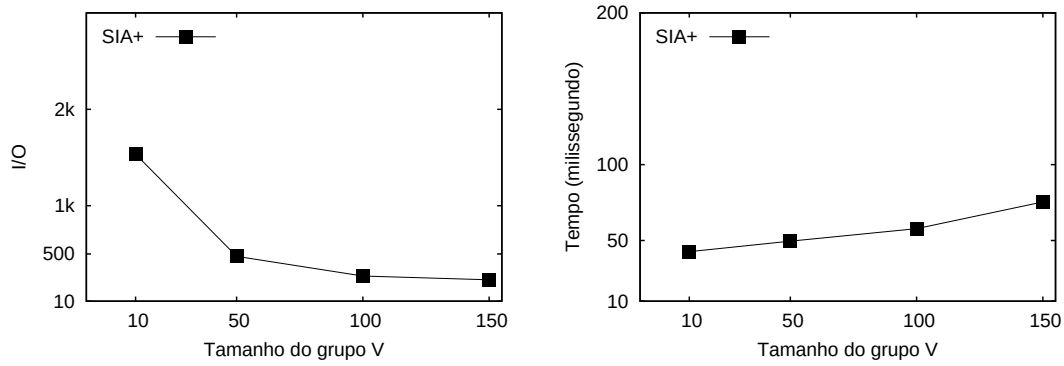
Nota-se na Tabela 5.7(b) que o desvio padrão do tempo de resposta para o algoritmo SIA$^+$ também é o menor entre os algoritmos apresentados. Desta forma, o SIA$^+$ é algoritmo que apresentou menor variação no custo para processar a consulta EPPC ao variar o tamanho da base de dados.

5.2.4 Variando o tamanho dos grupos (S2I+)

No SIA$^+$ os objetos de interesse são inseridos em um grupo V e o escore dos objetos deste grupo é calculado concorrentemente. O tamanho do grupo V tem impacto no I/O e no tempo de resposta. Um grupo V pequeno reduz o I/O, visto que uma menor parte do índice híbrido será acessado, entretanto mais consultas serão feitas ao S2I para computar o escore de todos os objetos de interesse $p \in P$. Este experimento estuda o impacto do tamanho desse grupo no I/O e no tempo de resposta da consulta.

A Figura 5.4 apresenta esse experimento no qual é variado o tamanho do grupo V e é coletado o tempo de resposta e a quantidade de páginas lidas. Neste experimento, o tamanho do grupo V foi variado para armazenar no máximo 10, 50, 100 ou 150 objetos de interesse. Os gráficos apresentados nesta subseção não utilizam escala logarítmica devido a pequena diferença encontrada no tempo de resposta.

Observa-se na Figura 5.4(a) que aumentar o tamanho do grupo V resulta em menos páginas acessadas para processar a consulta. Isto ocorre, pois, utilizar um grupo V grande implica em acessar o S2I poucas vezes para identificar quais objetos de referência possuem um termo t e atendem ao critério de vizinhança de todos os objetos de interesse.



(a) Páginas lidas. (b) Tempo de resposta.

Figura 5.4: Quantidade de páginas lidas e tempo de resposta ao variar o tamanho do grupo V utilizando seleção espacial ($Q.\psi = rng$).

A Figura 5.4(b) mostra que o aumento do grupo V resultou em um aumento no tempo de resposta; pois quanto maior o grupo V maior é a quantidade de objetos retornados pelo S2I, resultando em mais verificações e consequentemente maior custo computacional para computar o escore dos objetos de interesse em V. Assim, existe uma relação inversa entre a quantidade de páginas lidas e o tempo de resposta do algoritmo SIA$^+$. Uma vez que um grupo V grande, proporciona menor quantidade de páginas lidas e também um aumento no custo do cálculo do escore.

Nos demais experimentos, o tamanho do grupo V é definido em 102. Ou seja, o grupo V é capaz de armazenar no máximo 102 objetos de interesse. Este valor foi escolhido pois 102 objetos de interesse é a quantidade máxima de objetos que podem ser armazenados em uma página de disco de tamanho 4KB.

5.3 Vizinho mais próximo

Nesta seção são apresentados os resultados obtidos nos experimentos cujo critério de vizinhança utilizado foi o vizinho mais próximo. Esta seção é dividida em subseções, onde cada seção apresenta o I/O e o tempo de resposta de um experimento específico. Inicialmente é apresentado experimento que varia a quantidade k de objetos retornados pela consulta. Em seguida é apresentado o experimento que varia a quantidade de palavras-chave da consulta e o experimento que varia o tamanho da base de dados. Por fim, é feita uma variação no tamanho do grupo de objetos de interesse que o SIA$^+$ ranqueia concorrentemente.

5.3.1 Variando k

Neste experimento pretende-se estudar o impacto da variação do valor de k nos três algoritmos propostos. Para avaliar este impacto são coletados o tempo de resposta e o I/O de cada experimento. As Tabelas 5.8 e 5.9 apresentam os resultados obtidos ao variar o valor de k. A primeira coluna de cada uma destas tabelas listam os valores utilizados para k, e nas demais colunas são apresentadas a quantidade de páginas lidas (Tabela 5.8) ou o tempo de resposta, em milissegundos, (Tabela 5.9) para cada um dos algoritmos propostos.

Tabela 5.8: Quantidade de páginas lidas ao variar a quantidade de resultados (k) utilizando vizinho mais próximo ($Q.\psi = nn$).

Resultados (k)	**IFA**	**SIA**	**SIA$^+$**
1	84262.356	14519.460	284.346
5	84262.356	14519.459	284.346
10	84262.356	14519.460	284.346
15	84262.356	14519.459	284.345

Tabela 5.9: Tempo de resposta (ms) ao variar a quantidade de resultados (k) utilizando vizinho mais próximo ($Q.\psi = nn$).

Resultados (k)	**IFA**	**SIA**	**SIA$^+$**
1	2634.551	315.368	105.607
5	2629.568	315.504	105.650
10	2631.100	315.19	106.130
15	2647.022	360.892	112.840

Quando o critério de vizinhança é o vizinho mais próximo, é possível evitar o acesso a objetos que não estão próximos do objeto de interesse, como demonstrado nos algoritmos apresentados no Capítulo 4. Evitar o acesso a nós da aR-tree resultou em bons resultados para o SIA e o SIA$^+$, com destaque para o SIA$^+$ que obteve 2 ordens de magnitude a menos de páginas lidas no experimento apresentado na Tabela 5.8.

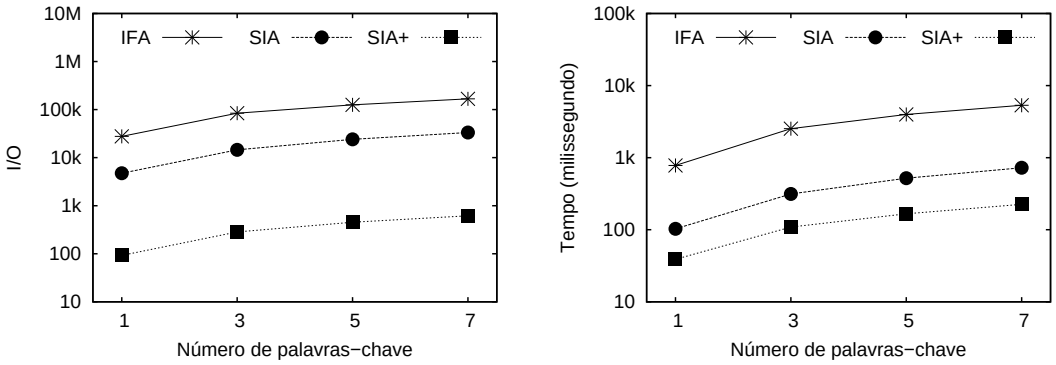
Na Tabela 5.9, observa-se que o SIA$^+$ obteve um tempo de resposta melhor em relação ao SIA em todos valores de k experimentados. A diferença entre o SIA$^+$ e o SIA é de quase uma ordem de magnitude, enquanto o IFA obteve um tempo de resposta maior do que os demais algoritmos propostos em quase uma ordem de magnitude.

Assim como no critério de vizinhança seleção espacial ($Q.\psi = rng$), é esperado que a quantidade de páginas lidas e o tempo de resposta se mantenham estáveis independentemente do valor de k escolhido. Uma vez que para qualquer valor de k, o escore de cada objeto de interesse em P precisa ser calculado. Por isto, nota-se

nas Tabelas 5.8 e 5.9 que o tempo de resposta e a quantidade de páginas lidas se mantém estáveis mesmo aumentando o valor de k.

5.3.2 Variando o número de palavras-chave

Neste experimento pretende-se estudar o impacto da quantidade de palavras-chave em uma consulta EPPC utilizando os três algoritmos propostos. Para avaliar este impacto são coletados o tempo de resposta e o I/O de cada experimento. A Figura 5.5 apresenta o número de páginas lidas e o tempo de resposta ao variar o número de palavras-chave da consulta.



(a) Páginas lidas. (b) Tempo de resposta.

Figura 5.5: Quantidade de páginas lidas e tempo de resposta ao variar a quantidade de palavras-chave utilizando vizinho mais próximo ($Q.\psi = nn$).

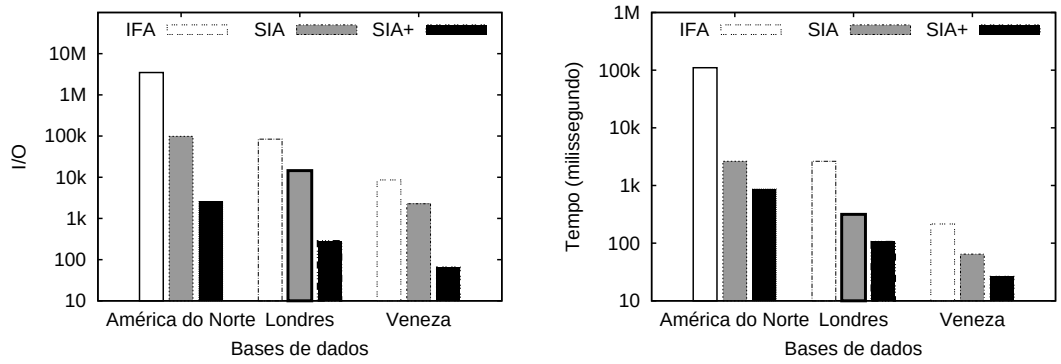
No experimento da Figura 5.5(a), o aumento do acesso a páginas de disco é consequência do aumento do número de palavras-chave, que proporciona um aumento no volume de dados acessados durante o processamento da consulta. Ainda assim, o SIA$^+$ apresentou duas ordens de magnitude a menos de páginas lidas em todos os casos experimentados.

O bom desempenho do SIA$^+$ se mantém quando é analisado o tempo de resposta. Na Figura 5.5(b), o menor tempo de resposta é mantido pelo algoritmo SIA$^+$ mesmo variando o número de palavras-chave.

Assim como no critério seleção espacial, o bom desempenho em quantidade de páginas lidas do SIA$^+$ não é refletido plenamente no tempo de resposta da consulta ao utilizar o critério vizinho mais próximo. Apesar disto, o SIA$^+$ obteve o menor tempo de resposta em todos os experimentos realizados. Portanto, computar o escore dos objetos de interesse paralelamente se mostrou uma boa abordagem para processar a consulta EPPC.

5.3.3 Variando o tamanho da base de dados

Neste experimento, a Figura 5.6 apresenta o tempo de resposta e a quantidade de páginas lidas durante o processamento da consulta em três bases de dados de tamanhos diferentes. A vantagem do SIA$^+$ para os outros algoritmos se mantém quando é variado o tamanho da base de dados. Observa-se na Figura 5.6(a) que mesmo aumentando o tamanho da base de dados, o SIA$^+$ é o algoritmo que acessa menos páginas de disco.



(a) Páginas lidas. (b) Tempo de resposta.

Figura 5.6: Quantidade de páginas lidas e tempo de resposta ao variar o tamanho da base de dados utilizando vizinho mais próximo ($Q.\psi = nn$).

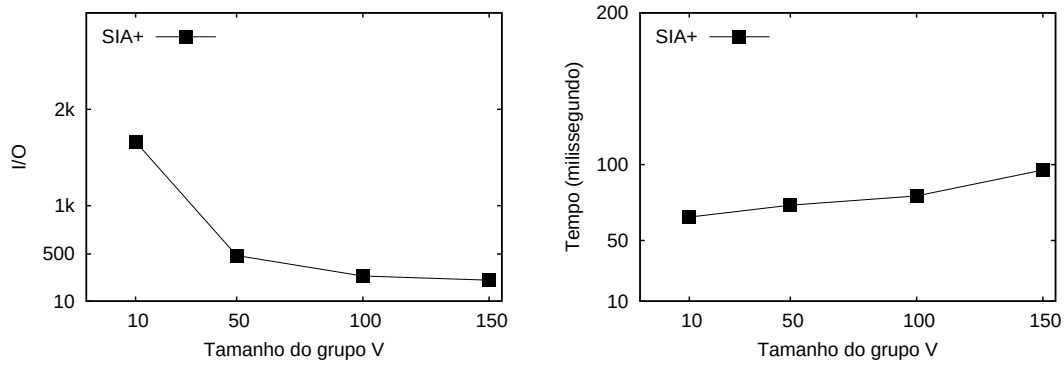
Ao aumentar o tamanho da base dados, aumenta-se o tamanho do conjunto de objetos de interesse e o tamanho do conjunto de objetos de referência. Na Figura 5.6(b), observa-se que mesmo aumentando o tamanho da base de dados, o tempo de resposta do SIA$^+$ é o menor entre os três algoritmos analisados.

Nota-se que o SIA$^+$ acessou menos o disco ao processar a consulta EPPC utilizando o vizinho mais próximo do que utilizando seleção espacial. Isto indica que a abordagem para filtrar objetos de referência com o critério $Q.\psi = nn$ foi mais eficiente.

5.3.4 Variando o tamanho dos grupos (S2I+)

No experimento apresentado na Figura 5.7 é variado o tamanho do grupo V e é coletado o tempo de resposta e a quantidade de páginas lidas. Neste experimento, o tamanho do grupo V foi variado para armazenar no máximo 10, 50, 100 ou 150 objetos de interesse. Os gráficos apresentados nesta subseção não utilizam escala logarítmica devido a pequena diferença encontrada no tempo de resposta.

Os resultados apresentados na Figura 5.7 são semelhantes aos obtidos no experimento realizado com o critério de vizinhança seleção espacial. A quantidade de páginas lidas para processar a consulta EPPC utilizando o SIA$^+$ diminui a medida



(a) Páginas lidas. (b) Tempo de resposta.

Figura 5.7: Quantidade de páginas lidas e tempo de resposta ao variar o tamanho do grupo V utilizando vizinho mais próximo ($Q.\psi = nn$).

que o tamanho do grupo V aumenta (Figura 5.7(a)); enquanto o tempo de resposta aumenta a medida que o tamanho do grupo V aumenta.

Entretanto, nota-se na Figura 5.7(a) que o tempo de resposta da consulta utilizando o critério vizinho mais próximo é maior do que com o critério seleção espacial.

O IFA utiliza um Arquivo Invertido Adaptado para indexar os objetos espaço-textuais de referência. Como o IF é um índice que não tem recursos para filtrar objetos de uma determinada localização espacial, os resultados desta abordagem foram inferiores quando comparado as outras duas abordagens que utilizam o índice hibrido (S2I).

Utilizar índices híbridos resultou em otimizações no desempenho da consulta EPPC, uma vez que a consulta EPPC necessita conhecer a relação espacial entre os objetos, e o S2I permite obter essa relação espacial com maior eficiência, os algoritmos que utilizaram este índice obtiveram melhores resultados tanto na quantidade de páginas acessadas ao disco, quanto no tempo de resposta da consulta.

O processamento concorrente do escore dos objetos de interesse permitiu otimizar ainda mais o desempenho da consulta EPPC. Esta abordagem proporcionou ao SIA$^+$ um resultado superior em tempo de resposta e principalmente em I/O. O algoritmo SIA$^+$ obteve os melhores resultados em todos os experimentos realizados para os critérios de vizinhança seleção espacial ($Q./psi = rng$) e vizinho mais próximo ($Q./psi = nn$).

5.4 Influência

O critério de vizinhança influência é o critério que adiciona mais custo para processar a consulta EPPC. Neste critério, todos objetos que possuem pelo menos um termo

t do conjunto de palavras-chave $Q.D$ é um possível candidato a ser o objeto de referência mais relevante na vizinhança espacial do objeto de interesse. Portanto, não é possível filtrar ramos da aR-tree como demonstrado no Capítulo 4, Seção 4.2.2. Sendo assim, o $influenceScore$ precisa ser calculado para todos os objetos de referência, resultando em baixo desempenho para os algoritmos baseados no S2I (SIA e SIA$^+$).

Por esse motivo, os gráficos desta seção apresentam apenas os resultados obtidos ao processar a consulta EPPC utilizando o algoritmo IFA. Esta seção é dividida em subseções, onde cada seção apresenta o I/O e o tempo de resposta de um experimento específico. Inicialmente é apresentado experimento que varia a quantidade k de objetos retornados pela consulta. Em seguida é apresentado o experimento que varia a quantidade de palavras-chave da consulta. Por fim, é discutido o experimento que varia o tamanho da base de dados.

5.4.1 Variando k

Nas Tabelas 5.10 e 5.11 são apresentados o número de páginas lidas e o tempo de resposta enquanto foi realizada a variação da quantidade de objetos retornados pela consulta (k). A primeira coluna de cada uma destas tabelas listam os valores utilizados para k, e nas demais colunas são apresentadas a quantidade de páginas lidas (Tabela 5.10) ou o tempo de resposta, em milissegundos, (Tabela 5.11) para o algoritmo IFA.

Pode-se observar na Tabela 5.10 que o algoritmo IFA obteve resultados bem próximos ao variar a quantidade k de objetos de interesse retornados. A quantidade de páginas acessadas se mantém a mesma pois independentemente do valor de k, o escore de cada objeto de interesse em P precisa ser calculado.

Tabela 5.10: Quantidade de páginas lidas ao variar a quantidade de resultados (k) utilizando influência ($Q.\psi = inf$).

Resultados (k)	**IFA**
1	84262.356
5	84262.356
10	84262.356
15	84262.356

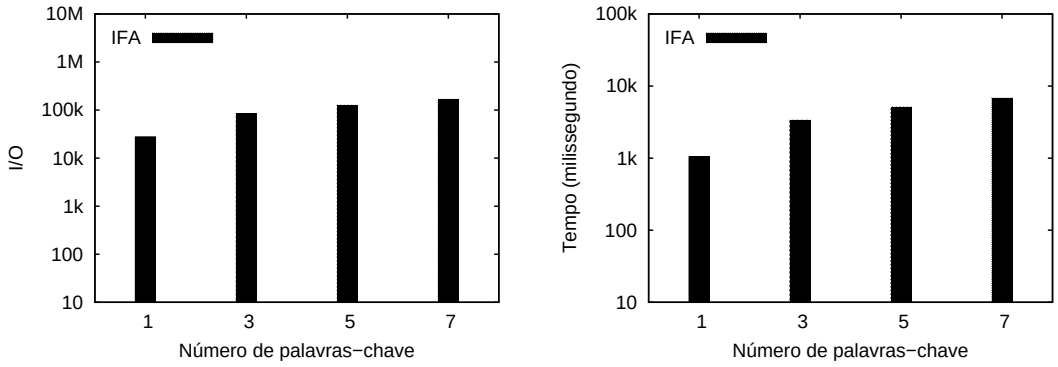
Na Tabela 5.11 o tempo de resposta se manteve com pequenas variações, uma vez que apresentar mais objetos como resposta ao usuário implica em adicionar pouco custo computacional ao processamento da consulta. É necessário apenas ter uma *heap* maior e fazer mais verificações quando for inserir um novo elemento nesta *heap*, como demonstrado no Capítulo 1.

Tabela 5.11: Tempo de resposta (ms) ao variar a quantidade de resultados (k) utilizando influência ($Q.\psi = inf$).

Resultados (k)	**IFA**
1	3384.473
5	3392.223
10	3389.958
15	3393.014

5.4.2 Variando o número de palavras-chave

A Figura 5.8 apresenta o número de páginas lidas e o tempo de resposta ao variar o número de palavras-chave da consulta. Como esperado, observa-se na Figura 5.8(a) que a quantidade de páginas acessadas aumenta ao aumentar o número de palavras-chave. Isto ocorre, pois quanto mais termos existirem no conjunto de palavras-chave, mais vezes será necessário acessar o IF e mais objetos de referência serão acessados, como demonstrado no Algoritmo 1.



(a) Páginas lidas. (b) Tempo de resposta.

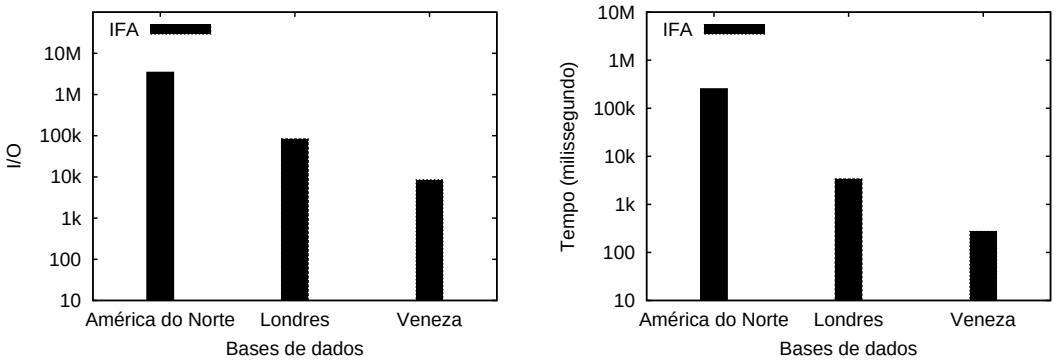
Figura 5.8: Quantidade de páginas lidas e tempo de resposta ao variar a quantidade de palavras-chave utilizando influência ($Q.\psi = inf$).

Observa-se na Figura 5.8(b) que o tempo de resposta aumenta proporcionalmente à quantidade de termos no conjunto de palavras-chave. Isto ocorre devido à maior quantidade de informação que precisa ser acessada durante o processamento da consulta.

5.4.3 Variando o tamanho da base de dados

Por fim, a Figura 5.9 apresenta o tempo de resposta e a quantidade de páginas lidas durante o processamento da consulta em três bases de dados de tamanhos diferentes. A quantidade de páginas acessadas aumentou proporcionalmente ao aumento do

tamanho da base de dados, como pode ser visto na Figura 5.9(a). Ao aumentar o tamanho da base de dados consequentemente aumenta-se a quantidade de objetos de referência a serem analisados, resultando em um maior I/O.



(a) Páginas lidas. (b) Tempo de resposta.

Figura 5.9: Quantidade de páginas lidas e tempo de resposta ao variar o tamanho da base de dados utilizando influência ($Q.\psi = inf$).

Através da Figura 5.9(b) observa-se que aumentar o tamanho da base de dados resulta também em mais custo para processar a consulta EPPC. Pois, é necessário calcular o escore textual de uma quantidade maior de objetos de referência e de objetos de interesse.

Capítulo 6

Considerações Finais

"O homem que vê o mundo aos 50
da mesma forma que o via aos 20,
desperdiçou 30 anos de sua vida "
– Muhammed Ali

Esta dissertação apresentou a proposta de uma nova consulta espaço-textual e algoritmos que proporcionam diferentes abordagens para processar esta nova consulta. A proposição desta consulta baseou-se no estudo das consultas espaço-textuais existentes na literatura. Neste estudo, observou-se uma grande variedade de consultas espaço-textuais e também uma grande diversidade de algoritmos para processá-las.

Observou-se que a consulta Espacial Preferencial Tradicional poderia ser modificada para trazer mais benefícios para o usuário. Sendo assim, foi proposta uma nova consulta espaço-textual, baseada na consulta Espacial Preferencial Tradicional que permite ao usuário utilizar um conjunto de palavras-chave para representar o objeto espacial que ele deseja obter da base de dados.

A metodologia empregada neste trabalho compreendeu etapas que foram desde a proposta da consulta Espacial Preferencial por Palavra-chave (EPPC), até a avaliação experimental dos algoritmos propostos para processar esta nova consulta. Nas etapas intermediárias foram preparadas algumas bases de dados onde foram executados os experimentos, além da proposta de três novos algoritmos para processar a consulta EPPC.

6.1 Contribuições

As principais contribuições desta pesquisa estão na especificação da consulta e nos três algoritmos propostos para processar esta consulta. A consulta EPPC incorpora a busca textual à consulta Espacial Preferencial Tradicional, acrescentado as seguintes vantagens a esta consulta:

1. Do ponto de vista do usuário, elimina-se a restrição de utilizar apenas conjuntos pré-definidos de bases de dados, permitindo que o usuário descreva o que deseja obter utilizando um conjunto de palavras-chave.
2. Do ponto de vista da aplicabilidade, esta consulta ganhou uma abrangência maior, pois não requer que as bases espaço-textuais sejam pré-categorizadas. Assim, é necessário apenas que a base possua objetos espaço-textuais, como os objetos providos pelo Twitter, para que a consulta EPPC possa ser utilizada.

Além destas contribuições, foram disponibilizadas as bases de dados utilizadas nos experimentos, permitindo que outros autores utilizem estas bases para realizar seus próprios experimentos, e permitindo inclusive a comparação de resultados. Além disto, a fundamentação teórica realizada também representa uma contribuição deste trabalho. Esta fundamentação reuniu conceitos e técnicas utilizadas no processamento de consultas espaço-textuais que podem auxiliar outros pesquisadores em trabalhos semelhantes.

6.2 Pesquisas Futuras

A seguir são apresentadas algumas possibilidades de estender o trabalho apresentado nesta dissertação.

Bases de dados. Nesta dissertação, todos algoritmos propostos foram avaliados utilizando bases de dados reais provenientes do OpenStreetMap. Entretanto, não foram realizadas avaliações em bases com características diferentes, como: Twitter[1] ou Wikipedia[2].

Uma base de dados proveniente do Twitter possui um volume de dados e um número de termos maiores do que as bases do OpenStreetMap. Enquanto uma base de dados proveniente da Wikipedia é composta por objetos que possuem uma descrição textual maior do que os objetos provenientes do OpenStreetMap.

EPPC *on road networwks*. Na consulta EPPC, a distância entre o objeto de interesse e o objeto de referência é dada pela distância Euclidiana entre estes dois pontos. Em uma situação cotidiana, é comum que a distância entre dois estabelecimentos não seja uma reta, e sim, um conjunto de ruas ou estradas. Portanto, uma

[1] https://twitter.com/
[2] https://www.wikipedia.org/

extensão natural seria adaptar a consulta EPPC para que esta leve em consideração a rota entre o objeto de interesse e o objeto de referência (ex: uma rua que liga um bar a um hotel).

Dado um conjunto de objetos de interesse, um conjunto de objetos de referência, um conjunto de palavras-chave e um critério de vizinhança. Uma consulta EPPC *on road networwks* retorna k objetos de interesse que possuem pelo menos um objeto de referência relevante textualmente para o conjunto de palavras-chave fornecido pelo usuário. A distância entre um objeto de interesse e um objeto de referência é obtida pelo menor caminho (rota) existente entre estes dois objetos.

Esta consulta não pode ser processada utilizando R-trees [Rocha-Junior e Nørvåg 2012], portanto novos algoritmos e índices espaço-textuais devem ser propostos para processar esta consulta. A princípio, esta é uma consulta inovadora e que apresenta ainda mais desafios para ser processada.

Proposta de novos algoritmos. Os resultados obtidos nos experimentos evidenciam que todos os algoritmos propostos reduzem significativamente o número de páginas acessadas para processar a consulta EPPC. Entretanto, a diferença entre o tempo de resposta do SIA e do SIA$^+$ não é tão significante. Logo, investigar novos algoritmos que otimizem ainda mais o tempo de resposta seria uma contribuição relevante.

Além disto, pode-se investigar um algoritmo capaz de processar a consulta EPPC utilizando o critério influência de forma mais eficiente. Visando propor um algoritmo que seja capaz de filtrar objetos espaciais utilizando o critério de vizinhança influência, algoritmos como o *skyline* [Borzsony et al. 2001, Rocha-Junior et al. 2010] podem ser estudados com este objetivo, ou estudar a possibilidade em utilizar índices híbridos diferentes do S2I, como a IR-tree [Li et al. 2011].

Consulta EPPC em Sistemas Distribuídos. Durante as últimas décadas, o grande número de fontes de dados e a alta taxa de produção destes dados dificulta o armazenamento de todo este volume de informação em um único local. Em decorrência disto, o gerenciamento e armazenamento de dados tem se tornado cada vez mais distribuído [Goldszmidt e Yemini 1995, Rocha-Junior 2012, Venugopal et al. 2006]. Sendo assim, podem ser estudadas as modificações necessárias para processar a consulta EPPC em um ambiente distribuído, ampliando a aplicabilidade da consulta.

Referências Bibliográficas

[Balke et al. 2002] Balke, W.-T., Güntzer, U., e Kießling, W. (2002). On real-time top k querying for mobile services. In *On the Move to Meaningful Internet Systems 2002: CoopIS, DOA, and ODBASE*, pp. 125–143. Springer.

[Baruffolo 1999] Baruffolo, A. (1999). R-trees for astronomical data indexing. In *Astronomical Data Analysis Software and Systems VIII*, volume 172, pp. 375.

[Bayer e McCreight 1970] Bayer, R. e McCreight, E. (1970). Organization and maintenance of large ordered indexes. In *Workshop on Data Description and Access*, Houston, Texas. ACM-SIGFIDET.

[Beckmann et al. 1990] Beckmann, N., Kriegel, H.-P., Schneider, R., e Seeger, B. (1990). The R*-tree: An efficient and robust access method for points and rectangles. In *SIGMOD*, pp. 322–331, Atlantic City. ACM.

[Bentley e Friedman 1979] Bentley, J. L. e Friedman, J. H. (1979). Data structures for range searching. *ACM Computing Surveys (CSUR)*, 11(4):397–409.

[Borzsony et al. 2001] Borzsony, S., Kossmann, D., e Stocker, K. (2001). The skyline operator. In *Data Engineering, 2001. Proceedings. 17th International Conference on*, pp. 421–430. IEEE.

[Cao et al. 2012] Cao, X., Chen, L., Cong, G., Jensen, C. S., Qu, Q., Skovsgaard, A., Wu, D., e Yiu, M. L. (2012). Spatial keyword querying. In *ER*, pp. 16–29. Springer.

[Chen et al. 2013] Chen, L., Cong, G., Jensen, C. S., e Wu, D. (2013). Spatial keyword query processing: an experimental evaluation. In *PVLDB*, volume 6, pp. 217–228. VLDB Endowment.

[Chomicki 2003] Chomicki, J. (2003). Preference formulas in relational queries. *ACM Transactions on Database Systems (TODS)*, 28(4):427–466.

[Cohen et al. 2003] Cohen, W., Ravikumar, P., e Fienberg, S. (2003). A comparison of string metrics for matching names and records. In *KDD Workshop on Data Cleaning and Object Consolidation*, volume 3, pp. 73–78.

[Comer 1979] Comer, D. (1979). Ubiquitous b-tree. *ACM Computing Surveys (CSUR)*, 11(2):121–137.

[Cong et al. 2009] Cong, G., Jensen, C. S., e Wu, D. (2009). Efficient retrieval of the top-k most relevant spatial web objects. In *PVLDB*, volume 2, pp. 337–348. VLDB Endowment.

[Deri 2015] Deri (2015). E-tourism working group. *Disponível em: http://e-tourism.deri.at/index.html*.

[Fu et al. 2012] Fu, B., Fink, E., Gibson, G., e Carbonell, J. G. (2012). Indexing and fast near-matching of billions of astronomical objects. *Proceedings of the Fourth Workshop on Interfaces and Architecture for Scientific Data Storage*.

[Gao e Xia 2006] Gao, S. e Xia, Y. (2006). Gdcic: A grid-based density-confidence-interval clustering algorithm for multi-density dataset in large spatial database. In *Intelligent Systems Design and Applications, 2006. ISDA'06. Sixth International Conference on*, volume 1, pp. 713–717. IEEE.

[Georgiadis et al. 2008] Georgiadis, P., Kapantaidakis, I., Christophides, V., Nguer, E., e Spyratos, N. (2008). Efficient rewriting algorithms for preference queries. In *Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on*, pp. 1101–1110. IEEE.

[Goldszmidt e Yemini 1995] Goldszmidt, G. e Yemini, Y. (1995). Distributed management by delegation. In *Distributed Computing Systems, 1995., Proceedings of the 15th International Conference on*, pp. 333–340. IEEE.

[Güting 1994] Güting, R. H. (1994). An introduction to spatial database systems. *The VLDB Journal–The International Journal on Very Large Data Bases*, 3(4).

[Guttman 1984] Guttman, A. (1984). *R-trees: a dynamic index structure for spatial searching*, volume 14. ACM.

[Guttman e Stonebraker 1982] Guttman, A. e Stonebraker, M. (1982). Using a relational database management system for computer aided design data. *IEEE Database Eng. Bull.*, 5(2):21–28.

[Hariharan et al. 2007] Hariharan, R., Hore, B., Li, C., e Mehrotra, S. (2007). Processing spatial-keyword (sk) queries in geographic information retrieval (gir) systems. In *Scientific and Statistical Database Management, 2007. SSBDM'07. 19th International Conference on*, pp. 16–16. IEEE.

[Ilyas et al. 2008] Ilyas, I. F., Beskales, G., e Soliman, M. A. (2008). A survey of top-k query processing techniques in relational database systems. *ACM Computing Surveys (CSUR)*, 40(4):11.

[Khodaei et al. 2010] Khodaei, A., Shahabi, C., e Li, C. (2010). Hybrid indexing and seamless ranking of spatial and textual features of web documents. In *Database and Expert Systems Applications*, pp. 450–466. Springer.

[Kießling 2002] Kießling, W. (2002). Foundations of preferences in database systems. In *Proceedings of the 28th international conference on Very Large Data Bases*, pp. 311–322. VLDB Endowment.

[Kießling e Köstler 2002] Kießling, W. e Köstler, G. (2002). Preference sql: design, implementation, experiences. In *Proceedings of the 28th international conference on Very Large Data Bases*, pp. 990–1001. VLDB Endowment.

[Knuth 1968] Knuth, D. (1968). The art of computer programming.

[Lacroix e Lavency 1987] Lacroix, M. e Lavency, P. (1987). Preferences; putting more knowledge into queries. In *VLDB*, volume 87, pp. 1–4.

[Lasker et al. 2008] Lasker, B. M., Lattanzi, M. G., McLean, B. J., Bucciarelli, B., Drimmel, R., Garcia, J., Greene, G., Guglielmetti, F., Hanley, C., Hawkins, G., et al. (2008). The second-generation guide star catalog: description and properties. *The Astronomical Journal*, 136(2):735.

[Li et al. 2011] Li, Z., Lee, K. C., Zheng, B., Lee, W.-C., Lee, D. L., e Wang, X. (2011). Ir-tree: An efficient index for geographic document search. *Knowledge and Data Engineering, IEEE Transactions on*, 23(4):585–599.

[Liu et al. 2011] Liu, J., Yu, G., e Sun, H. (2011). Subject-oriented top-k hot region queries in spatial dataset. In *Proceedings of the 20th International Conference on Information and Knowledge Management*. ACM.

[Manning et al. 2008] Manning, C. D., Raghavan, P., Schütze, H., et al. (2008). *Introduction to information retrieval*, volume 1. Cambridge university press.

[Papadias et al. 2001] Papadias, D., Kalnis, P., Zhang, J., e Tao, Y. (2001). Efficient OLAP operations in spatial data warehouses. In *SSTD*, pp. 443–459. Springer.

[Rigaux et al. 2001] Rigaux, P., Scholl, M., e Voisard, A. (2001). *Spatial databases: with application to GIS*. Morgan Kaufmann.

[Rocha-Junior 2012] Rocha-Junior, J. B. (2012). *Efficient processing of preference queries in distributed and spatial databases*. PhD thesis, Norwegian University of Science and Technology.

[Rocha-Junior et al. 2011] Rocha-Junior, J. B., Gkorgkas, O., Jonassen, S., e Nørvåg, K. (2011). Efficient processing of top-k spatial keyword queries. In *SSTD*, pp. 205–222. Springer.

[Rocha-Junior e Nørvåg 2012] Rocha-Junior, J. B. e Nørvåg, K. (2012). Top-k spatial keyword queries on road networks. In *EDBT*, pp. 168–179. ACM.

[Rocha-Junior et al. 2010] Rocha-Junior, J. B., Vlachou, A., Doulkeridis, C., e Nørvåg, K. (2010). Efficient processing of top-k spatial preference queries. In *PVLDB*, volume 4, pp. 93–104. VLDB Endowment.

[Salminen e Tompa 1994] Salminen, A. e Tompa, F. W. (1994). Pat expressions: an algebra for text search. *Acta Linguistica Hungarica*, 41(1):277–306.

[Samet 1984] Samet, H. (1984). The quadtree and related hierarchical data structures. *ACM Computing Surveys (CSUR)*, 16(2):187–260.

[Twitter 2014a] Twitter (2014a). The search api. *Disponível em: https://dev.twitter.com/rest/public/search*.

[Twitter 2014b] Twitter (2014b). The streaming apis. *Disponível em: https://dev.twitter.com/streaming/overview*.

[Vaid et al. 2005] Vaid, S., Jones, C. B., Joho, H., e Sanderson, M. (2005). Spatio-textual indexing for geographical search on the web. In *SSTD*, pp. 218–235. Springer.

[Venugopal et al. 2006] Venugopal, S., Buyya, R., e Ramamohanarao, K. (2006). A taxonomy of data grids for distributed data sharing, management, and processing. *ACM Computing Surveys (CSUR)*, 38(1):3.

[Wu et al. 2012a] Wu, D., Cong, G., e Jensen, C. S. (2012a). A framework for efficient spatial web object retrieval. *The VLDB Journal–The International Journal on Very Large Data Bases*, 21(6):797–822.

[Wu et al. 2012b] Wu, D., Yiu, M. L., Cong, G., e Jensen, C. S. (2012b). Joint top-k spatial keyword query processing. *Knowledge and Data Engineering, IEEE Transactions on*, 24(10):1889–1903.

[Yiu et al. 2007] Yiu, M. L., Dai, X., Mamoulis, N., e Vaitis, M. (2007). Top-k spatial preference queries*. In *IEEE 23rd International Conference on Data Engineering, 2007. ICDE 2007.*, pp. 1076–1085. IEEE.

[Yiu et al. 2011] Yiu, M. L., Lu, H., Mamoulis, N., e Vaitis, M. (2011). Ranking spatial data by quality preferences. *IEEE Transactions on Knowledge and Data Engineering*, 23(3).

[Yueh et al. 2007] Yueh, Y. T., Chiu, D. K., Leung, H.-f., e Hung, P. C. (2007). A virtual travel agent system for m-tourism with semantic web service based design and implementation. In *Advanced Information Networking and Applications, 2007. AINA'07. 21st International Conference on*, pp. 142–149. IEEE.

[Zhou et al. 2005] Zhou, Y., Xie, X., Wang, C., Gong, Y., e Ma, W.-Y. (2005). Hybrid index structures for location-based web search. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pp. 155–162. ACM.

[Zhu et al. 2011] Zhu, S., Wu, J., Xiong, H., e Xia, G. (2011). Scaling up top-K cosine similarity search. *Data & Knowledge Engineering*, 70(1):60–83.

[Zobel e Moffat 2006] Zobel, J. e Moffat, A. (2006). Inverted files for text search engines. In *ACM Computing Surveys (CSUR)*, volume 38, pp. 6. ACM.