UNIVERSIDADE FEDERAL DE SÃO CARLOS
CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA
CAMPUS SÃO CARLOS

Leila Aparecida da Silva

UMA ANÁLISE EXPERIMENTAL DE MODELOS DE RELACIONAMENTOS DIRECIONAIS EM BANCO DE DADOS ESPACIAIS

São Carlos
Abril de 2022

LEILA APARECIDA DA SILVA

UMA ANÁLISE EXPERIMENTAL DE MODELOS DE RELACIONAMENTOS DIRECIONAIS EM BANCO DE DADOS ESPACIAIS

Trabalho de Conclusão de Curso submetido à Universidade Federal de São Carlos, como requisito necessário para obtenção do grau de Bacharel em Engenharia de Computação

São Carlos, Abril de 2022

UNIVERSIDADE FEDERAL DE SÃO CARLOS

LEILA APARECIDA DA SILVA

Esta Monografia foi julgada adequada para a obtenção do título de Bacharel em Engenharia de Computação, sendo aprovada em sua forma final pela banca examinadora:

Orientador: Prof. Dr. Anderson Chaves Carniel
Universidade Federal de São Carlos
UFSCar

Profa. Dra. Marcela Xavier Ribeiro
Universidade Federal de São Carlos
UFSCar

Prof. Dr. Renato Bueno
Universidade Federal de São Carlos
UFSCar

São Carlos, Abril de 2022

Agradeço e dedico este trabalho aos meus pais, à minha irmã e ao meu marido Gustavo, por todo o apoio durante todos estes anos de graduação. Esta monografia é a prova de que todo o investimento e dedicação valeram a pena.

Agradecimentos

Agradeço primeiramente a Deus, pela minha vida, e por me permitir superar todos os obstáculos encontrados ao longo da graduação e da realização deste trabalho. Aos meus pais Maria José e Gilvan pelo incentivo durante toda a graduação. Ao meu marido Gustavo pela compreensão e apoio nestes meses do desenvolvimento do trabalho. Aos professores que contribuíram com a minha formação, em especial ao meu orientador Prof. Dr. Anderson Chaves Carniel.

Resumo

Palavras-chave: banco de dados espaciais, relacionamentos direcionais, consultas espaciais

A utilização de dados espaciais está em constante crescimento nas mais variadas áreas do conhecimento. Estas informações são armazenadas em Bancos de Dados Espaciais e acessadas por meio de consultas espaciais, que utilizam aproximações para simplificar os objetos e assim reduzir o tempo de execução. As consultas espaciais usam relacionamentos espaciais, tais como relacionamentos direcionais. Existem vários modelos propostos na literatura para identificar relacionamentos direcionais, como o *Direction-Relation Matrix* (DRM) simples e refinado, e o *Objects Interaction Matrix* (OIM), que não possuem implementações práticas. Assim, os desenvolvedores de aplicações que desejam utilizar relacionamentos direcionais precisam implementar suas próprias versões destes modelos, o que é custoso, demorado e pode acarretar erros. Deste modo, este trabalho propõe implementações dos modelos DRM e OIM na linguagem R, realizando ainda análises experimentais para avaliar e comparar estes modelos. De acordo com a análise realizada, utilizando um conjunto de dados com aproximadamente 100 mil objetos espaciais do estado de São Paulo, o modelo DRM simples se mostrou o mais rápido, uma vez que utiliza operações espaciais menos complexas.

Abstract

Keywords: spatial databases, directional relations, spatial queries

The use of spatial data is constantly growing in several areas of knowledge. This information is stored in Spatial Databases, which use approximations to simplify objects and thus reduce execution time. Spatial queries employ spatial relations, such as directional relationships. There are a number of models proposed in the literature to identify directional relationships, such as the simple and detailed Direction-Relation Matrix (DRM) and the Objects Interaction Matrix (OIM), that do not have practical implementations. Therefore, application developers who want to use directional relationships need to implement their own versions of these models, which is expensive, time-consuming task and can lead to errors. Thus, this work proposes implementations of the DRM and OIM models in the R language, also performing experimental analysis to evaluate and compare these models. According to the analysis performed, using a dataset with approximately 100,000 spatial objects from the state of São Paulo, the simple DRM model proved to be the fastest since it uses less complex spatial operations.

Lista de abreviaturas e siglas

BDE	Banco de Dados Espaciais
DRM	*Direction-Relation Matrix*
GEOS	*Geometry Engine, Open Source*
OIG	*Objects Interaction Grid*
OIM	*Objects Interaction Matrix*
SGBD	Sistema de Gerenciamento de Banco de Dados
SIG	Sistema de Informação Geográfica
TCC	Trabalho de Conclusão de Curso
UFSCar	Universidade Federal de São Carlos

Lista de ilustrações

Lista de códigos

Sumário

1 Introdução

Este é o capítulo introdutório deste trabalho de conclusão de curso (TCC), sendo composto pelas seguintes seções: A seção 1.1 apresenta o contexto no qual este TCC está inserido. As motivações que levaram à escolha deste tema e desenvolvimento do trabalho são apresentadas na seção 1.2. As seções 1.3, 1.4 e 1.5 apresentam, respectivamente, a justificativa, os objetivos e a forma como este trabalho está organizado.

1.1 Contextualização

Com o crescente número de dispositivos conectados à Internet, um volume cada vez maior de dados é gerado a todo momento. Esses dados são analisados e utilizados para automatizar campanhas de marketing, otimizar os gastos na produção agrícola, sugerir as melhores rotas de viagem em tempo real, além de muitas outras aplicações que estão se tornando cada vez mais presentes no cotidiano.

Em muitas áreas do conhecimento há uma necessidade de armazenar e manipular, além dos dados convencionais compostos de atributos alfanuméricos, dados *espaciais* (dados geográficos), ou seja, dados que possuem uma relação com o espaço. Dados espaciais possuem duas componentes: sua localização geográfica, isto é, a posição do objeto em um sistema de coordenadas conhecido, e atributos alfanuméricos relacionados. Para representar o formato dos objetos espaciais, são definidos tipos de dados espaciais, como pontos, linhas e regiões (polígonos) [1]. Uma loja pode ser representada como um ponto em uma cidade, e possuir informações associadas, como nome, segmento, horário de funcionamento, entre outros. Por outro lado, alguns fenômenos geográficos necessitam de estruturas mais complexas [2] para representar sua extensão, como as ilhas que compõem um arquipélago.

Devido à popularidade da coleta e armazenamento de dados espaciais, a recuperação de informação espacial é uma tarefa comum em sistemas digitais. Existem sistemas específicos para manipular e armazenar dados espaciais, os chamados Sistemas de Informação Geográfica (SIG) e os Sistemas Gerenciadores de Bancos de Dados Espaciais (SGBDE). Estes sistemas processam dados espaciais e alfanuméricos, com foco em análises espaciais e modelagens de superfícies. Relacionamentos espaciais são utilizados como condições e predicados em consultas espaciais, descrevendo a forma como dois objetos espaciais estão relacionados.

Consultas espaciais geralmente utilizam relacionamentos topológicos, métricos ou de direção, e podem ser utilizadas para determinar quais objetos devem ser retornados

a partir de uma ou mais condições. SIGs e SGBDEs viabilizam o processamento de consultas espaciais através de bibliotecas espaciais. Visando padronizar o formato de objetos espaciais e suas operações, o padrão formal (ISO 19125-1:2004) *Simple Features* (SF) descreve como objetos do mundo real podem ser representados computacionalmente, com ênfase na geometria espacial destes objetos. Também descreve como os objetos podem ser armazenados e recuperados de bancos de dados, e quais operadores geométricos devem ser definidos para eles. O padrão SF é amplamente implementado em GIS comerciais, bancos de dados espaciais (como PostGIS), e forma a base de dados vetorial para bibliotecas como GEOS [3]. GEOS é uma biblioteca desenvolvida em C/C++ para geometria computacional com um foco em algoritmos utilizados em software SIG. Implementa o modelo de geometria estabelecido pelo OGC Simple Features e provê todas as funções espaciais do padrão, além de muitas outras.

1.2 Motivação

Relacionamentos direcionais são relações espaciais qualitativas, que descrevem a posição direcional de um objeto em relação a outros objetos. Podem ser expressos utilizando termos simbólicos que denotam os relacionamentos cardinais: *north* (N), *northeast* (NE), *east* (E), *southeast* (SE), *south* (S), *southwest* (SW), *west* (W), *northwest* (NW), e *same* (0). *Same* se refere à área neutra, quando os dois objetos estão muito sobrepostos. Estas relações podem ser utilizadas para realizar consultas como "Encontre todos os objetos na direção norte em relação a um objeto espacial a". Uma aplicação prática para este tipo de consulta poderia ser uma função de recomendação de estabelecimentos (como hotéis e restaurantes) em um aplicativo de viagens, de modo que o usuário encontre as melhores opções disponíveis saindo o mínimo possível da sua rota original.

Na literatura, existem modelos formais que visam identificar a existência de relacionamentos direcionais entre objetos, como *Direction-Relation Matrix* (DRM) [4] e *Objects Interaction Matrix* (OIM) [5], que são analisados neste trabalho. Estes modelos são uma base importante para que possam ser especificadas consultas espaciais utilizando as operações direcionais. Entretanto, existe uma lacuna em relação à aplicação prática destes modelos, que foram definidos na literatura mas não possuem implementações em bibliotecas de geometria computacional.

1.3 Justificativa

Existem trabalhos científicos que propõem implementações de relacionamentos direcionais, porém apesar de serem bem estabelecidos na literatura, os modelos DRM e OIM não possuem implementações disponíveis em bibliotecas de geometria computacional ou sistemas de banco de dados espaciais, como o PostGIS. Os trabalhos relacionados em

geral descrevem implementações de modelos e análises de desempenho. São utilizados conceitos e estruturas como: Range Queries, MBRs, árvores B-tree e R-tree, e *rectangle algebra*. Os trabalhos não implementam de fato estes modelos em softwares, se restringindo a descrições no próprio texto.

Deste modo, os desenvolvedores de aplicações que desejam utilizar relacionamentos direcionais precisam implementar suas próprias versões destes modelos, o que é custoso, demorado e pode acarretar erros. Este trabalho de conclusão de curso (TCC) está inserido neste contexto, buscando contribuir com uma implementação destes dois modelos, conforme mostrado nos objetivos da subseção a seguir.

1.4 Objetivos

Tendo em vista a necessidade de implementações de relacionamentos direcionais em sistemas de bancos de dados espaciais, o objetivo geral deste TCC é: *Implementar os modelos DRM e OIM e conduzir análises experimentais dessas implementações no processamento de consultas direcionais.* Visando atingir este objetivo, são definidos os seguintes objetivos específicos:

- Objetivo 1. Implementar os modelos DRM e OIM, bem como consultas espaciais utilizando os relacionamentos direcionais definidos por estes modelos.
- Objetivo 2. Conduzir análises experimentais para avaliar e comparar os dois modelos, realizando testes de desempenho e apresentando pontos positivos e negativos de cada um deles.

Para atingir esses objetivos, este trabalho utiliza o *Simple Features for R* (sf) [6], um pacote da linguagem R baseado no GEOS, para prover os relacionamentos direcionais para a linguagem R.

1.5 Organização

Este trabalho possui 5 capítulos, iniciando com o capítulo atual de Introdução e outros 4 capítulos, organizados da seguinte forma:

- Capítulo 2 - Neste capítulo de fundamentação teórica, são apresentados os conceitos necessários para compreensão dos assuntos abordados ao longo dos demais capítulos, incluindo: (i) Banco de Dados Espaciais; (ii) Tipos de Dados Espaciais; (iii) Relacionamentos Espaciais; e (iv) Modelos para Identificar Relacionamentos Direcionais.

- Capítulo 3 - É realizada uma revisão bibliográfica de trabalhos relacionados, nos quais são implementados relacionamentos direcionais. Neste sentido, é apresentado um resumo acerca de cada um destes trabalhos, seus pontos positivos e negativos, além de apresentar o diferencial deste TCC em relação aos trabalhos apresentados.
- Capítulo 4 - As implementações dos relacionamentos direcionais realizadas são apresentadas, descrevendo os algoritmos desenvolvidos para (i) Modelo Direction-Relation Matrix; (ii) Objects Interaction Matrix; (iii) Consultas Espaciais usando Relacionamentos Direcionais. Como resultado, este capítulo demonstra como este TCC atinge aos objetivos 2 e 3.
- Capítulo 5 - Apresenta as conclusões e trabalhos futuros.

2 Fundamentação Teórica

Neste capítulo são apresentados termos técnicos e conceitos utilizados ao longo deste trabalho. A seção 2.1 descreve os tipos de dados espaciais, e define relacionamentos espaciais e direcionais. Na seção 2.2 são abordados os Modelos para Identificar Relacionamentos Direcionais, DRM e OIM.

2.1 Banco de Dados Espaciais

Desde o surgimento de bancos de dados relacionais, tem-se tentado armazenar objetos deste tipo em tabelas relacionais, com tuplas de tamanho fixo, o que se mostrou cada vez mais inviável conforme aumenta o número de objetos. Além disso, as estruturas de dados utilizadas também deveriam dar suporte para a dinamicidade dos dados espaciais.

Güting [1] define, de modo geral, um sistema de banco de dados espacial como um sistema de banco de dados que oferece tipos de dados espaciais em seu modelo de dados, uma linguagem de consulta e suporte a tipos de dados espaciais em sua implementação, provendo indexação espacial e algoritmos eficientes para junções espaciais. É importante ressaltar que informação espacial na prática está sempre relacionada a outros dados alfanuméricos. Além disso, bancos de dados espaciais tendem a ser volumosos e são complexos de serem gerenciados.

2.1.1 Tipos de dados espaciais

A representação de fenômenos da natureza em bancos de dados espaciais é feita por meio de instâncias de tipos de dados espaciais, que podem ser: (a) ponto simples, (b) ponto complexo, (c) linha simples, (d) linha complexa, (e) região simples e (f) região complexa [1]. Estas instâncias podem ser simples ou complexas, dado que objetos simples são aqueles compostos por apenas um componente, enquanto os complexos possuem um número finito de componentes. Na Figura 1 são ilustrados exemplos destes tipos de dados.

Um objeto espacial do tipo ponto é utilizado quando o objeto em questão pode ser representado geograficamente apenas pela localização, sem levar em consideração sua extensão ou área, de modo que é tratado como um objeto de dimensão 0. É utilizado para representar, por exemplo, uma casa em relação a uma cidade. Para representar objetos como rodovias, rios e conexões de energia elétrica, faz-se necessário expressar, além da localização, também uma extensão linear. Assim, é utilizado um objeto espacial do tipo linha, constituído de dois ou mais pontos, que possui dimensão 1. Uma região é a abstração de algo que contém uma extensão no espaço bidimensional, como um país,

um lago ou uma fazenda. Uma região pode ter “buracos” e também pode ser formada por várias peças disjuntas. Por exemplo, países (como a Itália) podem ser formados por múltiplos componentes, como a porção continental e ilhas, e podem possuir buracos como o Vaticano, que apesar de estar contido, não faz parte do país [2].

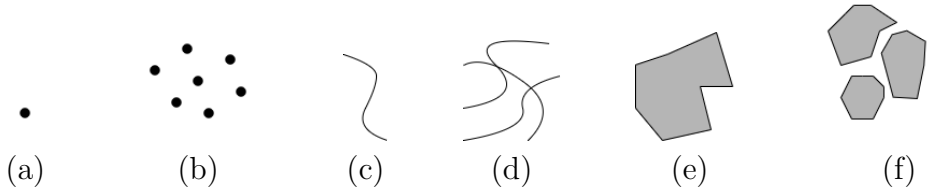


Figura 1 – Tipos de dados espaciais

2.1.2 Relacionamentos espaciais

Dentre as operações oferecidas pelas álgebras espaciais, relacionamentos espaciais são consideradas as mais importantes [1]. Elas tornam possível realizar uma consulta para retornar todos os objetos em um dado relacionamento com um objeto de parâmetro. Pode-se distinguir várias classes de relacionamentos espaciais, como Relacionamentos topológicos, por exemplo, *adjacente*, *contém* e *disjunto* são predicados invariáveis em transformações topológicas como translação, escala e rotação.

Outra classe de relacionamentos espaciais são os relacionamentos métricos, como "distância < 100". E ainda os relacionamentos direcionais, como *acima*, *abaixo*, *ao norte*, *ao sudeste*, entre outros. O foco deste TCC é nos relacionamentos direcionais.

Devido à complexidade dos relacionamentos topológicos, o tempo de processamento para objetos complexos pode ser superior ao tempo disponível, por isso podem ser utilizadas técnicas de aproximação, como o uso de *Minimum Boundary Rectangles* (MBRs), que são utilizados neste trabalho. Um MBR é formado ao utilizar as coordenadas máximas e mínimas de x e y, o que resulta em um retângulo que envolve todo o entorno do objeto espacial [7]. A Figura 2 mostra um exemplo de MBR.

2.1.3 Relacionamentos direcionais

Relacionamentos direcionais expressam a posição de um objeto espacial em relação a outro, chamado de objeto de referência. Estes relacionamentos são utilizados como critérios de seleção e junção em consultas espaciais, e são definidos por modelos que identificam as direções cardinais entre dois objetos. Neste trabalho, são abordados os modelos *Direction-Relation Model* e *Objects Interaction Model* (OIM).

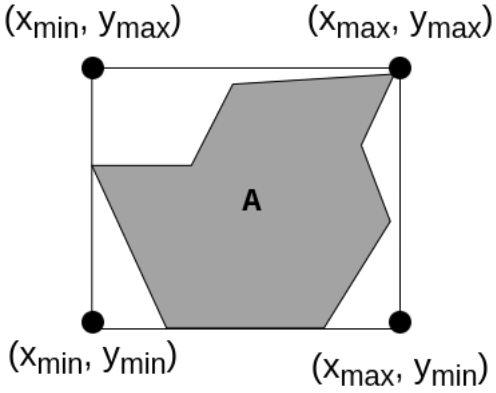


Figura 2 – MBR de um objeto

2.2 Modelos para Identificar Relacionamentos Direcionais

2.2.1 Direction Relation Matrix

O modelo Direction Relation Matrix foi apresentado inicialmente em 1997 por Goyal e Egenhofer [4] e é baseado nas direções cardinais de um objeto de *referência* em relação a um objeto *alvo*, que derivam do *projection-based direction system* [8]. Assim, existem nove possíveis partições de direção: *north* (N), *northeast* (NE), *east* (E), *southeast* (SE), *south* (S), *southwest* (SW), *west* (W), *northwest* (NW), e *same* (0). Same se refere a região neutra, que está contida no MBR do objeto de referência.

Com os objetos de referência e alvo expressos como regiões, conforme mostrado na Figura 3, a matrix DRM possui o formato mostrado abaixo e registra as partições de direção que contém o objeto alvo. Esta representação é expressa em termos dos índices de números de linha e coluna, onde a_{11} corresponde a NW, a_{12} a N, a_{13} a NE, a_{21} a W, a_{22} a Same, a_{23} a E, a_{31} a SW, a_{32} a S e a_{33} a SE.

$$DRM = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

Por exemplo, a direção para o objeto de referência ao objeto alvo mostrada na Figura 3 é representada pela matriz de intersecções abaixo, na qual o símbolo de vazio indica que o objeto alvo não intersecta o quadrante, enquanto o símbolo de não-vazio indica que há uma intersecção neste quadrante. Para obter a direção cardinal entre dois

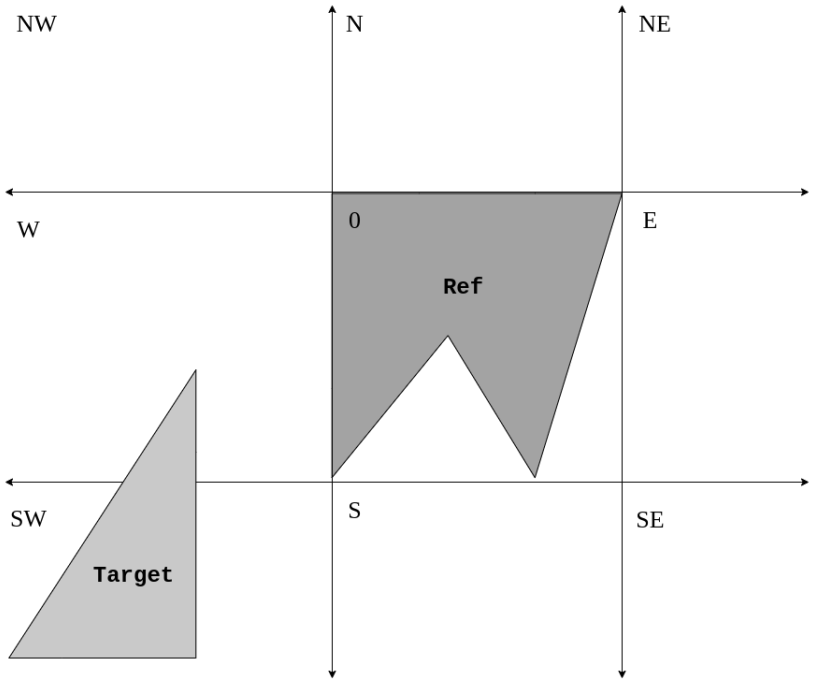


Figura 3 – Objeto de referência e objeto alvo

objetos, é necessário construir a DRM do objeto de referência ao objeto alvo e vice-versa, do objeto alvo ao objeto de referência.

$$DRM(Ref, Target) = \begin{bmatrix} \emptyset & \emptyset & \emptyset \\ \neg\emptyset & \emptyset & \emptyset \\ \neg\emptyset & \emptyset & \emptyset \end{bmatrix}$$

A matriz de intersecções muitas vezes é suficiente para inferências de direção, porém para fornecer um maior detalhamento sobre o relacionamento direcional, é proposta também uma matriz refinada, em que a área do objeto alvo contida em cada uma das partições é considerada. Em vez de gravar uma área absoluta para cada partição, considera-se a porcentagem do objeto alvo que está sobrepondo cada uma das partições não-vazias. Para contabilizar a área, em cada partição é registrado um valor normalizado entre 0 e 1. Um elemento da DRM refinada é igual a 0 se o objeto alvo não cai naquela partição, é 1 se o objeto alvo é contido completamente nesta partição, e está entre 0 e 1 se apenas uma parte do objeto está na área daquela partição.

O valor dos elementos da matriz é dado pela equação abaixo, onde i = 1, ..., 3 e j = 1, ..., 3; $A_{i,j}$ é a área do objeto alvo na partição $a_{i,j}$ da matriz DRM. A soma dos valores de $A_{i,j}$ é igual a 1, pois se trata da soma das proporções.

$$a_{ij} = \frac{A_{ij}}{A}$$

Para os objetos da Figura 3, quando considerada a área do objeto alvo em cada partição, a seguinte matriz refinada é gerada:

$$DRMRefinada = \begin{bmatrix} 0 & 0 & 0 \\ 0.36 & 0 & 0 \\ 0.64 & 0 & 0 \end{bmatrix}$$

2.2.2 Object Interaction Matrix

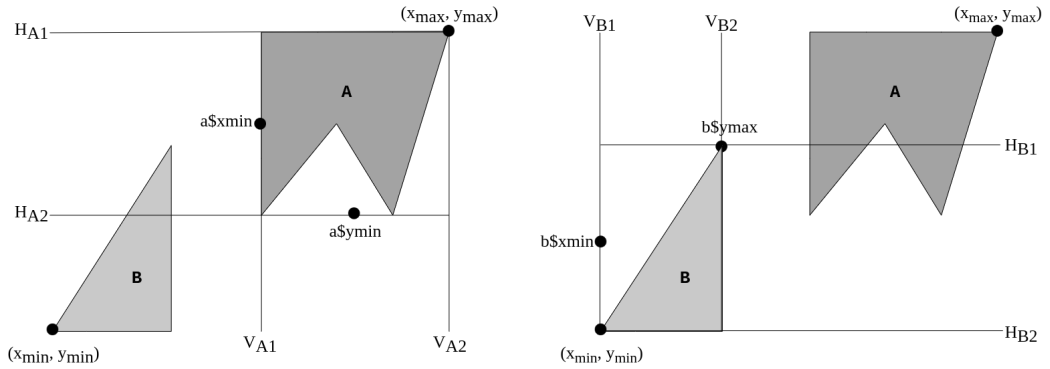
O Objects Interaction Matrix (OIM) é um modelo computacional proposto com o objetivo de superar as limitações das abordagens anteriores, o que requer que o modelo leve em consideração o formato dos objetos operandos, garanta a propriedade de inversão de direções cardiais (ou seja, para um relacionamento p seja tal que *A p B <-> B inv(p) A*), aceite objetos do tipo região com estruturas complexas como operandos e evite resultados errados computados por algumas abordagens.

Assim, o OIM consiste em um método com duas fases que inclui uma fase de ladrilhamento (*tiling*), seguida por uma fase de interpretação. Na primeira fase, é aplicada uma estatégia para determinar inicialmente as áreas que correspondem a nove direções cardiais em relação a cada objeto região. É realizada então a intersecção entre as áreas dos dois objetos, o que resulta em um *grid* chamado Objects Interaction Grid (OIG). Para cada célula do grid, é derivada a informação sobre os objetos que a intersectam, e é armazenada na Objects Interaction Matrix (OIM). Na segunda fase, um método de interpretação é aplicado à matriz OIM para determinar a direção cardial.

Inicialmente, é obtido o Objects Interaction Grid Space (OIGS), que corresponde aos MBRs dos dois objetos. Sejam A e B duas regiões não vazias e min_{ax} é o menor valor para a coordenada x de A, max_{ax} é o maior valor para a coordenada x de A, min_{ay} é o menor valor para a coordenada y de A, max_{ay} é o maior valor para a coordenada x de A. De forma análoga, o mesmo vale para o objeto B.

A ideia geral da estratégia de ladrilhamento é sobrepor o *grid* OIG para uma configuração de dois objetos complexos. Este grid é determinado pelas duas linhas horizontais e verticais de particionamento de cada objeto. Estas linhas são obtidas pela extensão infinita dos dois segmentos horizontais e verticais do MBR de um objeto, que é obtido pelo OIGS. Estas quatro linhas de particionamento criam uma partição do plano Euclidiano, que consiste de nove regiões exclusivas e direcionais, cuja região central é limitada e as oito regiões ao redor são ilimitadas. As Figuras 4 (a) e (b) apresentam os segmentos horizontais e verticais dos dois objetos separadamente, que quando sobrepostos formam o *grid* OIG da Figura 5.

Após obter o OIG, é definida a matriz OIM, que consiste em identificar os objetos presentes em cada partição, seguindo a seguinte lógica: 1 para os quadrantes que contém



(a) (b)

Figura 4 – Segmentos horizontais e verticais de cada um dos objetos

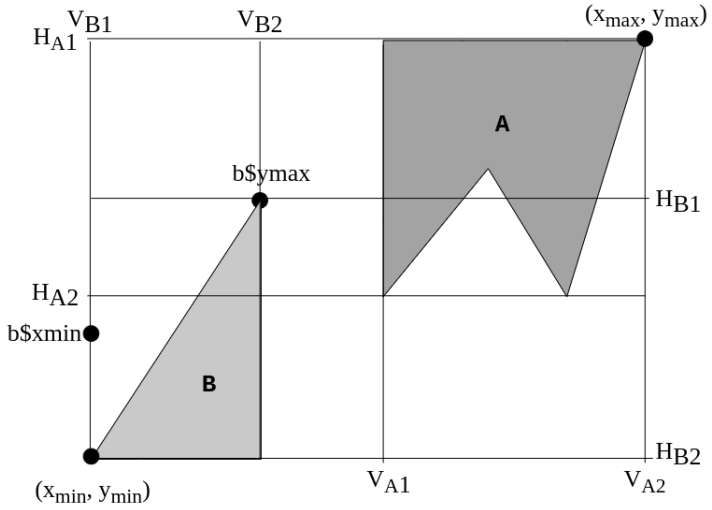


Figura 5 – Diagrama dos dois objetos

o objeto A, 2 para os quadrantes que contém o objeto B e 3 quando os dois objetos se intersectam em um mesmo grid.

Assim, para os objetos da Figura 5, a seguinte matriz OIM é gerada:

$$OIM = \begin{bmatrix} 0 & 0 & 1 \\ 2 & 0 & 1 \\ 2 & 0 & 0 \end{bmatrix}$$

Na fase de interpretação, a matriz OIM é utilizada para definir os relacionamentos direcionais entre os dois objetos.

2.2.3 Similaridades e Diferenças entre DRM e OIM

Diferente de muitos outros modelos de relacionamento, o DRM não é invertível. Isso implica em particular que, para duas regiões a e b, o DRM de b para a não pode ser determinado unicamente pelo DRM de a para b. Como consequência, o modelo DRM não satisfaz o critério de *converseness* [9]. Essa violação é vista como um sério problema, e por isso foi proposto o modelo OIM.

Uma vez que o DRM não é *closed under converse*, é necessário representar as direções cardiais das duas regiões a, b pelo DRM de ambos a para b e b para a. Diz-se que um par de DRMs (M_1, M_2) é consistente se houver duas regiões a, b tais que M_1 é o DRM de a para b e M_2 é o DRM de b para a. Cada par consistente de DRMs define uma relação binária, que é chamada uma relação bi-DRM. O modelo DRM possui o conceito de objeto de referência e objeto alvo, enquanto o OIM analisa os dois objetos de forma unificada, sem fazer essa distinção.

Assim como o DRM, o modelo OIM também é um modelo *tiling-based* que consiste de duas fases: a *tiling phase* e a *interpretation phase*. Dadas duas regiões a e b, a *tiling phase* computa uma matriz, denotada OIM(a,b), como a representação de baixo nível para a direção cardial de a para b. A *interpretation phase* então interpreta esta matriz como um conjunto de direções cardiais básicas, como N, NE. Também é demonstrado como definir predicados como *surrounds* e *strictly_north_cap_of* em bancos de dados espaciais baseado na interpretação.

Alguns destes predicados são, entretanto, contraintuitivos. Por exemplo, assim como o modelo OIM pode expressar sentenças verdadeiras como "Roma circunda o Vaticano", também suporta sentenças falsas como "O Vaticano circunda Roma". Isso porque a relação *surrounds* conforme definida é uma relação simétrica.

O trabalho de Li e Liu [10] compara os dois modelos e mostra que eles não são tão diferentes assim, uma vez que toda relação OIM é contida em uma única relação bi-DRM, e exceto para alguns casos (104 de 1677, ou seja, 6,2 %) toda relação do OIM é idêntica a uma relação bi-DRM.

3 Trabalhos Relacionados

Este capítulo resume as características principais de quatro trabalhos da literatura relacionados ao tema deste TCC, uma vez que descrevem implementações de relacionamentos direcionais em bancos de dados espaciais. As seções 3.1, 3.2, 3.3 e 3.4 descrevem cada um dos trabalhos. A seção 3.5 apresenta uma conclusão geral sobre os trabalhos discutidos.

3.1 Range Queries Involving Spatial Relations: A Performance Analysis

O estudo de Theodoridis e Papadias [7] mostra como consultas envolvendo relacionamentos espaciais em objetos do tipo região podem ser transformadas em consultas por abrangência e implementadas em SGBDs existentes. Constitui a primeira tentativa de modelar o desempenho de relações espaciais, uma vez que trabalhos anteriores focaram na utilização de janelas de consulta, nas quais é realizada a recuperação de objetos que compartilham pontos comuns com um determinado objeto ou área.

O trabalho utiliza MBRs e foca nos relacionamentos direcionais *east* e *northeast*, nas relações topológicas de adjacência (*meet*) e contenção (*inside*), além de algumas relações de distância. É realizada também uma análise de desempenho, comparando os dois métodos de indexação utilizados: B-trees e R-trees.

Assim, são propostos modelos analíticos para obter o custo de recuperação esperado destas relações espaciais, que se mostraram boas diretrizes para os otimizadores de consultas de SGBDs que suportam relações espaciais, a fim de estimar o custo destas consultas.

3.2 Supporting Direction Relations in Spatial Database Systems

O trabalho [11] define relações direcionais entre objetos de duas dimensões em diferentes níveis de resolução qualitativa, para atender as necessidades da aplicação. Mostra também como as relações definidas podem ser recuperadas de forma eficiente nos DBMSs existentes. O resultado deste trabalho é diretamente aplicável a Bancos de dados espaciais e GIS, onde a formalização de relacionamentos espaciais é crucial para interfaces de usuário e estratégias de otimização de consultas.

Utiliza o método baseado em projeção, no qual os relacionamentos direcionais são definidos usando linhas de projeção verticais aos eixos de coordenadas. A implementação

utiliza os MBRs dos objetos e é realizada de duas formas, com B-trees e R-trees, e é apresentada uma comparação de performance entre os dois, chegando à conclusão de que não é possível determinar qual estrutura de dados possui a performance melhor em todas as queries, mas sim definir de acordo com os seguintes fatores:

- O número e o tipo de restrições envolvidas na definição das relações de direção de interesse;
- O tamanho dos dados (ou seja, o tamanho dos MBRs primários);
- O tamanho da consulta (ou seja, o tamanho dos MBRs de referência).

Nas aplicações, poderia ser adicionado um otimizador que escolhe a estrutura mais adequada de acordo com a consulta inserida, considerando os fatores apontados acima.

3.3 On the Retrieval of Direction Relations using R-trees

Papadias et al. [12] descrevem como o método R-tree pode ser utilizado para armazenar e recuperar relacionamentos direcionais, usando uma variante chamada R*-tree, que consiste em agrupar os MBRs ao dividir a área global em partições com um nível mínimo ou nulo de sobreposição. De acordo com esta estratégia, "caixas"adjacentes são armazenadas em partições adjacentes e objetos que satisfazem certas relações topológicas são selecionadas rapidamente durante a etapa de filtragem.

Alguns dos relacionamentos direcionais definidos precisam de uma etapa de refinamento, enquanto outros podem ser encontrados utilizando apenas aproximações por MBR. Podem ser calculados relacionamentos direcionais entre qualquer combinação de objetos dos tipos região, linha e ponto. Foram implementados apenas alguns relacionamentos para teste, porém pode ser estendido para relacionamentos adicionais.

3.4 The Cardinal Direction relations and the rectangle algebra

O trabalho [13] propõe um novo modelo de representação de relacionamento de direções cardiais baseado na combinação dos modelos MBR e *Rectangle Algebra*. Ambos os modelos utilizam relacionamentos posicionais de dois triângulos, a diferença é que o *rectangle algebra* não descreve claramente o relacionamento direcional. São definidos os relacionamentos de direções cardiais possíveis, obtendo um novo método para representação de direções e um novo modelo que permite verificar a consistência de direções cardinais baseadas em MBR.

3.5 Considerações Finais

Analisando os trabalhos relacionados, pode-se concluir que nenhum destes trabalhos implementa modelos genéricos de detecção de relacionamentos direcionais. Assim, os desenvolvedores de aplicações que desejam utilizar relacionamentos direcionais precisam implementar suas próprias versões, o que é custoso, demorado e pode acarretar erros.

Deste modo, este TCC contribui neste aspecto, pois realiza implementações reais das matrizes que formalizam a identificação dos relacionamentos direcionais, na forma de uma biblioteca que ficará disponível no repositório do R (CRAN), além de conduzir testes experimentais que comparam a utilização dos modelos na identificação de relacionamentos direcionais em consultas espaciais.

4 Implementação dos Relacionamentos Direcionais

Este capítulo apresenta as implementações dos dois modelos estudados neste TCC, sendo que a seção 4.1 apresenta o Direction Relation Matrix (DRM), a seção 4.2 apresenta o Objects Interaction Matrix (OIM), e a seção 4.3 realiza uma análise experimental comparando os dois modelos. As implementações utilizam como base o pacote *sf* da linguagem R[1].

4.1 DRM

4.1.1 DRM Simples

O Código 4.1 contém a função que implementa o modelo DRM. A função drm() recebe três argumentos: *ref*, um objeto sfg do tipo *polygon* que representa o objeto de referência; *target*, um objeto sfg do tipo *polygon* que representa o objeto alvo; e *refined*, uma variável inteira que indica se deve ser calculado o DRM refinado (refined=1) ou DRM simples (refined=0). Em seguida, a função st_bbox do pacote sf é utilizada para obter o MBR de cada um dos objetos, no formato de uma lista contendo os valores de xmin, ymin, xmax e ymax (linhas 2 a 5). Assim, *mbr_target$xmin* se refere ao menor valor de x para o objeto alvo, por exemplo.

A matriz DRM é armazenada em um vetor com nove valores booleanos, tais que os valores *True* indicam as posições em que o objeto alvo intersecta cada uma das regiões do *grid* - formado a partir da extensão das retas do MBR do objeto de referência. Assim, o vetor é inicializado com *False* em todas as posições e em seguida são adicionados rótulos para cada um dos quadrantes (linhas 6 e 7). São então calculados os valores Booleanos para cada um dos quadrantes, seguindo a lógica da figura (linhas 9 a 32).

4.1.2 DRM Refinada

No Código 4.1, caso o parâmetro *refined* seja igual a zero, a função retorna a matriz DRM simples (linhas 34 a 36). Caso contrário, a função continua sendo executada e são calculados os 16 pontos (linhas 38 a 55), que são utilizados para formar os quadrantes correspondentes às partições (linhas 58 a 74). Com os quadrantes definidos, a é calculada a intersecção entre o objeto alvo e cada um dos quadrantes, bem como a distribuição da sua área nestes quadrantes, que é retornada como uma lista (linhas 76 a 78).

[1] <https://r-spatial.github.io/sf/>

```
drm <- function(ref, target, refined){
  mbr_ref <- st_bbox(ref)
  mbr_target <- st_bbox(target)
  x <- c(mbr_target$xmin, mbr_target$xmax)
  y <- c(mbr_target$ymin, mbr_target$ymax)
  vetor <- rep(c(FALSE), times=c(9))
  names(vetor) <- c("NW","N","NE","W","O","E", "SW","S","SE")

  vetor['NW'] <- mbr_target$xmin <= mbr_ref$xmin
                 & mbr_target$ymax >= mbr_ref$ymax
  vetor['N'] <- mbr_target$xmax >= mbr_ref$xmin
                 & mbr_target$xmin <= mbr_ref$xmax
                 & mbr_target$ymax >= mbr_ref$ymax
  vetor['NE'] <- mbr_target$xmax >= mbr_ref$xmax
                 & mbr_target$ymax >= mbr_ref$ymax
  vetor['W'] <- mbr_target$xmin <= mbr_ref$xmin
                 & mbr_target$ymax >= mbr_ref$ymin
                 & mbr_target$ymin <= mbr_ref$ymax
  vetor['O'] <- mbr_target$xmax >= mbr_ref$xmin
                 & mbr_target$xmin <= mbr_ref$xmax
                 & mbr_target$ymax >= mbr_ref$ymin
                 & mbr_target$ymin <= mbr_ref$ymax
  vetor['E'] <- mbr_target$xmax >= mbr_ref$xmax
                 & mbr_target$ymax >= mbr_ref$ymin
                 & mbr_target$ymin <= mbr_ref$ymax
  vetor['SW'] <- mbr_target$xmin <= mbr_ref$xmin
                 & mbr_target$ymin <= mbr_ref$ymin
  vetor['S'] <- mbr_target$xmax >= mbr_ref$xmin
                 & mbr_target$xmin <= mbr_ref$xmax
                 & mbr_target$ymin <= mbr_ref$ymin
  vetor['SE'] <- mbr_target$xmax >= mbr_ref$xmax
                 & mbr_target$ymin <= mbr_ref$ymin

  if(!refined){
    return(names(which(vetor==TRUE, arr.ind=TRUE)))
  }

  p1 <- c(mbr_target$xmin, mbr_target$ymax)
  p2 <- c(mbr_ref$xmin, mbr_target$ymax)
  p3 <- c(mbr_ref$xmax, mbr_target$ymax)
  p4 <- c(mbr_target$xmax, mbr_target$ymax)
  p5 <- c(mbr_target$xmin, mbr_ref$ymax)
```

```
43    p6 <- c(mbr_ref$xmin, mbr_ref$ymax)
44    p7 <- c(mbr_ref$xmax, mbr_ref$ymax)
45    p8 <- c(mbr_target$xmax, mbr_ref$ymax)
46    p9 <- c(mbr_target$xmin, mbr_ref$ymin)
47    p10 <- c(mbr_ref$xmin, mbr_ref$ymin)
48    p11 <- c(mbr_ref$xmax, mbr_ref$ymin)
49    p12 <- c(mbr_target$xmax, mbr_ref$ymin)
50    p13 <- c(mbr_target$xmin, mbr_target$ymin)
51    p14 <- c(mbr_ref$xmin, mbr_target$ymin)
52    p15 <- c(mbr_ref$xmax, mbr_target$ymin)
53    p16 <- c(mbr_target$xmax, mbr_target$ymin)
54
55    pontos_quadrantes <- st_multipoint(rbind(p1, p2, p3, p4, p5,
56          p6, p7, p8, p9, p10, p11, p12, p13, p14, p15, p16))
57
58    quadrantes <- vector()
59    quad_nw <- st_polygon(list(rbind(p1, p2, p6, p5, p1)))
60    quad_n <- st_polygon(list(rbind(p2, p3, p7, p6, p2)))
61    quad_ne <- st_polygon(list(rbind(p3, p4, p8, p7, p3)))
62    quad_w <- st_polygon(list(rbind(p5, p6, p10, p9, p5)))
63    quad_o <- st_polygon(list(rbind(p6, p7, p11, p10, p6)))
64    quad_e <- st_polygon(list(rbind(p7, p8, p12, p11, p7)))
65    quad_sw <- st_polygon(list(rbind(p9, p10, p14, p13, p9)))
66    quad_s <- st_polygon(list(rbind(p10, p11, p15, p14, p10)))
67    quad_se <- st_polygon(list(rbind(p11, p12, p16, p15, p11)))
68
69    quadrantes <- list(quad_nw, quad_n, quad_ne, quad_w, quad_o,
70                       quad_e, quad_sw, quad_s, quad_se)
71    names(quadrantes) <- c("NW","N","NE","W","O","E","SW","S","SE")
72
73    list_quads <- st_sfc(list(quad_nw, quad_n, quad_ne, quad_w,
74    quad_o, quad_e, quad_sw, quad_s, quad_se))
75
76    inter <- st_intersection(list_quads, target)
77    area <- st_area(inter)/ st_area(target)
78    return(area)
79  }
```

Código 4.1 – Função DRM

4.2 OIM

4.2.1 Objects Interaction Grid: Capturando a interação dos objetos

Seja min_x igual ao mínimo entre min_{ax} e min_{bx}, max_x igual ao máximo entre max_{ax} e max_{bx}, o código 4.2 implementa a primeira parte do modelo, que retorna o *Objects Interaction Grid* (OIG). A função recebe dois objetos a e b como parâmetros, calcula os MBRs de ambos e os valores mínimos e máximos para X e Y (linhas 2 a 8). Em seguida, são definidos dois segmentos horizontais e verticais para cada objeto: H_{A1}, H_{A2}, V_{A1}, V_{A2}, H_{B1}, H_{B2}, V_{B1} e V_{B2} (linhas 10 a 25).

A partir dos segmentos encontrados, pode-se obter os polígonos correspondentes às 9 partições (linhas 27 a 102). Por fim, os polígonos são armazenados em uma lista, que é retornada pela função (linhas 99 a 104).

```
1  oig <- function(a, b){
2    mbr_a <- st_bbox(a)
3    mbr_b <- st_bbox(b)
4
5    xmin <- min(mbr_a$xmin, mbr_b$xmin)
6    xmax <- max(mbr_a$xmax, mbr_b$xmax)
7    ymin <- min(mbr_a$ymin, mbr_b$ymin)
8    ymax <- max(mbr_a$ymax, mbr_b$ymax)
9
10   ha1 <- st_linestring(rbind(c(xmin, mbr_a$ymin),
11                              c(xmax, mbr_a$ymin)))
12   ha2 <- st_linestring(rbind(c(xmin, mbr_a$ymax),
13                              c(xmax, mbr_a$ymax)))
14   hb1 <- st_linestring(rbind(c(xmin, mbr_b$ymin),
15                              c(xmax, mbr_b$ymin)))
16   hb2 <- st_linestring(rbind(c(xmin, mbr_b$ymax),
17                              c(xmax, mbr_b$ymax)))
18   va1 <- st_linestring(rbind(c(mbr_a$xmin, ymin),
19                              c(mbr_a$xmin, ymax)))
20   va2 <- st_linestring(rbind(c(mbr_a$xmax, ymin),
21                              c(mbr_a$xmax, ymax)))
22   vb1 <- st_linestring(rbind(c(mbr_b$xmin, ymin),
23                              c(mbr_b$xmin, ymax)))
24   vb2 <- st_linestring(rbind(c(mbr_b$xmax, ymin),
25                              c(mbr_b$xmax, ymax)))
26
27   pol11 <- st_polygon(list(rbind(
28     c(xmin, ymax),
```

```
29      c(mbr_b$xmax, ymax),
30      c(mbr_b$xmax, mbr_b$ymax),
31      c(xmin, mbr_b$ymax),
32      c(xmin, ymax)
33    )))
34
35    pol12 <- st_polygon(list(rbind(
36      c(mbr_b$xmax, ymax),
37      c(mbr_a$xmin, ymax),
38      c(mbr_a$xmin, mbr_b$ymax),
39      c(mbr_b$xmax, mbr_b$ymax),
40      c(mbr_b$xmax, ymax)
41    )))
42
43    pol13 <- st_polygon(list(rbind(
44      c(mbr_a$xmin, ymax),
45      c(xmax, ymax),
46      c(xmax, mbr_b$ymax),
47      c(mbr_a$xmin, mbr_b$ymax),
48      c(mbr_a$xmin, ymax)
49    )))
50
51    pol21 <- st_polygon(list(rbind(
52      c(xmin, mbr_b$ymax),
53      c(mbr_b$xmax, mbr_b$ymax),
54      c(mbr_b$xmax, mbr_a$ymin),
55      c(xmin, mbr_a$ymin),
56      c(xmin, mbr_b$ymax)
57    )))
58
59    pol22 <- st_polygon(list(rbind(
60      c(mbr_b$xmax, mbr_b$ymax),
61      c(mbr_a$xmin, mbr_b$ymax),
62      c(mbr_a$xmin, mbr_a$ymin),
63      c(mbr_b$xmax, mbr_a$ymin),
64      c(mbr_b$xmax, mbr_b$ymax)
65    )))
66
67    pol23 <- st_polygon(list(rbind(
68      c(mbr_a$xmin, mbr_b$ymax),
69      c(xmax, mbr_b$ymax),
70      c(xmax, mbr_a$ymin),
```

```
71      c(mbr_a$xmin, mbr_a$ymin),
72      c(mbr_a$xmin, mbr_b$ymax)
73    )))
74
75    pol31 <- st_polygon(list(rbind(
76      c(xmin, mbr_a$ymin),
77      c(mbr_b$xmax, mbr_a$ymin),
78      c(mbr_b$xmax, ymin),
79      c(xmin, ymin),
80      c(xmin, mbr_a$ymin)
81    )))
82
83    pol32 <- st_polygon(list(rbind(
84      c(mbr_b$xmax, mbr_a$ymin),
85      c(mbr_a$xmin, mbr_a$ymin),
86      c(mbr_a$xmin, ymin),
87      c(mbr_b$xmax, ymin),
88      c(mbr_b$xmax, mbr_a$ymin)
89    )))
90
91    pol33 <- st_polygon(list(rbind(
92      c(mbr_a$xmin, mbr_a$ymin),
93      c(xmax, mbr_a$ymin),
94      c(xmax, ymin),
95      c(mbr_a$xmin, ymin),
96      c(mbr_a$xmin, mbr_a$ymin)
97    )))
98
99    list_polygons <- list(
100     pol11, pol12, pol13,
101     pol21, pol22, pol23,
102     pol31, pol32, pol33)
103
104   return(list_polygons)
105 }
```

Código 4.2 – Função OIG

A função *oim_matrix* do código 4.3 utiliza o OIG encontrado para calcular a Matrix OIM. Inicialmente são obtidas as partições em que os objetos A e B intersectam cada polígono do grid (linhas 3 a 5). Em seguida, os quadrantes que contém o objeto *a* são multiplicados por 1 e os quadrantes que contém o objeto *b* são multiplicados por 2, retornando assim a matrix resultante (linhas 7 a 10).

```
1
2  oim_matrix <- function(a, b){
3    matrix_polygons <- st_sfc(matrix(oig(a, b), ncol=3, nrow=3))
4    int_a_matrix <- st_relate(a, matrix_polygons, 'T********')
5    int_b_matrix <- st_relate(b, matrix_polygons, 'T********')
6
7    oim_matrix <- t(matrix(1*as.matrix(int_a_matrix)
8                    + 2*as.matrix(int_b_matrix), ncol=3, nrow=3))
9
10   return(oim_matrix)
11 }
```

Código 4.3 – Função OIM Matrix

Com a matriz OIM obtida anteriormente, a função *oim* do código 4.4 obtém os relacionamentos cardinais entre os objetos a e b. Inicialmente é recuperada a localização de cada um dos objetos na matriz OIM (linhas 3 a 6), em seguida é inicializada uma lista para armazenar as direções, e uma lógica com condicionais para obter todas as direções do objeto a em relação ao objeto b (linhas 13 a 37). Por fim, são removidos possíveis direções duplicadas e a lista é retornada (linhas 38 e 39).

```
1  oim <- function(a, b){
2
3    m <- oim_matrix(a, b)
4
5    loc_a <- which(m==1 | m ==3, arr.ind=TRUE)
6    loc_b <- which(m==2 | m == 3, arr.ind=TRUE)
7
8    ai <- loc_a[,1]
9    bi <- loc_b[,1]
10   aj <- loc_a[,2]
11   bj <- loc_b[,2]
12
13   cd <- list()
14
15   for(x in 1:length(loc_b[, 2])){
16     for(y in 1:length(loc_a[, 2])){
17       if(bi[x] < ai[y] & bj[x] == aj[y]){
18         cd <- append(cd, 'N')
19       } else if (bi[x] < ai[y] & bj[x] < aj[y]){
20         cd <- append(cd, 'NW')
21       } else if (bi[x] == ai[y] & bj[x] < aj[y]){
22         cd <- append(cd, 'W')
```

```
23        } else if (bi[x] > ai[y] & bj[x] < aj[y]){
24          cd <- append(cd, 'SW')
25        } else if (bi[x] > ai[y] & bj[x] == aj[y]){
26          cd <- append(cd, 'S')
27        } else if (bi[x] > ai[y] & bj[x] > aj[y]){
28          cd <- append(cd, 'SE')
29        } else if (bi[x] == ai[y] & bj[x] > aj[y]){
30          cd <- append(cd, 'E')
31        } else if (bi[x] < ai[y] & bj[x] > aj[y]){
32          cd <- append(cd, 'NE')
33        } else if (bi[x] == ai[y] & bj[x] == aj[y]){
34          cd <- append(cd, 'O')
35        }
36      }
37    }
38    cd <- unlist(unique(cd))
39    return(cd)
40  }
```

Código 4.4 – Função OIM

4.3 Análise Experimental dos Modelos de Relacionamentos Direcionais

Para realizar a análise experimental e comparar os dois modelos, foram utilizados dados do OpenStreetMap [14] extraídos no dia 28 de janeiro de 2022, sendo 104.465 objetos do tipo região, correspondentes a edifícios no estado de São Paulo. Para diminuir o número de objetos, foram considerados dois tipos de edifícios: *amenity* e *leisure*. *Amenities* são facilidades como bancos, farmácias, cafés, estacionamentos e escolas. *Leisures* são edifícios relacionados a esportes e lazer, como parques, estádios, piscinas e academias. Um dos objetos foi selecionado como objeto de referência, e corresponde ao centróide do estado de São Paulo, com raio de 1 km.

O código 4.5 contém a leitura do conjunto de dados utilizado (linha 1), a definição do objeto de referência (linhas 3 a 40), e as funções utilizadas para executar os modelos sobre o objeto de referência e cada objeto alvo contido no conjunto de dados: DRM simples (linhas 42 a 48), DRM refinado (linhas 50 a 56) e OIM (linhas 58 a 64).

```
1  nc <- st_read("./leisures_and_amenities_inside_sp.shp")
2
3  d <- data.frame(a = 1:2)
4  d$geom <- c("POLYGON((-5422833.46948292
```

```
-2545421.00183794,-5422852.68420252
-2545616.09215996,-5422909.58995041
-2545803.6852703,-5423001.99987062
-2545976.57207096,-5423126.36270173
-2546128.10861913,-5423277.8992499
-2546252.47145024,-5423450.78605056
-2546344.88137045,-5423638.37916091
-2546401.78711834,-5423833.46948292
-2546421.00183794,-5424028.55980494
-2546401.78711834,-5424216.15291529
-2546344.88137045,-5424389.03971594
-2546252.47145024,-5424540.57626411
-2546128.10861913,-5424664.93909522
-2545976.57207096,-5424757.34901543
-2545803.6852703,-5424814.25476332
-2545616.09215996,-5424833.46948292
-2545421.00183794,-5424814.25476332
-2545225.91151592,-5424757.34901543
-2545038.31840557,-5424664.93909522
-2544865.43160492,-5424540.57626411
-2544713.89505675,-5424389.03971594
-2544589.53222564,-5424216.15291529
-2544497.12230543,-5424028.55980494
-2544440.21655754,-5423833.46948292
-2544421.00183794,-5423638.37916091
-2544440.21655754,-5423450.78605056
-2544497.12230543,-5423277.8992499
-2544589.53222564,-5423126.36270173
-2544713.89505675,-5423001.99987062
-2544865.43160492,-5422909.58995041
-2545038.31840557,-5422852.68420252
-2545225.91151592,-5422833.46948292
-2545421.00183794))")

df <- st_as_sf(d, wkt = "geom")
ref <- df$geom[[1]]

drm_for_loop <- function(ref, nc){
  result <- list()
  for(i in 1:length(nc$geom)){
    result <- c(result, drm(ref, nc$geom[[i]], 0))
  }
```

```
47   return(result)
48 }
49
50 drm_refin_for_loop <- function(ref, nc){
51   result <- list()
52   for(i in 1:length(nc$geom)){
53     result <- c(result, drm(ref, nc$geom[[i]], 1))
54   }
55   return(result)
56 }
57
58 oim_for_loop <- function(ref, nc){
59   result <- list()
60   for(i in 1:length(nc$geom)){
61     result <- c(result, oim(ref, nc$geom[[i]]))
62   }
63   return(result)
64 }
```

Código 4.5 – Análise Experimental

O pacote Microbenchmark [15] foi utilizado para comparar o tempo de execução das funções dos três modelos: DRM, DRM Refinado e OIM, conforme mostrado no Código 4.6. A função *microbenchmark()* executa as funções em um número determinado de vezes, e gera uma tabela com os tempos de execução. Os parâmetros são as três funções definidas anteriormente *drm_for_loop*, *drm_refin_for_loop* e *oim_for_loop*, aplicadas sobre o objeto de referência e objetos alvo, além de um parâmetro *times* que define o número de execuções, neste caso foram realizadas 10 execuções.

```
1 mbm <- microbenchmark("DRM"=drm_for_loop(ref,nc),
2                       "DRM Refinado"=drm_refin_for_loop(ref,nc),
3                       "OIM"=oim_for_loop(ref,nc),
4                       times=10)
5 mbm
6 autoplot(mbm)
```

Código 4.6 – Microbenchmark

Após a execução, foram obtidos os tempos de execução mostrados na Figura 6, em que *neval* é o número de execuções (neste caso 10), além dos tempos mínimo, quartil inferior, médio, a mediana, quartil superior e máximo. Nesta tabela é possível observar que o modelo OIM possui um tempo mais elevado em todas as métricas. Utilizando a função *autoplot()* da biblioteca Ggplot2 [16] foi gerado o gráfico da Figura 7, para ilustrar a distribuição dos tempos de execução.

```
Unit: seconds
         expr      min       lq     mean   median       uq      max neval
          DRM 215.0945 215.9388 232.7639 227.1037 237.7174 271.4042    10
 DRM Refinado 815.1215 815.4676 858.9119 818.2896 923.9668 972.0106    10
          OIM 824.0431 891.4869 922.8149 946.5078 961.4781 990.6982    10
```

Figura 6 – Comparação dos tempos de execução dos modelos

Analisando o *ViolinPlot* da Figura 7 pode-se perceber uma grande divergência entre os tempos de execução da implementação do modelo DRM em relação aos modelos DRM refinado e OIM. Isso indica que o cálculo das áreas realizado no DRM refinado adiciona um alto custo computacional se comparado ao DRM simples, uma vez que o tempo médio salta de 232,76 segundos para 858,91 segundos (269% de aumento). O modelo OIM se mostrou ainda mais custoso, pois apresentou uma média de 922,81 segundos, e um número maior de execuções com tempo mais elevado do que o DRM Refinado. Isso se deve às operações realizadas para obter os vértices e posteriormente os polígonos dos quadrantes (linhas 10 a 102 do Código 4.2), que não ocorrem no modelo DRM.

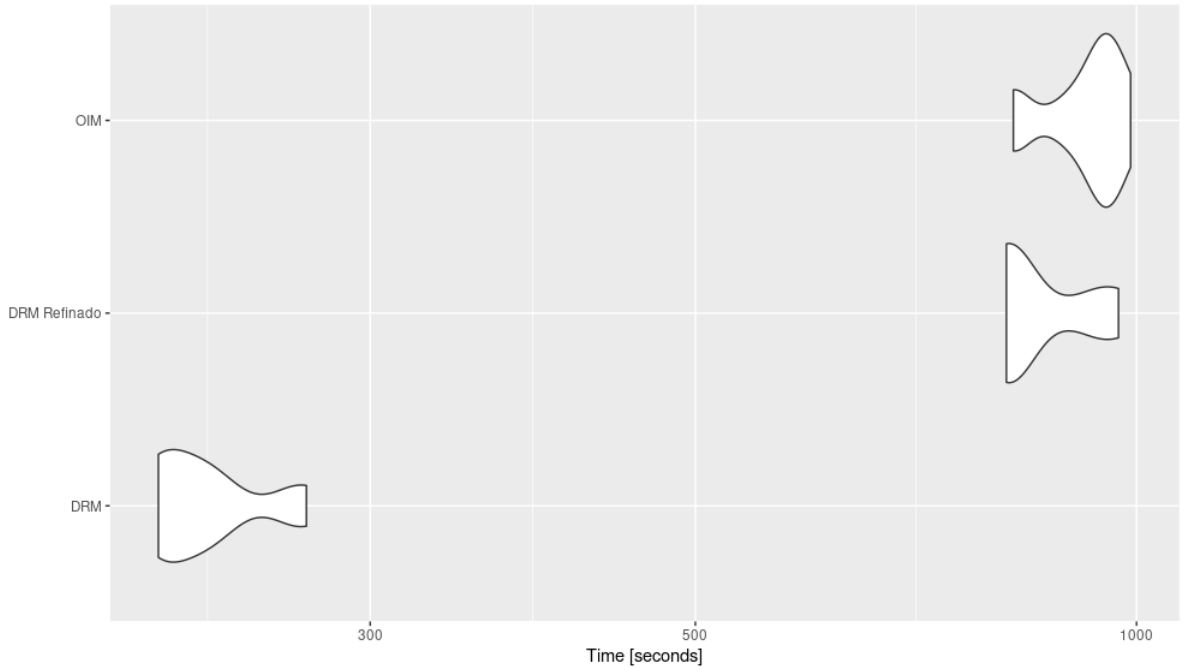


Figura 7 – Gráfico de comparação dos modelos

Deste modo, o modelo DRM é o mais simples e com menor tempo de execução, podendo ser utilizado em casos que necessitam de um tempo de resposta rápido e com menor

criticidade em relação à precisão, sem considerar a área dos objetos. O modelo DRM pode ser interessante quando a área dos objetos varia muito e possui grande influência sobre as direções que devem ser retornadas. Entretanto, para um pequeno subconjunto (104 de 1.677, ou 6,2%) [10] dos possíveis relacionamentos entre objetos espaciais, o modelo OIM apresenta resultados mais precisos do que o DRM. Assim, dependendo da criticidade da aplicação, e para um conjunto de dados reduzido, pode ser mais interessante utilizar o modelo OIM, assegurando a precisão apesar do tempo superior empregado no processamento.

5 Conclusões

Este capítulo encerra o trabalho atual, comentando as conclusões sobre o desenvolvimento e os resultados obtidos. A seção 5.1 apresenta as considerações finais sobre o trabalho desenvolvido, a seção 5.2 descreve orientações para trabalhos futuros.

5.1 Considerações Finais

Com o crescimento da utilização de dados espaciais nas mais variadas áreas do conhecimento, faz-se necessário buscar novas formas para obter os relacionamentos entre diferentes objetos no espaço. Há muitas abordagens descritas na literatura, porém muitas acabam não sendo implementadas para utilização em aplicações reais.

Dentro deste contexto, neste trabalho foram estudados e implementados os modelos DRM e OIM, com o objetivo de comparar os tempos de execução da implementação de cada um deles. De acordo com os resultados encontrados na análise, é possível observar que o DRM simples foi o mais rápido. Desse modo, este modelo se torna mais adequado para aplicações de banco de dados espaciais que precisam processar rapidamente consultas espaciais com predicados relacionais. Todavia, o modelo OIM também pode ser utilizado em casos específicos, apesar do tempo de execução elevado, quando há a necessidade de uma precisão maior dos relacionamentos direcionais.

5.2 Trabalhos Futuros

As implementações realizadas neste trabalho possibilitam a utilização dos modelos DRM e OIM de forma prática, aplicando-os para definir relacionamentos direcionais entre objetos reais. Podem ser aplicadas as seguintes atividades futuras para estender o trabalho realizado:

- **Vetorizar as funções dos modelos:** uma boa prática de programação na linguagem R é trabalhar com funções vetorizadas, ou seja, a função opera sobre todos os elementos de um vetor sem a necessidade de utilizar laços de repetição para acessar cada elemento. Isso torna o código mais conciso, legível e menos suscetível a erros. Assim, as funções apresentadas neste trabalho podem ser replicadas para operar sobre vetores.

- **Executar testes de desempenho mais extensos:** os testes realizados podem ser estendidos para operar sobre conjuntos de dados maiores, bem como simular casos

de uso cotidianos, com um menor volume de dados mas um limite estabelecido para o tempo de resposta (assumindo que o retorno das funções seria utilizado em uma aplicação real, por exemplo).

- **Adicionar novos modelos de identificação de relacionamentos direcionais:** além dos modelos DRM e OIM, podem ser analisados e implementados outros modelos definidos teoricamente na literatura que não possuam implementações.

Referências

1 GüTING, R. H. An introduction to spatial database systems. *The VLDB Journal*, v. 3, n. 4, p. 357–399, 1994. Citado 3 vezes nas páginas 19, 23 e 24.

2 SCHNEIDER, M.; BEHR, T. Topological relationships between complex spatial objects. *ACM Trans. on Database Systems*, v. 31, n. 1, p. 39–81, 2006. Citado 2 vezes nas páginas 19 e 24.

3 GEOS contributors. *GEOS coordinate transformation software library*. [S.l.], 2021. Disponível em: <https://libgeos.org/>. Citado na página 20.

4 GOYAL, R. K.; EGENHOFER, M. J. The direction-relation matrix: A representation for directions relations between extended spatial objects. In: *The Annual Assembly and the Summer Retreat of University Consortium for Geographic Information Systems Science*. [S.l.: s.n.], 1997. Citado 2 vezes nas páginas 20 e 25.

5 CHEN, T. et al. The objects interaction matrix for modeling cardinal directions in spatial databases. In: *Database Systems for Advanced Applications, 15th International Conference*. [S.l.: s.n.], 2010. v. 15, n. 1, p. 297–302. Citado na página 20.

6 PEBESMA, E. Simple Features for R: Standardized Support for Spatial Vector Data. *The R Journal*, v. 10, n. 1, p. 439–446, 2018. Disponível em: <https://doi.org/10.32614/RJ-2018-009>. Citado na página 21.

7 THEODORIDIS, Y.; PAPADIAS, D. Range queries involving spatial relations: A performance analysis. In: *Int. Conf. on Spatial Information Theory*. [S.l.: s.n.], 1995. p. 537–551. Citado 2 vezes nas páginas 24 e 31.

8 FRANK, A. Qualitative spatial reasoning: Cardinal directions as an example. *International Journal of Geographical Information Science*, v. 10, p. 269–290, 04 1996. Citado na página 25.

9 SCHNEIDER, M. et al. Cardinal directions between complex regions. *ACM Trans. on Database Systems*, v. 37, n. 2, 2012. Citado na página 29.

10 LI, S.; LIU, W. Cardinal directions: a comparison of direction relation matrix and objects interaction matrix. *Int. Journal of Geographical Information Science*, v. 29, n. 2, p. 194–216, 2015. Citado 2 vezes nas páginas 29 e 46.

11 THEODORIDIS, Y.; PAPADIAS, D.; STEFANAKIS, E. Supporting direction relations in spatial database systems. In: *Int. Symp. on Spatial Data Handling*. [S.l.: s.n.], 1996. Citado na página 31.

12 PAPADIAS, D.; THEODORIDIS, Y.; SELLIS, T. The retrieval of direction relations using R-trees. In: *Int. Conf. on Database and Expert Systems Applications*. [S.l.: s.n.], 1994. p. 173–182. Citado na página 32.

13 LIU, Y.-S.; HAO, Z.-X. The cardinal direction relations and the rectangle algebra. In: *Int. Conf. on Machine Learning and Cybernetics*. [S.l.: s.n.], 2005. p. 3115–3118. Citado na página 32.

14 OpenStreetMap contributors. *Planet dump retrieved from https://planet.osm.org* . 2017. <https://www.openstreetmap.org>. Citado na página 42.

15 MERSMANN, O. *microbenchmark: Accurate Timing Functions*. [S.l.], 2021. R package version 1.4.9. Disponível em: <https://CRAN.R-project.org/package=microbenchmark>. Citado na página 44.

16 WICKHAM, H. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016. ISBN 978-3-319-24277-4. Disponível em: <https://ggplot2.tidyverse.org>. Citado na página 44.