UNIVERSIDADE FEDERAL DE CAMPINA GRANDE

CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA

UNIDADE ACADÊMICA DE SISTEMAS E COMPUTAÇÃO

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

CARLOS VINICIUS ALVES MINERVINO PONTES

SPATIAL PATTERN MATCHING WITH QUANTITATIVE AND QUALITATIVE CONSTRAINTS

CAMPINA GRANDE – PB

2024

Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Coordenação de Pós-Graduação em Ciência da Computação

Spatial Pattern Matching with Quantitative and Qualitative Constraints

Carlos Vinicius Alves Minervino Pontes

Dissertação submetida à Coordenação do Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Campina Grande - Campus I - como parte dos requisitos necessários para obtenção do grau de Mestre em Ciência da Computação.

Área de Concentração: Ciência da Computação
Linha de Pesquisa: Sistemas de Informação Geográfica

Cláudio E. C. Campelo
(Orientador)

Campina Grande, Paraíba, Brasil

P814s Pontes, Carlos Vinicius Alves Minervino.
Spatial Pattern Matching with Quantitative and Qualitative Constraints / Carlos Vinicius Alves Minervino Pontes. – Campina Grande, 2024.
114 f. : il. color.

Dissertação (Mestrado em Ciência da Computação) – Universidade Federal de Campina Grande, Centro de Engenharia Elétrica e Informática, 2024.
"Orientação: Prof. Dr. Claúdio Elízio Calazans Campelo".
Referências.

1. Geographic Information Retrieval. 2. Geo-Textual Retrieval. 3. Spatial Keyword Search. 4. Spatial Pattern Matching. 5. POI Search. 6. Qualitative Spatial Reasoning. 7. Topological Relations. I. Campelo, Cláudio Elízio Calazans. II. Título.

CDU 004.78:025.4.036:911(043)

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECÁRIA MEIRE EMANUELA DA SILVA MELO CRB-15/568

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
POS-GRADUACAO EM CIENCIA DA COMPUTACAO
Rua Aprígio Veloso, 882, Edifício Telmo Silva de Araújo, Bloco CG1, - Bairro Universitário, Campina Grande/PB, CEP 58429-900
Telefone: 2101-1122 - (83) 2101-1123 - (83) 2101-1124
Site: http://computacao.ufcg.edu.br - E-mail: secretaria-copin@computacao.ufcg.edu.br / copin@copin.ufcg.edu.br

FOLHA DE ASSINATURA PARA TESES E DISSERTAÇÕES

CARLOS VINICIUS ALVES MINERVINO

SPATIAL PATTERN MATCHING WITH QUANTITATIVE AND QUALITATIVE CONSTRAINTS

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação como pré-requisito para obtenção do título de Mestre em Ciência da Computação.

Aprovada em: 31/07/2024

Prof. Dr. CLÁUDIO ELÍZIO CALAZANS CAMPELO, Orientador, UFCG

Prof. Dr. DIMAS CASSIMIRO DO NASCIMENTO FILHO, Examinador Interno, UFAPE

Prof. Dr. DANIEL DOS SANTOS KASTER, Examinador Externo, UEL

Documento assinado eletronicamente por **CLAUDIO ELIZIO CALAZANS CAMPELO**, **PROFESSOR(A) DO MAGISTERIO SUPERIOR**, em 01/08/2024, às 10:58, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

Documento assinado eletronicamente por **Daniel dos Santos Kaster**, **Usuário Externo**, em 01/08/2024, às 18:59, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

Documento assinado eletronicamente por **Dimas Cassimiro do Nascimento Filho**, **Usuário Externo**, em 01/08/2024, às 19:59, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

A autenticidade deste documento pode ser conferida no site https://sei.ufcg.edu.br/autenticidade, informando o código verificador **4651153** e o código CRC **C78B374B**.

Resumo

Buscas geo-textuais envolvem palavras-chave e restrições de localização espacial. Um exemplo é a busca por Pontos de Interesse (POIs), como escolas e supermercados, em aplicativos como Google Maps. Notavelmente, a maioria dos sistemas existentes realiza buscas separadas para cada tipo de POI. Estudos recentes propuseram mecanismos para recuperar grupos de objetos geo-textuais heterogêneos, espacialmente próximos e relevantes a um conjunto de palavras-chave. Por exemplo, um tipo de busca chamado Correspondência de Padrão Espacial, do inglês *Spatial Pattern Matching* (SPM), recupera grupos de POIs ou outros objetos geo-textuais com base em padrões espaciais com palavras-chaves e limites de distância, porém não considera requisitos qualitativos, como a conectividade entre objetos. Assim sendo, algoritmos SPM não podem resolver de forma eficiente consultas tais como "encontrar *shoppings* que contenham uma academia de musculação em seu interior".

Nesse sentido, esta dissertação investiga a "Correspondência de Padrão Espacial Quantitativo e Qualitativo" (CPEQQ), um tipo mais flexível de busca geo-textual com palavras-chave, restrições de distância, relação topológica e exclusão entre objetos geo-textuais buscados. Propõe-se uma formalização matemática e uma abordagem com três estratégias eficientes de solução para consultas CPEQQ.

A primeira solução proposta, QQESPM-Quadtree, é independente de bancos de dados espaciais e usa a indexação IL-Quadtree em disco. A segunda, QQESPM-Elastic, converte o padrão espacial da busca em consultas espaciais nativas do Elasticsearch. A terceira, QQESPM-SQL, transforma os requisitos espaço-textuais da busca CPEQQ em uma única consulta SQL eficiente, utilizando funções e indexação espaciais no PostgreSQL.

Experimentos com dados de POIs de Londres compararam a eficácia e eficiência das três soluções propostas para o tipo de busca QQ-SPM. Os resultados mostraram a eficácia da formalização e abordagem propostas. A solução QQESPM-SQL destacou-se em escalabilidade por apresentar tempos de execução robustos com conjuntos de dados maiores. Entretanto, QQESPM-Quadtree e QQESPM-Elastic mostraram vantagens em alguns cenários específicos.

Abstract

Geo-textual searches involve keywords and spatial location restrictions. One example is the search for Points of Interest (POIs), such as schools and supermarkets, in applications such as Google Maps. Notably, most systems perform separate searches for each type of POI. Recent studies have proposed mechanisms to retrieve groups of geo-textual heterogeneous objects, closely located and relevant to a set of keywords. The Spatial Pattern Matching (SPM) query retrieves groups of POIs or other geo-textual objects based on spatial patterns with keywords and distance thresholds, although it does not consider qualitative requirements such as connectivity between objects. Consequently, SPM algorithms cannot efficiently solve queries such as "finding shopping malls that contain a training gym inside".

In this sense, this dissertation investigates "Quantitative and Qualitative Spatial Pattern Matching" (QQ-SPM), a more flexible type of geo-textual search with keywords, distance, topological and exclusion constraints between the searched geo-textual objects. A mathematical formalization and an efficient approach composed of three solution strategies for QQ-SPM searches are proposed in this research.

The first proposed solution, QQESPM-Quadtree, is independent of spatial databases and uses on-disk IL-Quadtree indexing. The second, QQESPM-Elastic, converts the spatial pattern of the search into native spatial Elasticsearch queries. The third, QQESPM-SQL, transforms the spatio-textual search requirements into a single and efficient SQL query, employing spatial functions and indexing in PostgreSQL.

Experiments using a dataset of POIs from London compared the effectiveness and efficiency of the three proposed solutions for QQ-SPM queries. The results confirmed the effectiveness of the proposed formalization and approach. The QQESPM-SQL solution excelled in scalability by presenting robust execution times for larger datasets. However QQESPM-Quadtree and QQESPM-Elastic presented advantages for some specific search scenarios.

Agradecimentos

Agradeço primeiramente à Deus, por iluminar meu caminho, e por sempre estar abrindo portas.

À minha amada esposa Julyanne pelo companheirismo e apoio, e por ser uma leiga porém excelente ouvinte.

Ao meu pai, eterno professor de Matemática e exemplo de vida. À minha mãe, por todo carinho. Aos meus irmãos por todo amor.

Aos meus orientadores, professores Dr. Cláudio Campelo e Dr. Maxwell Guimarães de Oliveira, pela paciência, compreensão, e excelente orientação.

À todos os meus colegas de laboratório do LACINA, em especial Dr. Salatiel, pelo companheirismo e compartilhamento de conhecimentos.

Aos membros da banca, professores Dr. Daniel Kaster e Dr. Dimas Cassimiro, pelos valiosos elogios, sugestões e críticas construtivas.

À coordenação de aos demais integrantes do PPGCC, pelo excelente apoio a mim e a todos os demais estudantes.

Finalmente, à todos os que, direta ou indiretamente, me apoiaram durante o mestrado ou durante etapas anteriores da minha vida, por terem contribuído com o meu crescimento acadêmico e pessoal.

Contents

List of Acronyms

POI - *Point of Interest*

SPM - *Spatial Pattern Matching*

QQ-SPM - *Quantitative and Qualitative Spatial Pattern Matching*

QQESPM - *Quantitative and Qualitative Efficient Spatial Pattern Matching*

QSR - *Qualitative Spatial Reasoning*

IR - *Information Retrieval*

CoSKQ - *Collective Spatial Keyword Query*

NPQ - *Neighborhood Preference Query*

List of Figures

List of Tables

List of Source Codes

Chapter 1

Introduction

With the emergence and popularization of technologies such as GPS, mobile Internet, Artificial Intelligence, and location-based services, a significant amount of geo-textual data is being generated and consumed daily[24, 27, 28, 38, 50, 52, 81, 120]. The smartphone era has led to a growing demand for keyword search systems, and the need for efficient indexing systems and algorithms capable of processing this large volume of data properly[24, 61].

Geo-tagged microblog posts and web pages related to entities with physical locations are examples of geo-textual content, encompassing both textual and geographical information, constantly being generated[28, 38]. Additionally, the inclusion of new Points of Interest (POIs), which are places of social function such as hospitals, schools, and banks, in the maps' life cycle is a continuous geo-textual data generation process[80]. As outlined in[13, 28, 37, 116], a geo-textual object has an associated geographical location attribute and a textual attribute, and it may have additional attributes. Examples include geo-tagged tweets, POI data, check-in data, or any geo-referenced web content.

Searching for POIs on applications like Google Maps[1] is an important example of geo-textual search, using keywords and location restrictions[81]. Users often have personal preferences and specific requirements when looking for POIs that match their needs and geographic location. In certain situations, there is a need to find not just one, but a group of POIs that are closely located and collectively satisfy the needs of a specific person or enterprise. A specific arrangement of POIs (or other types of spatio-textual objects) based on a set of constraints about their relative positions is generally called a spatial pattern,

[1] https://www.google.com/maps

spatial configuration, or spatial scene[11, 50, 61]. In this work, the term spatial pattern is used, following Fang et al.[50] and Chen et al.[24].

There are numerous situations where people could benefit from a spatial pattern searching system. The following scenarios illustrate some examples:

- **Finding residential areas**: A user might be moving to a new city and want to find a house within 1km of a good primary school for their children. The person might also want a shopping mall nearby, for example, within 2km.
- **Trip planning**: A tourist visiting a foreign country may want to stay in a hotel close to major tourist attractions or specific types of attractions, such as a hotel near a museum and a beach.

Users typically find POIs through recommendation systems or keyword-based search systems[50, 52]. In keyword-based searches, two fundamental types of spatio-textual queries have been extensively studied: **range queries** and **k-Nearest Neighbors (kNN) queries**[28, 108]. Range queries retrieve spatio-textual objects within a specified distance from a central point while matching the query keywords (e.g., locating all restaurants within a 5-kilometer radius from a user's current location). In turn, kNN queries identify the top-k nearest objects to a given central point that satisfy specific keywords (e.g., finding the three closest gas stations to a particular address)[28, 108].

Numerous algorithms and indexes have been developed to address these classical spatial queries efficiently. However, more complex spatial queries, such as those requiring the retrieval of a group of POIs of different types that collectively meet a set of requirements, have received limited attention in the literature. Few researches have focused on optimizing the performance of spatio-textual queries with more elaborate parameters and complex search constraints[27, 44, 50, 65].

The Spatial Pattern Matching (SPM) query, formally proposed by Fang et al.[50] and investigated in various works[24, 51, 52, 81], aims to identify groups of POIs that conform to a user-defined spatial pattern established by keywords and distance requirements. For example, suppose a user seeks an apartment next to a primary school for their children and wants a hospital nearby due to a chronic disease. However, the person wants to avoid living too close to the hospital for hygiene reasons. An apartment between $200m$ and $1km$ away

from a hospital and at most $2km$ away from a school would be reasonable. Such requirement can be modeled as a spatial pattern graph using keywords and distance requirements. The spatial pattern for this search can be represented as a graph as outlined in Figure 1.1 (A).

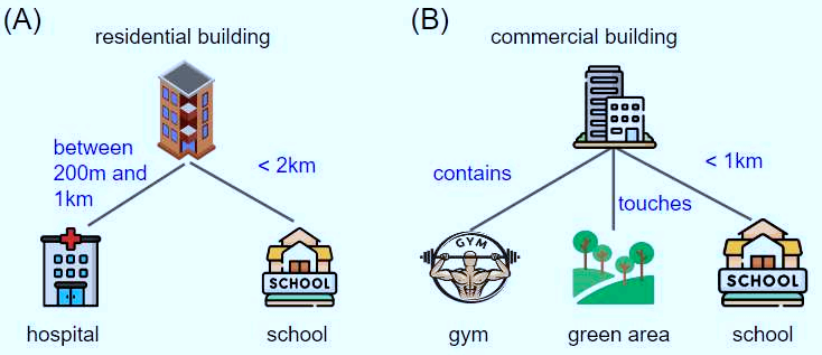

Figure 1.1: Example of a distance-based spatial pattern (A) and a qualitative and quantitative spatial pattern (B)

1.1 The Problem

While the traditional SPM query proposed by Fang. et al[50] is highly effective for scenarios with only distance constraints among queried POIs, it cannot address qualitative topological (connectivity) requirements between these entities. Consequently, traditional SPM algorithms are limited to searching spatial patterns with only distance constraints. In certain situations, users need more versatile solutions[90] to search for diverse spatial pattern configurations with both quantitative and qualitative requirements. For example, in a query such as "find a school adjacent to a wooded area".

To illustrate a more complex search scenario, consider an individual seeking a rental space within a commercial building for installing a small business. This user wants an onsite gym and a park with a green area touching the building, in way that the person does not need to cross any streets to reach the park. Preferably, the commercial building should be located within $1km$ of an elementary school for their child's convenience. This scenario can be modeled using a spatial pattern graph that incorporates both quantitative (distance) and qualitative (connectivity) constraints, as shown in Figure 1.1 (B).

Suppose the region for the scope of the described search is as depicted in Figure 1.2,

comprising three commercial buildings (CB1, CB2, CB3), a residential building (RB1), and two schools (S1, S2). The search for the spatial pattern illustrated in Figure 1.1 (B) would yield only two possible solutions (matches), formed by the POIs circled in red, namely (CB1, S1) and (CB1, S2). Note that the POI CB2 cannot constitute a search solution as it lacks a training gym, while CB3 also cannot be a solution candidate as it is not adjacent to any green area. Thus, (CB1, S1) and (CB1, S2) are the only two solutions for the user's search pattern in this region.

The situation outlined in the previous example cannot be efficiently addressed by using the traditional SPM algorithms (e.g., MSJ[50] and ESPM[24]), as it requires a more generic approach to represent spatial patterns with both quantitative and qualitative requirements, offering users a more diverse and flexible search parameter format[90]. There are several geospatial search platforms with appropriate tools for handling such searches, although they lack a specific or standard way of representing spatial pattern searches. Consequently, the task of formulating optimized SQL queries, or another query syntax, to address diverse spatial pattern search scenarios becomes a continuous difficulty and responsibility for technology specialists. Such difficulties could be overcome by the rise of specific guidelines for composing and representing spatial pattern searches in different geospatial technologies, as well as by the availability of software libraries designed specifically for efficiently addressing spatial pattern searches.

In addition to the absence of technologies designed to address more complex spatial pattern search scenarios, there is the challenge of computational cost and query execution

Figure 1.2: Finding matches for a spatial pattern quantitative and qualitative search

time. Numerous geospatial and spatio-textual indexing methods have been developed in the literature[38, 113, 120], along with studies focused on efficient solutions for basic spatial queries, such as range queries and top-k nearest neighbor queries. However, the literature review has revealed a limited number of works concerned with proposing solutions to more flexible and generic spatial pattern search settings, and directly addressing the performance issues of its solution approaches.

In this context, this master's thesis proposes and systematically investigates the Quantitative and Qualitative Spatial Pattern Matching (QQ-SPM) query as a versatile search approach for POIs and other types of spatio-textual objects. This search approach allows the representation of queries with multiple keywords, distance, and topological constraints. This work proposes guidelines on how to address such a complex type of spatio-textual search efficiently and details the design of efficient and effective algorithmic solutions.

1.2 Research Objective and Questions

In response to the outlined problem scenarios, the objective of this work is to propose a solution for the problem of retrieving groups of geo-textual objects (e.g., POIs) conforming to a set of keywords, distance and topological constraints, minimizing query execution time.

To address such an objective, this master's thesis envisions answering the following research questions (RQ):

- RQ_1: How to adapt the formalization of the SPM search problem from the literature to formalize the more generic QQ-SPM query, encompassing qualitative topological constraints?
- RQ_2: Can Inverted Linear Quadtree index be employed to efficiently solve QQ-SPM queries?
- RQ_3: Can Elasticsearch and PostGIS geospatial capabilities be employed to efficiently solve QQ-SPM queries?
- RQ_4: Does the number of search results, dataset size, query keywords and query spatial constraints impact the execution time of QQ-SPM queries?

These research questions are treated throughout the document. The question RQ_1 is treated in Section 4.1. Question RQ_2 is handled in Section 4.2. Question RQ_3 is investigated in Sections 4.3 and 4.4. Finally, Question RQ_4 is answered in Chapter 5.

1.3 Relevance

There are numerous scenarios in which society can benefit from spatial pattern search systems[24]. In the field of Human Settlement Analysis, geographers or urban planners may wish to analyze the occurrence of similar spatial patterns in different locations. They may employ a spatial pattern search system to identify areas where certain spatial patterns co-occur, or to pinpoint the presence of unique functions in specific regions, aiding in the classification of distinct types of human settlements.

Another significant application is Scene Recognition, which aims to identify locations based on incomplete information regarding attributes of various objects and their approximate relative positions, without explicit addresses[27]. For instance, individuals may seek to recall the name of a place they visited based on remembered POIs, such as a coffee shop, a nearby Chinese restaurant, and a visible car park located approximately 300 meters to the right of the restaurant. A spatial pattern search for this configuration could yield relevant places containing this pattern.

Spatial pattern search systems can also play a crucial role in decision-making processes aimed at preventing natural and urban disasters and addressing urban planning challenges[39, 128]. For example, government entities may utilize these systems to identify areas with specific spatial patterns that have historically posed risks for natural or urban disasters. This information can inform resource allocation and strategic decision-making to mitigate potential hazards. Additionally, certain urban spatial layouts may contribute to traffic accidents or facilitate criminal activities such as drug dealing[27]. When planning new districts, governments can verify existing areas with proposed layouts using spatial pattern search systems. If areas with similar layouts exhibit significant issues related to natural or urban disasters, governments may reconsider their plans and propose alternative layouts to enhance safety and security.

Other practical application scenarios benefiting from spatial pattern search systems can

arise in business place selection[12]. When a company considers opening a new branch, validating the appropriate location can involve checking existing regions with the proposed spatial configuration. This assessment helps determine if the spatial configurations align with the company's business objectives. Additionally, as discussed previously, spatial pattern search systems can assist in finding residential areas and in trip planning based on user preferences. These are common situations where such systems can provide valuable aid. In conclusion, there are numerous practical scenarios where decision-making processes can be directly optimized through the use of spatial pattern search systems.

1.4 Contributions

This master's thesis presents the following key contributions:

- A formal representation of QQ-SPM search, accompanied by mathematical proofs demonstrating effective pruning methods for finding QQ-SPM query solutions.
- The QQESPM-Quadtree algorithm, which offers a comprehensive solution for the investigated search scenarios.
- An open-source Python library implementing QQESPM-Quadtree search procedure which functions independently of spatial databases. It utilizes its own implementation of the IL-Quadtree indexing structure on disk, allowing the reading of data slices as needed in a scalable manner to prevent memory crashes.
- The open-source QQESPM-Elastic Python library which automatically converts QQ-SPM search requests from spatial pattern representation to Elasticsearch geo-queries. An algorithm controls the invocation of Elasticsearch geo-queries, which sequentially combine to find the solutions for the entire search pattern.
- The open-source QQESPM-SQL Python library which automatically converts QQ-SPM search requests represented in a spatial pattern format into a complete and efficient SQL query, by employing PostgreSQL.
- Extensive performance experimentation involving the execution of thousands of spatial queries using the proposed libraries.

- A QQ-SPM application prototype, including a backend API and a frontend layer for the Web environment, demonstrating the investigated query type in a practical POI search scenario (shown in Appendix A).

1.4.1 Bibliographic Contributions

This research has thus far produced the following bibliographic contributions.

- The research paper titled "QQESPM: A Quantitative and Qualitative Spatial Pattern Matching Algorithm" presented and published at Geoinfo 2023.
- The research paper titled "QQESPM: Spatial Keyword Search Based on Qualitative and Quantitative Spatial Patterns" passing through revisions for publication in the Journal of Information and Data Management (JIDM).

1.5 Document Structure

The document is structured as follows: Chapter 2 provides a concise review of key concepts pertinent to this research. In Chapter 3, a discussion on related works is presented. In Chapter 4, a formalization for the QQ-SPM search is proposed along with the introduction of an effective approach for solving QQ-SPM queries efficiently. This approach is subdivided into three solutions: QQESPM-Quadtree approach, QQESPM-Elastic and QQESPM-SQL. Chapter 5 outlines the setup for performance experiments, compares the proposed solutions, and discusses the results. Finally, Chapter 6 offers concluding remarks, summarizing key achievements and suggesting future research directions.

Chapter 2

Background

To efficiently address common spatial queries, such as range queries and kNN queries, researchers have proposed various spatial and spatio-textual indexing structures. This chapter briefly reviews some of these indexing techniques. Additionally, it discusses concepts related to qualitative spatial reasoning and topological relations. This discussion will provide background for the design of efficient QQ-SPM solutions in Chapter 4, which demonstrates how to effectively use existing spatio-textual indexes to solve the QQ-SPM query, simultaneously involving distance and topological constraints.

2.1 Spatial Indexing

A spatial index contains summarized information designed to support efficient spatial data access methods. These methods enable the selection of objects that satisfy specific spatial requirements, such as proximity to a given point[7]. Spatial indexes typically rely on hierarchical tree structures. These structures expedite the removal of irrelevant data during query execution by partitioning either the space or the data into clusters based on spatial proximity. Essentially, spatial indexes organize spatial data according to their spatial attributes, facilitating efficient data retrieval for common spatial queries.

2.1.1 Data-driven vs Space-driven Decomposition

In the existing literature, researchers have explored two primary approaches for hierarchically decomposing data based on spatial attributes to create tree-based spatial indexing structures [3, 4, 48, 93]. These approaches play a crucial role in optimizing spatial data retrieval. The first approach is data-driven decomposition, which centers around data-driven trees, commonly referred to as balanced trees. In this method, the space decomposition rule relies entirely on the spatial distribution of the input data. An illustrative example of data-driven decomposition is the R-tree index, proposed by Guttman[68]. In the R-tree, the bounding rectangles of the nodes are determined solely based on the spatial coordinates of the data objects. The R-tree effectively groups nearby objects into nodes, creating a hierarchical structure that facilitates efficient spatial queries.

In contrast, the second approach is known as space-driven decomposition. This method of spatial data indexing gives rise to what are commonly referred to as space-partitioning trees. In this case, the rule for dividing data relies solely on spatial considerations. Quadtree indexes, initially proposed by Finkel et al.[56], serve as prime examples of space-driven decomposition. They consistently divide the space hierarchically by bisecting it into smaller rectangles, which are nodes in the Quadtree. Chapter 4 demonstrates how to efficiently use a quadtree-based spatio-textual indexing to enhance the performance of the investigated QQ-SPM queries.

2.1.2 Quadtrees

A two-dimensional Quadtree[56] divides a bidimensional Euclidean space into four quadrants. During index construction, a universal bounding rectangle is assumed for all spatial data points. This rectangle serves as the root node of the quadtree. When the number of data points within this node exceeds a certain limit, it is subdivided into four child nodes using horizontal and vertical lines passing through the center of the space. These child nodes correspond to the southwest, southeast, northwest, and northeast quadrants, encoded as 00, 01, 10, and 11, respectively. Figure 2.1 (A) illustrates the space subdivision process in a quadtree, while Figure 2.1 (B) depicts the resulting hierarchical tree structure. Each rectangular subdivision of the space corresponds to a node in the tree. If any of these four

nodes at the first depth level contains more data points than the specified limit, it is further divided into four new nodes, each with the same size of bounding rectangle. This hierarchical and recursive node subdivision continues until all leaf nodes contain fewer objects than the threshold.

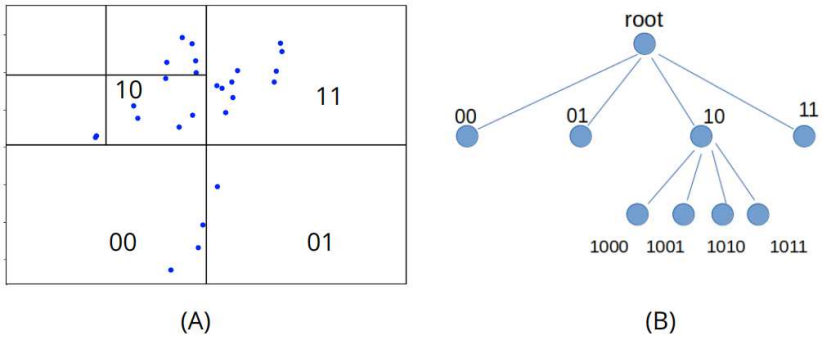

Figure 2.1: Example of a quadtree space subdivision (A) and its corresponding tree structure (B)

The delimitation of bounding rectangles for nodes in a two-dimensional quadtree is not determined by enclosing data objects. Instead, it follows a space-driven rule: the nodes bounding rectangle is always divided through its center, resulting in four equally-sized children nodes. Each of these nodes represents a specific bounding box within the space. Consequently, quadtrees fall into the category of space-partitioning trees.

To uniquely identify each node, a strategy following its path up to the root level is used. Although there is no universal standard, a common convention assigns an empty identifier to the root node. For other nodes, the identifier is formed by concatenating the parent node's identifier with a binary string:

- If a node is the southwest child of its parent, the identifier receives the suffix "00"
- If it is the southeast child, the suffix is "01"
- For the northwest child, the suffix is "10"
- And for the northeast child, the suffix is "11"

These binary numbers (00, 01, 10, 11) encode the hierarchical relationships. As an example, the node that is the northeast child of the northwest child of the root receives the identifier "1011".

Distinctly, the bounding rectangles for nodes in an R-tree are directly defined by the spatial data points. Specifically, the node's bounding rectangle should be the smallest rectangle that encloses the set of its inner objects. For this reason, R-trees are known for utilizing a data-driven space decomposition. Both R-tree indexes and Quadtrees share similarities and can be used interchangeably to index the spatial attributes of a dataset of objects.

The choice between R-trees and Quadtrees for spatial queries depends on several factors. Here are some observations: R-trees tend to outperform Quadtrees in nearest-neighbor queries. However, when dealing with range queries involving a radius greater than 10 miles (approximately 16.09 kilometers), Quadtrees may exhibit better performance, as demonstrated by Kothuri et al.[77]. Considering the query environment, frequent data updates may also favor Quadtrees. The PostgreSQL documentation recommends creating a spatial GiST index (R-Tree) for general purposes, but to consider the addition of spatial SP-GiST indexes (Quadtree and other space-partitioning indexes), when the dataset exceeds a few thousand rows. Such an approach can yield performance benefits for queries involving long datasets. In this work, a series of implementation decisions for the design of efficient QQ-SPM solutions is based in such recommendations, employing both Quadtrees and R-trees, as will be discussed in Chapter 4.

2.2 Spatio-Textual Indexing

Spatio-textual indexing structures organize spatio-textual data based on both textual and spatial attributes simultaneously. Spatio-textual objects consist of data entities simultaneously containing textual and spatial attributes[13, 28, 37, 116]. In this sense, for queries involving keywords in addition to spatial requirements, spatio-textual indexes are generally most suitable. Their combined indexing strategy, considering both text and location, optimizes query efficiency for both types of requirements. Two common spatio-textual index structures are the Inverted File R-Tree (IR-Tree), proposed by Wu et al.[113] and Cong et al.[38], and the Inverted Linear Quadtree (IL-Quadtree) index, proposed by Zhang et al.[120].

As highlighted in previous research[41, 111, 120, 129], the performance of spatial

keyword queries tends to suffer when objects are independently organized by separate textual and spatial indexes. To address this limitation, spatio-textual indexes have been developed. These hybrid indexing structures aim to enhance the efficiency of queries that involve both types of data attributes. In Chapter 4, it will be demonstrated how to effectively employ a spatio-textual indexing to solve QQ-SPM queries efficiently, specifically, the IL-Quadtree index.

2.2.1 IL-Quadtree

The Inverted Linear Quadtree (IL-Quadtree) index[120] was proposed as a spatio-textual indexing solution in comparison with other existing indexes, specifically designed to optimize the processing of top-k spatial keyword search problem. This structure is particularly well-suited for scenarios where data objects have both a spatial attribute (location) and a set of keywords. It is most suitable for datasets containing a relatively small number of distinct keywords. The IL-Quadtree approach involves creating and maintaining a separate and independent Quadtree index structure for each distinct keyword in the dataset. Thus, the Quadtree of each keyword indexes all the objects associated with that particular keyword. If an object has multiple keywords, it will be present in more than one Quadtree, specifically, one for each of its associated keywords.

The IL-Quadtree index, as outlined in its research article by Zhang et al.[120], is specifically designed to enhance the performance of the top-k spatial keyword search problem, which is similarly investigated by De Felipe et al.[41] and Zhou et al.[129]. In this scenario, a set of spatio-textual objects, a query location q, and a set of keywords are involved. The goal is to retrieve the k closest objects to the query location q, with each object containing all the specified keywords. Beyond this, other research has demonstrated the effective utilization of the IL-Quadtree indexing for more generic search problems. For instance, Chen et al.[24] proposed the employment of the IL-Quadtree indexing to efficiently addresses a spatial pattern matching (SPM) search that considers a set of keywords and pair-wise distance restrictions between the searched objects. This master's thesis proposes an efficient way of employing IL-Quadtrees for solving QQ-SPM queries. The detailed steps will be described in Sections 4.1 through 4.2.2.

2.3 Qualitative Spatial Reasoning

Understanding and reasoning about spatial aspects of the world is crucial. However, mapping human spatial reasoning into computational representation remains challenging due to its context-dependent interpretation, as investigated in[1, 36, 53, 58, 61, 62, 71, 97, 104] . Existing studies[8, 57, 60, 71, 103] have proposed geometrical approaches to formally interpret the meaning of natural-language spatial predicates. Although the ambiguity intrinsic in such definitions is inevitably persistent.

Qualitative spatial reasoning (QSR) involves developing calculi that allow machines to represent and reason with spatial entities while abstracting away metrical details[36, 62, 85, 91]. The term "Qualitative Spatial Calculi" (QSC) refers to formal theories aimed at emulating human spatial representation and reasoning abilities[59]. As outlined by Moratz and Ragni[91], in QSR, two primary investigative scenarios emerge: topological relation reasoning (involving concepts like "touches", "contains" and "traverses") and orientation reasoning (which considers cardinal directions and other orientation-related aspects).

In this work, the primary focus lies in the topological reasoning aspect. For instance, a user might seek a training gym WITHIN a shopping mall (topological reasoning), while the significance of searching for a training gym located to the LEFT (orientation reasoning) of a specific place may be less pronounced. Although modeling topological spatial reasoning from natural language to computational representation, such as in[5, 61, 104], is essential, this research does not delve into this direction.

The proposed approach for solving QQ-SPM queries assumes that each qualitative spatial predicate expression has an established formal definition without ambiguities. While prepositions like "near", "between" and "connected" are challenging to formally define, they can still be assigned distance thresholds and geometrical conventions as implementation decisions for computational representations, as investigated by Aflaki et al.[1] and Fogliaroni et al.[59]. However, in this current research, the use of such complex predicates was avoided. Instead, this research is restricted to formally defined named spatial predicates such as "contains", "intersects" and "within". These predicates can be precisely represented within a formal topological model, enabling their computational utilization in spatial pattern search systems. For instance, users can select from a fixed set of topological relations as qualitative

connectivity requirements between two searched entities.

2.3.1 Topological Relations

Topological relations, also known as connectivity relations, exhibit special properties: they are rotation-invariant, translation-invariant, and scaling-invariant. These relations involve qualitative spatial predicates that specifically represent information related to the topological aspects of two spatial regions (or geometries, shapes)[6, 32]. In this context, the scope of this master's thesis lies in binary topological relations within a 2-dimensional Euclidean space. Figure 2.2 illustrates examples of the core binary topological relations between two spatial regions.

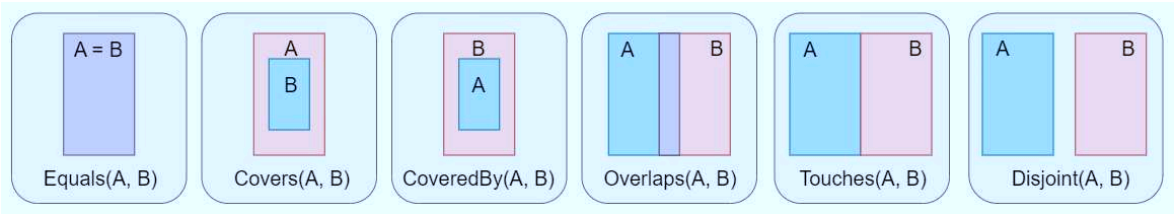

Figure 2.2: Example of Topological Relations

The solutions proposed in this work assume that the qualitative query constraints have a formal and unambiguous computational representation. This requires the use of a formal topological model to represent the query constraints computationally. The next subsections review the two most widely-adopted formal topological models. These models can be used interchangeably for designing and implementing a QQ-SPM solution approach.

Region Connection Calculus

Several models have been proposed in the literature to formally define and represent topological relationships, aiming to provide a non-ambiguous and computational representation for them[18]. One important such model is the Region Connection Calculus (RCC) proposed by Randell et al.[98] and Cohn et al.[35], which serves as a formal method for defining the possible topological relations between two spatial objects. RCC relies on axiomatic first-order logic to define the topological relations. The domain of discourse for this logic is the universe U of all existing spatial regions with a same given dimension. The

foundational binary topological relation, denoted by the Connected predicate (referred to as relation C), forms the basis upon which all other topological predicates are defined. RCC operates based on two fundamental axioms related to the C predicate. The first establishes that every spatial region is connected to itself. The second establishes that if a spatial region X is connected to another spatial region Y, the reciprocal also holds, i.e., Y is connected to X. Table 2.1 provides the boolean expressions that define the most common topological relations within RCC.

Topological Predicate	**Meaning**	**Boolean Expression Definition**
$DC(x, y)$	x is DISCONNECT from y	$\neg C(x, y)$
$P(x, y)$	x is a PART of y	$\forall z[C(z, x) \rightarrow C(z, y)]$
$PP(x, y)$	x is a PROPER PART of y	$P(x, y) \wedge \neg P(y, x)$
$EQ(x, y)$	x is EQUAL to y	$P(x, y) \wedge P(y, x)$
$O(x, y)$	x OVERLAPS y	$\exists z[P(z, x) \wedge P(z, y)]$
$PO(x, y)$	x PARTIALLY OVERLAPS y	$O(x, y) \wedge \neg P(x, y) \wedge \neg P(y, x)$
$DR(x, y)$	x is DISCRETE from y	$\neg O(x, y)$
$EC(x, y)$	x is EXTERNALLY CONNECTED to y	$C(x, y) \wedge \neg O(x, y)$
$TPP(x, y)$	x is a TANGENTIAL PROPER PART of y	$PP(x, y) \wedge \exists z[EC(z, x) \wedge EC(z, y)]$
$NTPP(x, y)$	x is a NON-TANGENTIAL PROPER PART of y	$PP(x, y) \wedge \neg\exists z[EC(z, x) \wedge EC(z, y)]$
$P^{-1}(x, y)$	x has y as a PART	$P(y, x)$
$PP^{-1}(x, y)$	x has y as a PROPER PART	$PP(y, x)$
$TPP^{-1}(x, y)$	x has y as a TANGENTIAL PROPER PART	$TPP(y, x)$
$NTPP^{-1}(x, y)$	x has y as a NON-TANGENTIAL PROPER PART	$NTPP(y, x)$

Table 2.1: Formal definitions of the most common topological predicates in RCC

In Table 2.1, it can be observed that the relation "Overlaps" differs from the foundational relation "Connected". The latter only requires a point in common, whereas the former necessitates the existence of a spatial region (with the same dimensionality as the two regions) that is a common part of both regions. Additionally, the relation "Part" ($P(x, y)$) is sometimes referred to as "Within", or simply "In". Similarly, the relation "Part Inverse" ($P^{-1}(x, y)$) is also known as "Contains". Furthermore, the relation "Externally Connected" is also known as "Touches" or "Meets". The *RCC8* standard[100] considers only eight relations: $EQ, DC, EC, TPP, NTPP, PO, TPP^{-1}$ and $NTPP^{-1}$. Notably, the logical inclusive disjunction of the specialized relations TPP and $NTPP$ is logically equivalent to the more general relation PP. Similarly, the logical conjunction of the relations

$TPP, NTPP$ and EQ is equivalent to the more general relation P.

Dimensionally Extended 9-Intersection Model

Another equally important topological formalization model is the Dimensionally-Extended 9 Intersections Model (DE-9IM), proposed in previous research[33, 34, 46, 47]. This model allows for more granular definitions of possible topological relations, including relations specific to regions with different dimensions (e.g., the relation "Cross"). In DE-9IM, each possible spatial region divides the entire space into three parts: its interior, its boundary, and its exterior. Unlike using first-order boolean logic expressions to define binary topological relations, DE-9IM employs a 3x3 matrix that captures the dimensions of intersections between the interior, boundary, and exterior of two spatial regions.

Each possible matrix represents a specific topological relation, and the computation of these intersections for a pair of spatial regions A and B inherently defines the topological relation between them. Figure 2.3 [1] exemplifies the computation of the DE-9IM matrix for two partially overlapping spatial regions A and B. The resulting matrix yields the topological relation defined by the numbers $2, 1, 2, 1, 0, 1, 2, 1, 2$. Notably, this represents a highly refined and specific topological relation. Only more general topological relations receive named spatial predicates. For example, the relation "Covers", is not determined by a single matrix but rather by a set of several matrices collectively fulfilling its configuration.

Each named spatial predicates is assigned a DE-9IM matrix pattern governing its definition. There is a standard[94, 95] for representing specific matrix patterns. Within this standard, matrix entries can be a number (representing the dimension size of an existing intersection), or a mask representing a more generic case, as follows:

- F is used to denote the nonexistence of an intersection.
- T represents the existence of an intersection, regardless of its dimension.
- An asterisk $(*)$ signifies either the existence or nonexistence of the intersection, or simply indicates missing information about the intersection.

Consider Figure 2.3, where the DE-9IM matrix for two geometries is shown. The matrix entries are $2, 1, 2, 1, 0, 1, 2, 1, 2$. These entries satisfy the matrix pattern $T * T * * * T * *$

[1] Image credit: By Krauss, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=20825354

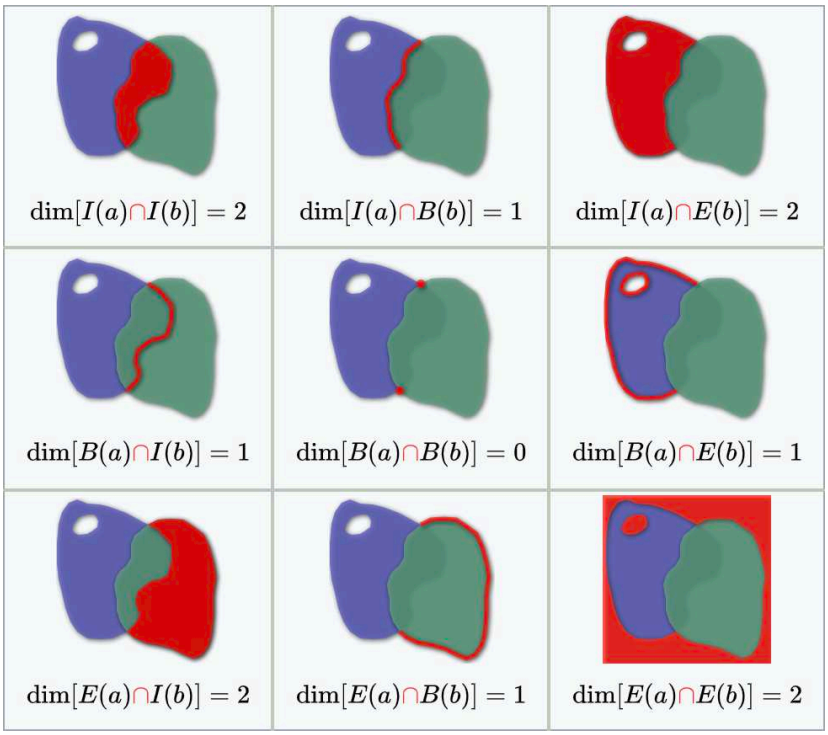

Figure 2.3: Example of the 9-intersections matrix for two overlapping geometries A and B

but do not meet the pattern $F0TTTT**T$. The pattern $T*T****T**$ actually defines the topological relation "Overlaps". Thus, the two geometries depicted in Figure 2.3 are overlapping. Matrix patterns allows the representation of various spatial predicates. For instance, the named spatial predicate "Disjoint" consists of all DE-9IM matrices that fulfill the pattern $FF*FF****$. Table 2.3 shows the matrix patterns defining the most common binary topological predicates in DE-9IM. The relation "Intersects" can be understood as the logical inclusive disjunction of the following DE-9IM relations: "Equals", "Covers", "CoveredBy", "Overlaps", and "Touches" (as in Figure 2.2)

In the context of the DE-9IM model, matrix patterns for opposite spatial relations exhibit an interesting symmetry: they are the transpose matrix of each other. For instance, this behavior occurs with the matrices patterns of the relations "Contains" and "Within", since Contains(A, B) is equivalent to Within(B, A). Also, In DE-9IM there are subtle distinctions between certain relations. Specifically, the relation "Contains" differs slightly from "Covers". When we say "A contains B", it implies that every point of B lies within A and the interiors of these geometries have a non-empty intersection. In contrast, "Covers" is less restrictive, only requiring that every point of B is a point of A. Similarly, the relation

Topological Predicate	**Meaning**	**DE-9IM matrix patterns**	**Equivalent To**
Equals(A, B)	the geometries A and B are topologically equal	T*F**FFF*	Equals(B, A)
Disjoint(A, B)	the geometries A and B have no point in common	FF*FF****	Not Intersects
Intersects(A, B)	the geometries A and B have at least one point in common	T******** or *T******* or ***T***** or ****T****	Not Disjoint
Touches/Meets(A, B)	the geometries A and B intersect but their interiors do not	FT******* or F**T***** or F***T****	Touches(B, A)
Crosses(A, B)	geometry A crosses/traverses geometry B	T*T****** (if dim(A) < dim(B)) or 0******** (if 1=dim(A)=dim(B))	
CoveredBy(A, B)	Every point of A is a point of B	T*F**F*** or *TF**F*** or **FT*F*** or **F*TF***	Covers(B, A)
Within(A, B)	A is covered by B and the interiors intersect	T*F**F***	Contains(B, A)
Overlaps(A, B)	the intersection of the interiors of A and B has the same dimension as the geometries, and none is covered by the other	T*T***T** (if dim = 0 or 2) or 1*T***T** (if dim = 1)	
Covers(A, B)	Every point of B is a point of A	T*****FF* or *T****FF* or ***T**FF* or ****T*FF*	CoveredBy(B, A)
Contains(A, B)	A covers B and the interiors intersect	T*****FF*	Within(B, A)

Table 2.2: Pattern Matrices for the most common topological predicates in DE-9IM

"A within B" is more stringent than "CoveredBy(A, B)". For "A within B", the interiors of both geometries must intersect. As a distinguishing example, if geometry A is equal to the boundary of geometry B, CoveredBy(A, B) evaluates to True, while Within(A, B) is False.

In the RCC (Region Connection Calculus) model, the PART/WITHIN relation doesn't explicitly require interior intersections. However, this condition occurs implicitly because both geometries must share the same dimension. DE-9IM's predicates "Within" and "CoveredBy" correspond to the RCC predicate "Part". Similarly, DE-9IM's predicates "Contains" and "Covers" align with the RCC predicate "Part Inverse". Although their definitions are more refined in DE-9IM, making these relations subtly distinct. Table 2.3 shows the equivalences between predicates in RCC and DE-9IM.

The topological relations "Equals" and "Touches" were not employed in this research, due to their infrequent occurrence in the analyzed datasets. Additionally, these relations are special cases of "Intersects" which is readily available for representation in QQ-SPM queries. However, the relation "Covers" is still retained, even though it is a special case of "Intersects". The reason for this choice is the intuition that "Intersects" may frequently output neighboring geometries, in cases where a user may rather specifically want a direct containment relation (e.g., a restaurant within a shopping mall).

In summary, this research encompasses a set of four core topological relations is used. In RCC, they are the relations Connected, Disconnected, Part, and Part Inverse. These correspond to the DE-9IM relations "Intersects", "Disjoint", "CoveredBy", and "Covers". This proposed search pattern is therefore restricted to these four qualitative spatial predicates. Such a choice is only intended to simplify the formalization of QQ-SPM queries presented here, and should not be considered a restrictive flexibility for future studies on QQ-SPM spatial queries.

2.4 Final Considerations

This chapter reviewed spatial and spatio-textual indexing, emphasizing that these structures are designed for efficiently answering specific spatial queries, particularly range and kNN queries. The chapter also explored the concepts of Qualitative Spatial Reasoning and Qualitative Spatial Relations, including a discussion of two formal methods for representing

RCC relation	**DE-9IM relation**
Connected	Intersects
Disconnected	Disjoint
Part/Within	CoveredBy/Within
Part Inverse/Contains	Covers/Contains
Externally Connected	Touches

Table 2.3: Equivalences between some RCC and DE-9IM relations

topological reasoning. These concepts are applied in Chapter 4 to describe the design of efficient solutions for QQ-SPM queries, which represent a more complex and flexible spatio-textual search format. The next chapter discusses various related works, outlining their strengths and weaknesses concerning different spatio-textual search requirements.

Chapter 3

Related Work

Spatial Keyword Queries, also known as Spatio-Textual Queries, typically involve a central point or region location and a set of keywords as input. The goal is to identify a set of geo-textual data objects from a dataset that are most closely related spatially and textually to the query input. According to the taxonomy of spatio-textual queries proposed by Chen et al.[26, 28], this research falls into the category of geo-textual queries over static data in Euclidean space, specifically related to group queries.

Carniel[22] proposes a taxonomy for spatial queries with quantitative and qualitative constraints, focusing on purely spatial queries (i.e., without keywords). The study highlights that the most significant spatial relations in the existing literature on spatial queries are metric relations (e.g., distance), directional relations (e.g., north, east, left, right), and topological relations (e.g., contains, intersects). The latter two are qualitative constraints, while the former is a quantitative constraint.

An extended SQL spatial language capable of representing a diverse set of spatio-textual queries is proposed by Mahmood et al.[87]. However, the research only outlines the guidelines for such a language, without providing implementation and performance analysis. Long et al[85] and Kothuri et al.[76] propose alternative methods to enable faster computation of distance and topological relations between two spatial geometries. Although these methods accelerate the verification of spatial predicates between geometrical shapes, they do not provide efficient ways to prune irrelevant regions during spatial queries. Therefore, designing efficient algorithms and indexing techniques for rapid space pruning and early discarding of numerous irrelevant spatial objects remains a critical research

objective.

In the following sections, a set of existing studies on spatio-textual queries is discussed, focusing on works that explore performance and efficient solutions for specific types of spatio-textual queries. Typically, these studies formalize a specific spatio-textual query, demonstrate its importance through application scenarios, and illustrate efficient methods to address such a query. These researches often propose appropriate indexing, design specialized algorithms, or compare various approaches for solving specific types of spatial queries efficiently.

3.1 Item-wise Queries

Item-wise queries, also known as standard queries, are designed to produce a set of results where each result consists of a single spatio-textual object, with each result being independent of the others. Numerous studies focus on enhancing the performance of item-wise queries. Below, a few significant categories of item-wise queries are highlighted:

- **Range/Window queries**[25, 31, 54, 56, 68, 69, 78]: aim to find all objects within a circular or rectangular region centered on a given location. For geo-textual queries, each result must satisfy the boolean condition of the keywords specified in the query. These queries are also known as Boolean-Boolean queries.

- **Point/Region queries**[17, 54]: aim to find all objects whose spatial attributes intersect with the input point or region of the query. For geo-textual queries, each result must satisfy the boolean condition of the keywords specified in the query.

- **Boolean k Nearest Neighbors queries**[19, 25, 54, 86, 109, 118, 120]: aim to find the k closest objects to a given central point, satisfying the query's input keywords. These queries are also known as Boolean-Ranking queries.

- **Top-k Spatial Keyword queries**[25, 38, 42, 74, 75, 82, 88, 96, 101, 106, 107, 116, 123]: aim to find the k objects most relevant to the query location and keywords, and potentially other query attributes, based on a specific cost function combining spatial and textual relevance to the query parameters. These queries are also known as Full-Ranking queries.

Despite the importance of these standard spatio-textual queries, the item-wise search configuration cannot account for scenarios where a group of spatially close geo-textual objects collectively satisfy specific user needs. Such situations highlight the need for the design of group queries, specialized in finding optimal groups of objects based on specific geo-textual search parameters, serving distinct user purposes.

Existing research explores the parallel processing of multiple standard queries. Fevgas and Bozanis[54] investigate the parallelization of point and region queries, while other studies[102, 115, 118] delve into the parallel batch processing of range, kNN or top-k queries. While this configuration enables efficient item-wise query processing, it does not fully address the challenge of identifying groups of diverse objects that collectively meet specific search criteria and are spatially close.

3.2 Group Queries

Spatial joins involve pair-wise spatial queries, typically focusing solely on spatial aspects without incorporating textual keywords. They entail identifying pairs of objects from two sets that meet a specified spatial relation, which can be quantitative (e.g., distance limit) or qualitative (e.g., topological relation). Various studies explore the performance of different spatial join algorithms and techniques[10, 63, 73]. Although spatial joins can be combined to search for more than two objects, this concatenation approach is not always the most efficient for locating groups of spatial objects. Different search scenarios and configurations require particular attention, leading to the development of specialized algorithms tailored to specific search tasks. Consequently, the literature contains several works dedicated to queries aiming to find groups of objects of arbitrary sizes, known as group queries.

In contrast to item-wise queries, group queries focus on identifying groups of closely associated spatio-textual objects that collectively meet the keyword preferences specified in the query. Consequently, each result of a group query comprises a cluster of objects characterized by proximity in spatial locations and high relevance to the query keywords.

Wallgrün et al.[112] propose a type of purely spatial group query, devoid of keywords, which involves solving qualitative queries depicted in sketchmaps by interpreting the directions between objects. The spatial pattern in a map comprises a sketchmap that

illustrates the relative positions of spatial objects and the directions (e.g., north, east) between pairs of objects. Their investigated search task is termed Qualitative Spatial Constraint Networks (QSCN). The term "network" refers to the graph-based structure of the input sketchmap, where nodes represent spatial objects and edges connecting nodes denote the annotated directional orientations between the locations of pairs of spatial objects in the pattern.

While the QSCN task enables the matching of spatial information among different map sources, it does not address spatial pattern scenarios involving keywords. Thus, it cannot accommodate search requirements such as locating a grocery store within 1 kilometer of a specified house and within 10 kilometers of a beach. QSCN does not cater to these scenarios as it deals with spatial objects that lack geo-textual attributes.

The following sections introduce other types of group queries, encompassing spatio-textual query settings. These queries filter results based not only on spatial constraints but also on keywords provided as search parameters.

3.2.1 Collective Spatial Keyword Queries

The Collective Spatial Keyword queries (CoSKQ) involve receiving a query location and a set of keywords, aiming to identify optimal groups of objects that collectively meet the query keywords and are closely situated. Studies such as[65, 121, 122] employ a cost function geared towards minimizing the diameter of the resulting group of objects. The objective of this CoSKQ-specific search is termed the m-Closest Spatial Keyword Cover (mCK) search task.

Various works[14, 16, 84] employ a cost function designed to minimize both the inter-object distance and the distance to the query central location simultaneously. In addition, studies such as[29, 30] utilize a cost function aiming to minimize the total number of objects in the resulting group, thereby simplifying the set of objects that fulfill the search requirements. Moreover, other research[20, 21, 43, 70, 105, 125, 126] explores more comprehensive cost functions for optimizing results in a CoSKQ search, incorporating a diverse array of factors to compose a unique metric that considers spatial and textual relevance.

While CoSKQs effectively incorporate query keywords and retrieve spatially proximate

objects, they do not exploit pairwise spatial constraints between two objects, be they quantitative or qualitative. For instance, CoSKQs do not accommodate binary spatial relations such as locating a gym *within* a shopping mall. CoSQKq primarily rely on a unified metric determined by a cost function for the entire retrieved group of objects. Moreover, such queries do not address proximity avoidance constraints. For instance, if a user prefers to find an apartment that is not in close proximity to a nightclub, traditional CoSKQ queries would not ensure such requirements and would not factor in such exclusion preferences in the cost function for result optimization.

3.2.2 Neighborhood-Preference Queries

The term "Neighborhood-Preference Query" (NPQ) is employed in this study to denote spatio-textual queries focused on identifying optimal data objects based on their proximity to other objects relevant to user preferences. In essence, an NPQ entails a primary keyword indicating the content or type of the primary searched object (referred to as the data object), along with a set of keywords representing user preferences for objects in proximity to the searched data object (referred to as feature objects). Several existing works[40, 44, 64, 80, 110, 119] address such specific search scenarios. The NPQ task involves identifying the top-k optimal data objects, which are those having the most favorable neighborhood according to the user's specified preferences, delineated by the keywords for the feature objects. The score of a data object takes into account its proximity to the user-expected feature objects and the quality of these surrounding feature objects, based on specific quality metrics.

For instance, if a tourist is seeking a hotel close to a renowned Chinese restaurant, the optimal hotels would be those nearest to highly-rated Chinese eateries. In such scenarios, a scoring metric could be defined by a cost function that considers the proximity of the searched data object to the nearest feature objects and their associated quality.

However, NPQs yield results limited to the central searched data object and do not provide the precise locations of the feature objects, which are also relevant to the user. Each result presents an optimal data object for the search, ensuring the presence of relevant feature objects in the surrounding proximity neighborhood of the central object. However, users must independently locate the relevant neighborhood themselves. Additionally, NPQs

do not consider proximity avoidance constraints (exclusion constraints), similar to CoSKQ queries, as mentioned.

3.2.3 Spatial Pattern Matching

The Spatial Pattern Matching (SPM) query was initially proposed by Fang et al.[50] and further investigated in other studies[24, 51, 52, 81]. SPM takes a spatial pattern as input, represented as a graph where vertices contain keywords for the searched objects, and each edge linking two vertices specifies a quantitative spatial condition with lower and upper bound distances between the paired objects corresponding to the vertices. Additionally, the SPM query introduces exclusion constraints, which involve proximity avoidance conditions. These constraints allow for the formulation of search scenarios where users can specify the preference for the non-existence of specific types of facilities in proximity to the query results. This enhances the richness and flexibility of search scenarios, catering to specific user requirements within such a search framework.

Guo et al.[67] suggests incorporating social factors to refine the results of SPM searches. For instance, by leveraging a social network that links users and POIs, the outcomes of an SPM search could be filtered based on their relevance to a user's circle of friends. However, the author does not offer a methodology for constructing such queries, as it necessitates quantifying complex social factors, and the proposed processing algorithms assume the availability of these metrics in the query parameters.

Chen et al.[27] delves into a variant of the SPM search known as Example-based Spatial Pattern Matching (EPM). The EPM search task entails identifying all occurrences of a given spatial pattern within a geo-textual dataset, where the spatial pattern is defined by a set of points, each linked to a keyword and spatial coordinates. Unlike traditional SPM, which relies on pair-wise distance constraints between objects, EPM focuses on specific example locations to configure the search pattern. A group of objects is considered a match if they can be derived from the search pattern through translations and rotations, with allowances for relative positioning tolerance. For instance, in a POI search with EPM, if a POI with keyword A is located between two others with keywords B and C in the search pattern, a match must replicate this relative spatial arrangement. On the other hand, traditional SPM queries define a match based solely on satisfying distance constraints between corresponding

pairs, irrespective of relative positions and orientations.

While the traditional SPM query and its mentioned variations offer flexibility and robustness in various search scenarios, they exclusively address quantitative distance constraints, thereby overlooking specific qualitative preferences expressed by users. For instance, a user might wish to locate a gymnasium situated *within* the premises of a shopping mall or a commercial building directly *adjacent* to a public park with a verdant expanse, ensuring seamless access without crossing streets. However, such nuanced qualitative and topological preferences remain unaddressed by SPM queries, requiring manual sifting through query results to identify those meeting these criteria.

3.3 Queries with Quantitative and Qualitative Constraints

Several studies address search scenarios encompassing both quantitative and qualitative criteria. For instance, Guo et al.[66] and Li et al.[79] introduce and explore the concept of Direction-Aware Nearest Neighbor Queries, which seek direction-diverse spatio-textual objects. For instance, consider a scenario where a person is driving and seeks nearby coffee shops. Ideally, the most suitable options would be establishments situated directly along the highway the person is traveling on.

Guo et al.[66] introduced a method to retrieve recommendations distributed across all possible directions from the central query location, aiming to encompass results from all directions in cases where the optimal direction is uncertain. On the other hand, Li et al.[79] offers algorithmic approaches to address the retrieval of the top-k objects confined within a defined angular direction range from the initial search location. While these studies incorporate directional spatial reasoning, they do not accommodate topological requirements.

The point and region queries examined by Fevgas et al.[54] and Carniel et al.[17] consider qualitative topological constraints, focusing on identifying all spatial objects holding specific topological relations with a designated fixed point or region of interest. However, these queries are purely spatial and do not incorporate keyword search capabilities. Additionally, they are not tailored to retrieve pairs of spatial objects with varying geometries, as one geometry is fixed (either the point or region of interest). Moreover, these queries yield

individual data objects as results, lacking the capability to retrieve groups of objects.

Spatial joins, based on topological relations between the geometries of two spatial datasets[10, 63, 73], represent another form of qualitative and topological spatial query. However, while many studies focus on the performance of spatial joins, they typically do not incorporate keyword search conditions.

The existing spatio-textual join literature predominantly focuses on spatio-textual similarity joins[9, 28, 49, 72, 83, 99, 127]. This type of join involves pairs of spatio-textual objects that are both spatially proximate and textually similar. Such an approach finds utility in various applications, such as social network friend recommendations, where close spatial references and overlapping textual content between two profiles may indicate a strong friendship. Additionally, this type of join facilitates the integration and matching of corresponding spatio-textual data from multiple sources, as the joined pairs exhibit high relevance in terms of keywords and spatial attributes.

However, this join operation is not intended for query scenarios like "finding a coffee shop within 1 kilometer of a theater and adjacent to a wooded area". Such example requires spatial proximity conditions with unrelated keywords, making it unsuitable for spatio-textual similarity joins.

As discussed in Section 3.2, the study by Wallgrün et al.[112] enables qualitative requirements, specifically cardinal directions, in the spatial pattern search using sketchmaps. However, it focuses solely on spatial aspects and does not address searches based on keywords.

Deng et al.[44] explored a Neighborhood-Preference query termed Clue-based Spatio-textual Query, which integrates quantitative and qualitative aspects. It aims to identify locations conforming to a specific spatial arrangement centered around a focal object with preferred surrounding objects. These surrounding objects must adhere to predefined directional relations relative to the central object outlined in the search pattern. However, this query does not accommodate qualitative topological constraints.

Rafael[97] introduced the concept of Qualitative Spatial Pattern Matching (QSPM), which adapts the definition of the SPM query to compose spatial search patterns with keywords and topological requirements between the searched objects. However, this study focuses solely on qualitative queries and does not explore or address search patterns

with both distance and topological pair-wise constraints between the searched objects. Additionally, exclusion constraints, such as scenarios where the user aims to avoid close proximity with unwanted facilities, cannot be represented in the QSPM query. These limitations highlight the need for solutions tailored to addressing more versatile and comprehensive spatial search patterns.

3.4 Comparison of the Types of Spatio-Textual Queries

Table 3.1 provides a summarized comparison of related works, highlighting the features present and absent in each type of spatio-textual query studied in the literature. The examined features include:

- Whether the queries are spatio-textual or purely spatial.
- Whether the queries represent quantitative constraints (e.g., distance range limits).
- Whether the queries represent qualitative constraints (e.g., directional or topological restrictions).
- Whether the queries specifically allow topological constraints (e.g., contains, intersects).
- Whether the queries retrieve individual items or groups of items.
- Whether the queries allow the representation of pair-wise constraints (e.g., specifying a spatial restriction between only two specific objects in the search).
- Whether the queries can represent exclusion constraints (e.g., “retrieve only POIs not located next to cemeteries”).

The SPM query type is closely related to the proposed QQ-SPM search task, although it cannot address qualitative topological requirements. Therefore, SPM algorithms do not optimize query processing for qualitative scenarios (e.g., “find a shopping mall containing a gym”). Similarly, the QSPM query covers five of the compared features but fails to represent distance and exclusion constraints. Notably, among the identified studies, this

research uniquely and comprehensively encompasses all seven search features outlined as the columns in Table 3.1.

Notably, simply answering multi-constrained spatio-textual queries is not a primary research focus, as it is feasible to develop applications that process various spatio-textual user requirements and generate straightforward queries against spatial databases based on specific search filters. Nonetheless, the research discussed in the realm of geo-textual queries focuses on efficiently and effectively answering specific spatio-textual queries. This involves addressing diverse search scenarios either by introducing novel approaches or utilizing existing indexing techniques and algorithms optimized for each specific query type. To our knowledge, this research represents the first systematic effort to define, propose, and investigate efficient solutions for QQ-SPM queries.

In summary, the proposed QQ-SPM query is designed to encompass both quantitative and qualitative requirements, including topological constraints. It aims to retrieve groups of closely located spatial objects that collectively meet the user's criteria. This type of query also allows for pair-wise restrictions between the searched objects and facilitates the use of exclusion constraints, which can be exemplified by scenarios where users need to find POIs that should not be near an unwanted type of facility. This generic, flexible, and comprehensive group query configuration is referred to as the QQ-SPM query. Such a query configuration is the central subject of investigation in this master's thesis. This research evaluates the performance of various search strategies tailored for efficiently addressing such a query.

3.5 Final Considerations

This chapter provided a discussion on the features present in various studies aimed at investigating different geo-textual queries. Each query is tailored to a specific search scenario, often necessitating specific indexing and algorithms for optimal processing. The discussion highlighted the effectiveness of the QQ-SPM query in accommodating both quantitative (distance) and qualitative (topological) user requirements, as well as exclusion constraints. Furthermore, this master's thesis has been compared and differentiated from existing research by identifying seven key features central to the QQ-SPM query task that

Query type	**Works**	**Spatio-Textual**	**Quantitative**	**Qualitative**	**Topological constraints**	**Group search**	**Pair-wise constraints**	**Exclusion constraints**
Range/Window, kNN and top-k queries	[2, 15, 19, 23, 25, 31, 38, 42, 54, 56, 64, 68, 69, 74, 75, 78, 82, 86, 88, 89, 92, 96, 101, 106, 107, 109, 114, 116, 117, 119, 120, 123, 124]	X	X					
Range/Window, kNN and top-k queries + Direction	[66, 79]	X	X	X				
Batch of Range/Window, kNN or top-k queries	[102, 115, 118]	X	X					
Point/Region queries	[17, 54]			X	X			
Batch of Point/Region queries	[54]			X	X			
Spatial Join	[10, 63, 73]		X	X	X		X	
QSCN	[112]			X		X	X	
CoSKQ	[14, 16, 20, 21, 29, 30, 43, 65, 70, 84, 105, 121, 122, 125, 126]	X	X			X		
NPQ	[80]	X	X			X		
NPQ with distance limits	[40, 64, 110, 119]	X	X			X	X	
NPQ with distance limits + Direction	[44]	X	X	X		X	X	
EPM	[27]	X	X			X	X	
SPM	[24, 50, 51, 52, 67, 81]	X	X			X	X	X
QSPM	[97]	X		X	X	X	X	
QQ-SPM	**This research**	**X**	**X**	**X**	**X**	**X**	**X**	**X**

Table 3.1: Related work

are not fully addressed by other studies. The next chapter provides a formal definition for the QQ-SPM query and delve into the design of solutions to address the proposed search problem efficiently.

Chapter 4

QQESPM

This chapter introduces QQESPM (Quantitative and Qualitative Efficient Spatial Pattern Matching), a solution approach for the QQ-SPM search problem. This approach is composed of three solutions. The chapter begins by formally defining the essential terms pertinent to QQ-SPM queries in Section 4.1. Two crucial mathematical theorems are established, to aid in designing an efficient algorithmic solution for the QQ-SPM query. Subsequently, Sections 4.2, 4.3 and 4.4 present in detail the three proposed solutions: QQESPM-Quadtree, QQESPM-Elastic and QQESPM-SQL. Several implementation decisions, that aimed at improving the performance of such solutions, are discussed in this chapter.

The first solution, QQESPM-Quadtree operates independently of geospatial storage technologies, by performing searches within its own implementation of the IL-Quadtree index structure stored on disk. The second solution, QQESPM-Elastic, leverages Elasticsearch's geospatial features. It converts spatial requirements from the query into a set of requests using Elasticsearch's native geospatial functions. The third, QQESPM-SQL, automatically translates search pattern requirements into an efficient SQL query, utilizing the PostGIS geospatial capabilities. The proposed solutions enhance the efficiency and accuracy of QQ-SPM queries across different geospatial technologies.

4.1 Problem Formalization

4.1.1 Problem Definition

In this section, the key concepts related to the QQ-SPM search problem are formally defined. This formalization is built upon the SPM search formalization proposed by Chen et al.[24], by incrementing minimum aspects related to the topological constraints. The design of such a formalization answers Research Question RQ_1.

It is assumed that a spatio-textual or geo-textual object is a multimodal data entity denoted by $o = (o.loc, o.doc)$. Here, $o.loc$ represents the spatial attribute of o, which can take the form of a point with coordinates or a more intricate shape format like a polygon. Additionally, $o.doc$ comprises a set of keywords associated with o.

Definition 1 (spatial pattern)**.** *A Spatial Pattern is a graph $G(V, E)$ with a set of n vertices $V = v_1, ..., v_n$ and a set E of m edges $e_{ij} = e(v_i, v_j)$, satisfying the following constraints:*

1. *each vertex $v_i \in V$ has an associated keyword w_i*
2. *each edge $e_{ij} \in E$ stores the spatial constraints for a pair of objects matching the keywords of its endpoint vertices v_i and v_j. The possible constraints consist of:*
   - *a topological spatial predicate $\Re_{ij}$, among the following: $\{contains, within, intersects, disjoint\}$*
   - *a distance lower bound l_{ij}, and a distance upper bound u_{ij}*
   - *an exclusion sign $\tau \in \{\leftarrow, \rightarrow, \leftrightarrow, -\}$*

Each possible spatial pattern graph represents a parameter for a QQ-SPM query. In a POI group search, the vertices specify the POIs' desired keywords, the connectivity predicates indicate the desired connectivity relationships between the POIs. The spatial pattern graph, as outlined in Definition 1, requires that for each vertex v_i, an object o_i from a dataset of geo-textual objects must be identified. The keyword associated with vertex v_i must match one of the keywords in $o_i.loc$. Furthermore, the distances and topological relationships between the objects must adhere to the spatial constraints specified by the edges of the search pattern. The

distances between the POIs are restricted by the lower (l_{ij}) and upper (u_{ij}) bounds associated with the edge. This query representation can be generalized to any other geo-textual objects search scenarios. The meanings of the possible exclusion signs for an edge are described below:

- $v_i \rightarrow v_j$ [v_i excludes v_j]: There must not exist any geo-textual object with keyword w_j within a distance less than l_{ij} from a matching object corresponding to v_i.

- $v_i \leftarrow v_j$ [v_j excludes v_i]: There must not exist any geo-textual object with keyword w_i in the dataset within a distance less than l_{ij} from a matching object corresponding to v_j.

- $v_i \leftrightarrow v_j$ [mutual exclusion]: The two-way restriction, i.e., v_i excludes v_j and v_j excludes v_i.

- $v_i \ v_j$ [mutual inclusion]: The presence of geo-textual objects with keywords w_i and w_j in the dataset with distance shorter than l_{ij} from the objects corresponding to v_i, v_j is allowed for the search results.

The key distinction between the SPM's spatial pattern representation and the QQ-SPM's representation lies in the latter's ability to represent qualitative topological constraints, as specified in condition 2(a) of Definition 1. The QQ-SPM query examined in this master's thesis is also more generic than the QSPM query analyzed by Rafael[97]. Unlike QSPM searches, it allows for the representation of exclusion constraints (through exclusion signs on the edges in the search pattern) and the specification of customized distance ranges between pairs of searched objects.

Edges with distance interval restrictions for the searched objects are called quantitative edges, and those with topological restrictions are called qualitative edges. Edges can be both quantitative and qualitative simultaneously. A quantitative edge that has one of the exclusion signs "$\leftarrow$", "$\rightarrow$" or "$\leftrightarrow$" is called an **exclusive** edge, and the search constraint specified by such an edge is called an exclusion constraint. Edges with the mutual inclusion sign ("$-$") are termed **inclusive** edges. For example, if an edge of the search pattern connects vertices with the keywords "apartment" and "school", and has $l_{ij} = 100m$, $u_{ij} = 1000m$, and a mutual inclusion sign of $\tau = \text{“} - \text{”}$, then an apartment A_1 located $105m$ far from a school

S_1 will appear among the search results, regardless of whether there are other schools closer than $100m$ to A_1. Conversely, if the edge has the sign $\tau =$ " $\rightarrow$ " between "apartment" and "school", then for the pair of objects (A_1, S_1) to meet the edge requirements, there must be an additional condition that no other school S exists closer than $l_{ij} = 100m$ from A_1. In one such school exists, the pair (A_1, S_1) will not figure among the search results, since the search pattern specifies that the apartment must be located sufficiently far from schools (exclusion constraint).

Notably, the relation "*within*" is the opposite of "*contains*". Although this redundancy could be discarded, the proposed model retain both relations because edges are directional in the search patterh graph, having specific starting and ending vertices, allowing multiple ways to generate the search pattern. The exclusion sign of an edge is also called a proximity avoidance constraint.

Definition 2 (matching pair of objects)**.** *A pair of geo-textual objects (o_i, o_j) is called a matching pair of objects for the edge $e(v_i, v_j)$ if o_i, o_j have the keywords of the vertices v_i, v_j respectively, and their spatial locations satisfy the constraints of the edge e.*

Definition 3 (match)**.** *A tuple of n geo-textual objects $S = (o_1, o_2, ..., o_n)$ is called a match for a spatial pattern $G(V, E)$ if $|V| = n$, and for each i, o_i has the keyword of the vertex v_i, and for each edge $e(v_i, v_j)$ of G, the pair of objecs (o_i, o_j) is a matching pair of objects for the edge e_{ij}.*

Note that the order of geo-textual objects in the tuple corresponds to the order of vertices in the pattern G, so the i^{th} object o_i in the matching tuple corresponds to the i^{th} vertex (v_i) in the pattern G. Notably, the QQ-SPM query format generalizes the SPM query by incorporating both SPM search criteria and topological requirements for specifying spatial pattern searches.

Problem 1 (QQ-SPM search problem)**.** *The QQ-SPM search problem or QQ-SPM query consists of finding all the matches of a spatial pattern $G(V, E)$ in a dataset D of geo-textual objects, i.e., finding all combinations of objects from D that match the requirements represented in the given spatial pattern graph.*

4.1.2 Pruning Space with Mathematical Theorems

This section presents two theorems that aid in designing an efficient pruning method for an algorithm solving a QQ-SPM query. The next section uses these theorems to design such an algorithm. To apply these theorems, the dataset of geo-textual objects must be indexed in a IL-Quadtree indexing structure. This section and the following aims at proposing an efficient employment of the IL-Quadtree index in the resolution of QQ-SPM queries, answering Research Question RQ_2. To efficiently find matching pairs of objects, a search procedure may prune unpromising regions using the concept of promising pairs of nodes, formally defined below.

Definition 4 (promising pair of nodes)**.** *Let ILQ be an IL-Quadtree of geo-textual objects, $G(V, E)$ a spatial pattern graph, $e(v_i, v_j)$ an edge of G, let ILQ_i and ILQ_j be the quadtrees for the keywords of vertices v_i and v_j, respectively, n_i, n_j be two nodes from ILQ_i and ILQ_j, respectively, and b_i, b_j their bounding boxes. We say that the node pair (n_i, n_j) is a promising pair of nodes for the edge $e(v_i, v_j)$ if $d_{min}(b_i, b_j) \leq u_{ij}$ and $d_{max}(b_i, b_j) \geq l_{ij}$, and additionally, the following conditions hold:*

1. *In case $v_i \rightarrow v_j$: there is no object x_j in ILQ_j such that $d_{max}(b_i, x_j) < l_{ij}$*
2. *In case $v_i \leftarrow v_j$: there is no object x_i in ILQ_i such that $d_{max}(x_i, b_j) < l_{ij}$*
3. *In case $v_i \leftrightarrow v_j$: (1) and (2) hold*
4. *In case e is qualitative with $\Re_{ij} \neq$ "$disjoint$": $b_i \cap b_j \neq \emptyset$*

The functions d_{min} and d_{max} represent the minimum and maximum distance between the bounding boxes of two nodes. The following intuition underlies the definition of a promising pair of nodes. Intuitively, a pair of nodes n_i, n_j is considered promising for the edge e if the bounding boxes of its nodes potentially contain matching pairs of objects for the edge e. Specifically, if the minimum and maximum distances between the nodes' bounding boxes b_i, b_j do not rule out the possibility of geo-textual objects o_i, o_j inside these nodes forming a matching pair for the edge e, then the children nodes or inner objects of n_i, n_j must be further examined. These are candidates for finding matching pairs of objects for the edge e in context.

Theorem 1. *Let $e(v_i, v_j)$ be an edge from a spatial pattern graph $G(V, E)$, and let ILQ_i, ILQ_j be the quadtrees for the keywords of the vertices v_i, v_j, respectively. Consider a pair o_i, o_j of geo-textual objects in the dataset D. If (o_i, o_j) is a matching pair of objects for the edge e, then for any nodes n_i, n_j from the quadtrees ILQ_i, ILQ_j, respectively, if o_i is inside n_i and o_j is inside n_j, then the node pair (n_i, n_j) is a promising pair of nodes of e.*

Proof. Since o_i, o_j is a matching pair of objects, then this pair satisfies the constraints of the edge e, in particular, we have $d_{max}(b_i, b_j) \geq d(o_i, o_j) \geq l_{ij}$, and $d_{min}(b_i, b_j) \leq d(o_i, o_j) \leq u_{ij}$. Also, if e is qualitative with $\Re_{ij} \neq$ "$disjoint$", then o_i, o_j are intersecting, and n_i, n_j will be intersecting too. Now let us analyse the possible signs of the edge e. In case $v_i \rightarrow v_j$, suppose there is an object x_j ILQ_j such that $d_{max}(b_i, x_j) < l_{ij}$, but since o_i is inside the bounding box b_i of the node n_i, then $l_{ij} > d_{max}(b_i, x_j) \geq d(o_i, x_j)$ which is contradictory to (o_i, o_j) being a matching pair of objects. So such x_j does not exist. The proof for the other possible signs are analogous, and thus, all the conditions for n_i, n_j to be a promising pair of nodes are satisfied. □

Using Theorem 1 in a QQ-SPM search procedure, only the promising pairs of nodes require interior verification, in order to identify all the matching objects. A pair of objects o_i, o_j outside all promising pairs of nodes for an edge e will never be a matching pair of objects for e. In other words, no solutions for an edge e exist outside its promising pairs of nodes. Thus, by focusing only on these promising pairs, an algorithm can find all matches of a spatial pattern without analyzing the entire dataset. The following Theorem provides the final piece for designing an efficient search for promising pairs of nodes.

Theorem 2. *If (n_i, n_j) is not a promising pair of nodes for the edge $e(v_i, v_j)$, then, any pair (n_i^c, n_j^c), where n_i^c is children of n_i and n_j^c is children of n_j, cannot be a promising pair of nodes for the edge $e(v_i, v_j)$.*

Proof. The proof for the quantitative restrictions of the edges is provided in a study by Chen et al.[24]. Regarding the additional proposed constraint to qualify as a promising pair of nodes, which is related to the connectivity requirement of the edge, we verify through the contrapositive version of the statement. It is important to note that if the node pair (n_i, n_j) constitutes a promising pair of nodes for an edge with a qualitative constraint other than "*disjoint*", the nodes will possess intersecting bounding boxes. Since their parent nodes cover

them, they will be intersecting as well, thus ensuring that if a pair of nodes is promising for an edge e, so is the pair of their parent nodes. □

By employing Theorem 2 in a QQ-SPM search procedure, the promising pairs of nodes for the next level can be determined without the need to compare the bounding boxes of every node at the current depth level of the quadtrees. Instead, a search procedure only need to examine the children node pairs of the promising pairs of nodes from the previous level. This is because for a node pair to be promising at the current level, its parent node pair must be promising at the preceding level. By leveraging Theorems 1 and 2, a search procedure can efficiently identify promising node pairs for edges level by level. Finally, such procedure can verify the inner objects of the promising node pairs at the leaf level to identify the matching objects. This process is detailed in Section 4.2.

4.1.3 Example of Search Pattern

The graph structure in Figure 4.1 (A) specifies a spatial pattern (Def. 1) which composes a query to find groups of commercial building, gym and school, such that the commercial building contains the gym inside, and is located between 2.5 and 5.5 distance units from the school. The edge "commercial building - school" is exclusive with sign "$\rightarrow$", indicating that the output commercial buildings must not be located closer than 2.5 distance units from any other school. This requires finding a commercial building sufficiently far from schools (exclusion constraint). Let's denote the vertices of the search pattern (commercial building, gym and school) as V_1, V_2, V_3 respectively, and the edges "commercial building - gym", "commercial building - school" as e_{CG} and e_{CS} respectively.

The area os POIs displayed in Figure 4.1 (B) denotes the target dataset of POIs for the query within a simple 2-D euclidean space. The POIs $CB1, CB2, CB3$ have the keyword "commercial building", the POIs $G1, G2, G3$ contain the keyword "gym", and $S1, S2$ contain the keyword "school". In this case, the pair of POIs $(CB1, S1)$ constitutes a matching pair of objects (Def. 2) for the edge e_{CS} ("commercial building - school") since this pair satisfy the distance and exclusion constraints of this edge. The object pair $(CB2, S1)$ does not constitute a matching pair of objects for the edge e_{CS} since the commercial building $CB2$ is too close to the school $S2$, thereby violating the exclusion constraint of the edge

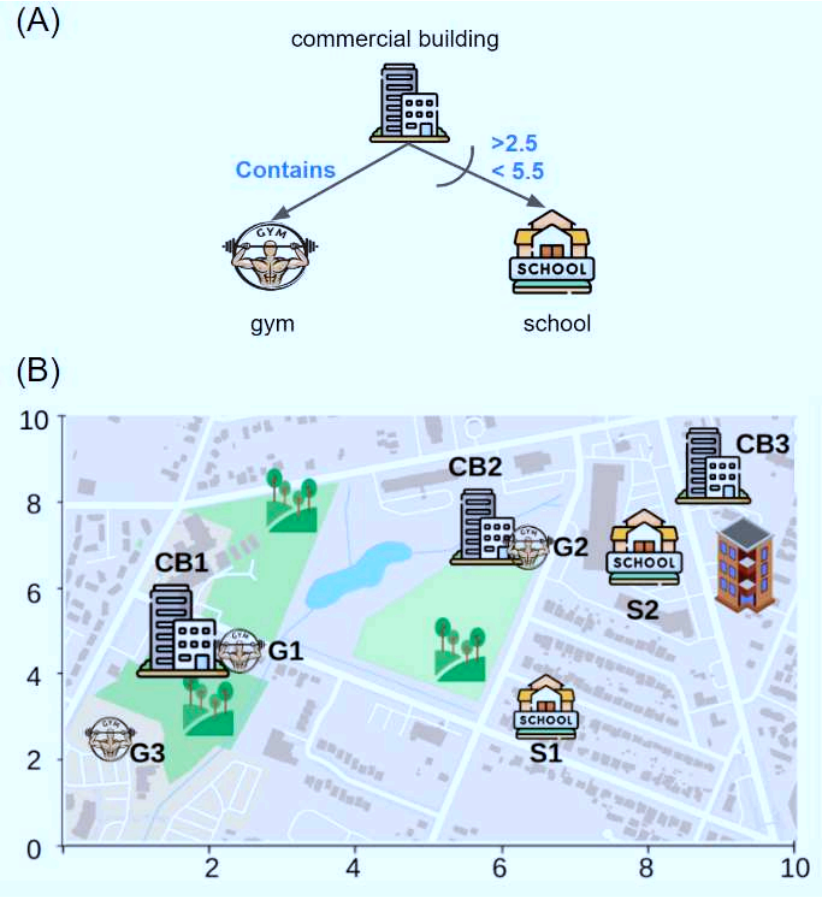

Figure 4.1: Map Visualization of Fictitious POIs dataset for Search Example

e_{CS}. In turn, the object pair $(CB1, G1)$ is a matching pair of objects for the edge e_{CG}. Furthermore, the tuple of objects $(CB1, G1, S1)$ constitutes a match (Def. 3) for the spatial pattern of the query (Figure 4.1 (A)), since it contains one object for each vertex of the search pattern, and satisfy the keywords, distance and topological constraints of the search pattern.

Figure 4.2 displays the constructed quadtrees for the geo-textual objects with the keywords "commercial building" (A), "gym" (B) and "school" (C). In this example, the node pair $(N_{C:30}, N_{G:3})$ is identified as a promising pair of nodes for the edge e_{CG} (commercial building - gym), since these nodes' bounding boxes are intersecting, allowing them to potentially contain inner objects that satisfy the spatial constraints of the edge e_{CG}. Likewise, the node pair $(N_{C:0}, N_{S:3})$ is a promising pair of nodes for the edge e_{CS}, since the minimum distance between these nodes' bounding boxes is 0 which is less than the edge's constraint $u_{ij} = 5.5$, and the maximum distances between these nodes' bounding boxes is 14.142

which is greater than $l_{ij} = 2.5$, thereby meeting the criteria for a promising pair of nodes from Def. 4.

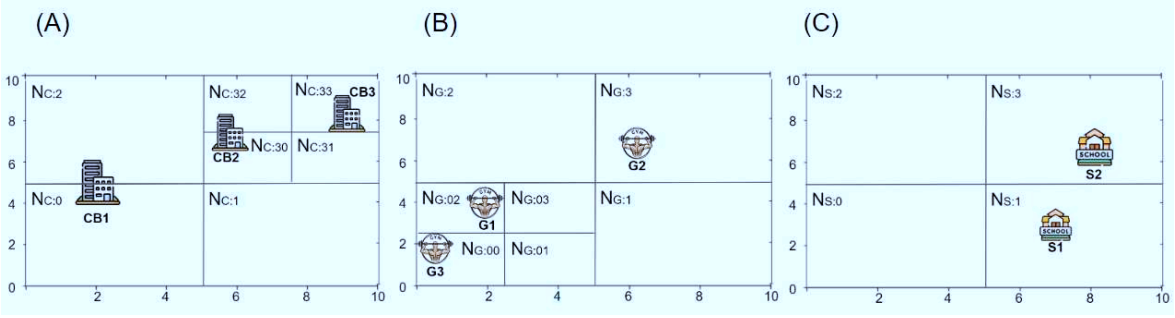

Figure 4.2: Quadtrees for the keywords in the fictitious POIs dataset

4.2 QQESPM-Quadtree

This section introduces the QQESPM-Quadtree solution. First, the design of its efficient QQ-SPM search algorithm is presented. To aid in understanding the algorithm procedure, a search example is provided. Further details of the QQESPM-Quadtree solution are subsequently explained.

4.2.1 The Algorithm

This section presents the QQESPM-Quadtree algorithm, designed specifically to address QQ-SPM queries, which involve four distinct topological relations between geo-textual objects: "*contains*", "*within*", "*intersects*" and "*disjoint*". Algorithm 1 (QQESPM-Quadtree) outlines the search procedure, detailing the high-level sequential steps for query execution, with an emphasis on achieving efficiency through the concept of promising pairs of nodes. QQESPM-Quadtree takes as input a spatial pattern represented as a graph $G(V, E)$ and a dataset of geo-textual objects indexed in an IL-Quadtree ILQ, comprising quadtrees ILQ_i corresponding to each keyword w_i in the dataset. It then finds all matches of the spatial graph G within the dataset of geo-textual objects.

The maximum depth level of the quadtree at which QQESPM-Quadtree operates corresponds to the deepest quadtree associated with the keywords in the search pattern (Line 1 of Algorithm 1). For each edge, the initial promising pair of nodes consists of root nodes of the quadtrees for the keywords of the edge's endpoint vertices (Lines 2 through 3 of Algorithm 1). The search procedure of QQESPM-Quadtree then computes promising node pairs at each depth level of the quadtrees by leveraging the promising pairs from previous levels (Lines 4 through 6 of Algorithm 1). The subroutine `Compute_Promising_Node_Pairs_Level` invoked in Line 6 of Algorithm 1 is detailed in Algorithm 2. This procedure receives the current depth level l and the promising node pairs from the previous level for each edge e which is kept in the variable $\Psi_{e(l-1)}$. It then computes the promising node pairs for each edge at the next depth level of the quadtrees.

QQESPM-Quadtree prioritizes processing edges with sign "$\leftarrow$", "$\rightarrow$" or "$\leftrightarrow$" (exclusive edges) over those with sign "$-$" (inclusive edges) by reordering the list of edges (Lines 2 through 6 of Algorithm 2). This order optimizes the algorithm by initially addressing the most restrictive spatial requirements, thereby reducing the number of candidate solutions accumulated during execution.

During the computation of promising node pairs for edges at a specific depth level, QQESPM-Quadtree maintains temporary sets of candidate nodes for each vertex in the search pattern (Lines 7 through 8 of Algorithm 2). At each iteration for a given level l, the algorithm identifies promising node pairs for the next level of each edge by analyzing the children of the promising node pairs from the same edge in the previous level, as outlined in Theorem 2. However, these promising node pairs can be restricted to those whose nodes are among the candidate nodes associated with adjacent edges (Lines 11 through 12 of Algorithm 2). For example, considering edges e_{12} and e_{13}, which share vertex v_1, the node for keyword w_1 in the promising node pairs of e_{13} must be among the nodes for w_1 in the promising pairs of e_{12}. This ensures that candidate solutions for subsequent edges are constrained by those processed earlier, underscoring the importance of prioritizing more restrictive edges. After identifying the promising node pairs for an edge, the candidate nodes for their endpoint vertices are updated (Lines 17 through 20 of Algorithm 2). Following this logic, after computing promising node pairs for each edge at a specific quadtree level, QQESPM-Quadtree reorders the edge list for the next level, giving priority to edges with

fewer promising node pairs in the previous level.

Definition 5 (skip-edge)**.** *Give an ordered sequence of edges* $\Pi = (e_1, e_2, ..., e_m)$ *from a spatial pattern graph* G*. An edge* e_k *from* Π *is said to be a skip-edge in the sequence* Π *if* e_k *is an inclusive edge and their endpoint vertices* v_i, v_j *lie in edges of the sub-sequence* $(e_1, e_2, ..., e_{k-1})$ *of the edges preceding* e_k*.*

After computing the promising node pairs for all edges at the leaf level, the edges are reordered into a new sequence $e_1, e_2, ..., e_m$ using a connected greedy strategy (Line 7 of Algorithm 1). The search procedure guiding this reordering is delegated to the subroutine `Generate_Connected_Edges_Ordering`, which is detailed in Algorithm 3. The ordering is considered connected because the first edge in the sequence, e_1, must be adjacent to the second edge, e_2, which in turn must be adjacent to the third, and so forth. The strategy is described as greedy because the first edge, e_1, is selected based on the minimum number of promising node pairs (Line 2 of Algorithm 3). The next edge is then chosen from among the adjacent edges of the previously added edge (Line 10 of Algorithm 3), prioritizing the edge with either the minimum or maximum number of promising node pairs (Lines 11 through 19 of Algorithm 3). At this stage, since the subroutine `Generate_Connected_Edges_Ordering` was invoked with the parameter alt set to false, the greedy strategy does not alternate between minimum and maximum values. Consequently, the next edge to be chosen is always the adjacent edge with the minimum number of promising node pairs. The greedy alternating strategy for edge ordering (with the parameter alt set to true) is employed during the joining phase in a subsequent step of the algorithm.

If no adjacent edges are available at a given point, the next edge in the sequence is chosen from those that are connected to any of the previously added edges (Lines 20 through 28 of Algorithm 3). The variable B (Line 4 of Algorithm 3) maintains a set of vertices from already added edges that still have remaining adjacent edges. This set is used to select the next edge when no adjacent edges are available to the most recently added edge.

At this stage, the algorithm identifies a subset of edges known as skip edges, where computing matching pairs of objects is unnecessary (Line 8 of Algorithm 1). This exception applies to inclusive edges that share identical endpoints with other edges for which matching pairs of objects have already been computed. Definition 5 formally defines the concept of a

skip edge. For these edges, the promising pairs of objects can be inferred by filtering through candidate solutions from adjacent edges and retaining only those that satisfy the skip edge's constraints. For non-skip edges, QQESPM-Quadtree calculates candidate pairs of objects by evaluating the proximity and connectivity relationships of all objects within the promising pairs of nodes at the leaf level (Lines 9 through 10 of Algorithm 1). For topological relations other than "disjoint", it is advantageous to test only bounding box intersections, which is computationally less intensive than exact connectivity checks. Since many candidate pairs of objects will be discarded during the joining phase, this approach avoids unnecessary connectivity computations for objects that will ultimately be excluded, thereby optimizing efficiency.

During the computation of promising pairs of nodes and objects for the edges, the algorithm often needs to verify whether a pair of nodes or objects A and B constitutes a promising pair for an exclusive edge. To check the exclusion constraint, it performs a range query centered on A to determine if an object too close to A shares the same keyword as B. In such cases, QQESPM-Quadtree utilizes previous computations by storing the results of these range queries in temporary memory, thus avoiding redundant tests throughout a QQ-SPM query. Specifically, this pruning based on exclusion constraints may occur during the nodes' filtering phase (Line 14 of Algorithm 2) or the objects' filtering phase (Line 10 of Algorithm 1).

For instance, consider an edge e with endpoints labeled "apartment" and "school", where $l_{ij} = 100$m, $u_{ij} = 1000$m, and the exclusion sign $\tau = \text{“} \rightarrow \text{”}$. To determine if a pair of objects (A_1, S_1) qualifies as a matching pair for the edge e, the exclusion constraint must be checked. Specifically, it needs to be verified whether a school S exists within a distance less than l_{ij} from A_1. This requires performing a radius search centered on A_1 with a radius $r = l_{ij} = 100$m to find objects with the keyword "school" in this vicinity. If such an object is located within this radius, the pair (A_1, S_1) is disqualified as a matching pair for the edge e. Additionally, through computation reuse, A_1 is excluded from consideration with any other schools. The results of the radius search are stored in memory throughout the query, thereby avoiding the need to test A_1 against other schools.

After computing candidate pairs of objects for each non-skip edge, QQESPM-Quadtree reorders the edges for the joining phase (Line 11 of Algorithm 1). This reordering utilizes the

same function, `Generate_Connected_Edges_Ordering` (Algorithm 3), previously invoked in Line 7. At this stage, the alt parameter for this subroutine is set to $true$. This indicates that the greedy ordering of the edges will alternate between those with the minimum and those with the maximum number of promising node pairs. This alternating greedy strategy helps to avoid costly nested loop operations in the joining of the matching object pairs of all edges.

After reordering, QQESPM-Quadtree proceeds to the joining routine, which merges promising pairs of objects for adjacent edges and eliminates mismatches (Lines 12 through 14 of Algorithm 1). Notably, edges sharing a common vertex must match the same geo-textual object for that vertex to form a valid join. Once all non-skip edges are joined, the algorithm evaluates the resulting partial solutions, discarding those that do not satisfy the constraints of the skip edges (Lines 15 through 16 of Algorithm 1). Finally, it verifies the actual topological relationships among objects within the final candidate solutions to determine the ultimate solutions (Lines 17 to 18 of Algorithm 1). This final verification is essential because, in Line 10, all topological constraints except "disjoint" were only superficially assessed through bounding box intersection checks.

The QQESPM-Quadtree framework builds upon the structural design of ESPM but introduces distinct criteria for promising node pairs and matching pairs of objects, incorporating qualitative topological requirements. At each level, QQESPM-Quadtree performs targeted computations, progressively identifying promising pairs of nodes from the root to the leaf level of the keyword's quadtrees. By integrating early filtering based on connectivity constraints through bounding boxes intersection checks, QQESPM-Quadtree effectively eliminates unpromising regions or objects, thereby reducing the cost of the joining phase. Additionally, the reordering of edges helps avoid costly computations in later stages of the search procedure. Ultimately, reusing computations for radius searches concerning exclusion constraints accelerates query performance by preventing redundant evaluations of nodes or objects unlikely to meet the exclusive edge requirements.

Algorithm 1: QQESPM-QUADTREE

Input: IL-Quadtree ILQ, spatial pattern $G(V, E)$

Output: ψ: all the matches of G

$L = max(depth(ILQ_i), 1 \leq i \leq n)$

for *edge* $e(v_i, v_j) \in E$ **do**

  $\Psi_{e(0)} \leftarrow \{ILQ_i$.root, ILQ_j.root$\}$

for *each level* $l = 1$ *to* L **do**

  for *each edge* e **do**

    $\Psi_{e(l)} \leftarrow$ RUN Compute_Promising_Node_Pairs_Level(l, $\Psi_{e(l-1)}, G(V, E)$)

$E \leftarrow$RUN Generate_Connected_Edges_Ordering(Ψ, $G(V, E)$, false) // reorder the edges using a greedy minimizer strategy

$SE \leftarrow$the skip-edges in sequence E according to Def. 5

for *each non-skip edge* $e \in E \backslash SE$ **do**

  $\Phi e \leftarrow$ the promising object pairs for e within the promising node pairs $\Psi_{e(l)}$ // filter based on distances or bounding box intersection

$E \leftarrow$ RUN Generate_Connected_Edges_Ordering(Φ, $G(V, E)$, true)// reorder the edges using a greedy alternating strategy

$\psi \leftarrow \emptyset$ // keep the partially constructed matches of the search pattern

for *each non-skip edge* $e \in E$ **do**

  $\psi \leftarrow \psi$.join(Φe)

for *each skip-edge* $e \in SE$ **do**

  filter out partial solutions in ψ not satisfying constraints of e

for *each edge* $e \in E$ *with topological constraint* **do**

  filter out partial solutions in ψ not satisfying the exact topological constraint of e

return ψ

Algorithm 2: COMPUTE_PROMISING_NODE_PAIRS_LEVEL

Input: l: current quadtree depth level, Ψ: promising node pairs for edges from previous depth level, $G(V, E)$: spatial pattern

Output: Ψ: the promising node pairs up to depth level l of the quadtrees for each edge

$L = max(depth(ILQ_i), 1 \leq i \leq n)$

$EE \leftarrow \{e \in E$: e is exclusive$\}$

$IE \leftarrow \{e \in E$: e is inclusive$\}$

Reorder EE in form $\{e_1, ..., e_k\}$ s.t. $\#\Psi_{e_1(l-1)} \leq ... \leq \#\Psi_{e_k(l-1)}$

Reorder IE in form $\{e_{k+1}, ..., e_m\}$ s.t. $\#\Psi_{e_{k+1(l-1)}} \leq ... \leq \#\Psi_{e_m(l-1)}$

$E \leftarrow$ the concatenation of the arrays EE and IE

for $v_i \in V$ **do**

    $C_{i(l)} \leftarrow$ all nodes of ILQ_i at level l

for $e \in E$ **do**

    $\Psi_{e(l)} \leftarrow \emptyset$

    for $(n_i, n_j) \in \Psi_{e(l-1)}$ **do**

        if $n_i \in C_{i(l)} \wedge n_j \in C_{j(l)}$ **then**

            for $(n'_i, n'_j) \in n_i.children \times n_j.children$ **do**

                if (n'_i, n'_j) *is promising for edge* e *according to Def. 4* **then**

                    $\Psi_{e(l)}.add((n'_i, n'_j))$

    $v_i, v_j \leftarrow$ the endpoint vertices of e

    $N_i \leftarrow$ all nodes of ILQ_i at level l figuring at some promising pair of $\Psi_{e(l)}$

    $N_j \leftarrow$ all nodes of ILQ_j at level l figuring at some promising pair of $\Psi_{e(l)}$

    $C_{i(l)} \leftarrow C_{i(l)} \cap N_i$

    $C_{j(l)} \leftarrow C_{j(l)} \cap N_j$

return Ψ

Algorithm 3: GENERATE_CONNECTED_EDGES_ORDERING

Input: Ψ: promising node or object pairs for the edges, $G(V, E)$: spatial pattern, alt: a boolean parameter (false specifies a greedy-minimum reordering, true specifies a greedy-alternating reordering

Output: Π: the edges from E in a greedy and connected reordering

```
1  Π ← ∅ // keep the edges already added to the reordering
2  e ← the edge from E that minimizes #Ψ_e
3  Π.add(e)
4  B ← ∅ //keep visited vertices with remaining edges
5  v_i, v_j ← the endpoint vertices of e
6  if v_i is incident to an edge besides e then
7      B.add(v_i)
8  v ← v_j //keep the current vertex
9  while E\Π ≠ ∅ (there are edges left) do
10     N ← {v' ∈ v.neighbors: e(v, v') ∉ Π}
11     if N ≠ ∅ then
12         B.add(v)
13         if alt is true ∧ #Π is an odd integer then
14             v' ← the vertex from N s.t. the edge e(v, v') maximizes #Ψ_e
15         else
16             v' ← the vertex from N s.t. the edge e(v, v') minimizes #Ψ_e
17         e ← the edge linking v to v'
18         Π.add(e)
19         v ← v'
20     else
21         B ← B\{v}
22         ℵ ← the set of vertices v' s.t. ∃e(v, v') ∈ E\Π with v ∈ B
23         if alt is true ∧ #Π is an odd integer then
24             e ← the edge e(v, v') s.t. v ∈ B ∧ v' ∈ ℵ ∧ e(v, v') that maximizes #Ψ_e
25         else
26             e ← the edge e(v, v') s.t. v ∈ B ∧ v' ∈ ℵ ∧ e(v, v') that minimizes #Ψ_e
27         Π.add(e)
28         v ← the endpoint vertex from e that is not in B
29 return Π
```

4.2.2 Example of QQESPM-Quadtree execution

Consider an illustrative scenario where the algorithm is tasked with identifying matches for the spatial pattern shown in Figure 4.1 (A) within a search area containing POIs, as depicted in Figure 4.1 (B). The dataset includes 3 commercial buildings ($CB1, CB2, CB3$), 2 schools ($S1, S2$), and 3 gyms ($G1, G2, G3$). Details of the keywords, polygonal boundaries, and centroids of the POIs are provided in Table 4.1. The algorithm requires a pre-built IL-Quadtree index based on the keywords and positions of the POIs, which consists of separate quadtrees for each keyword, constructed from all POIs associated with that keyword. The quadtrees for the keywords in the search pattern are illustrated in Figure 4.2. The QQESPM-Quadtree algorithm begins its search for promising nodes within these quadtrees. The vertices of the search pattern (commercial building, gym, and school) are V_1, V_2, and V_3, respectively, and the edges "commercial building - gym" and "commercial building - school" are denoted as e_{CG} and e_{CS}, respectively.

ID	Name	Keywords	Centroid	Geometry
S1	School 1	school	POINT (7 3)	-
S2	School 2	school	POINT (8 7)	-
G1	Gym 1	gym	POINT (1.9 3.5)	POLYGON ((1.4 3, 2.4 3, 2.4 4, 1.4 4, 1.4 3))
G2	Gym 2	gym	POINT (6 7)	POLYGON ((5.5 6.5, 6.5 6.5, 6.5 7.5, 5.5 7.5, 5.5 6.5))
G3	Gym 3	gym	POINT (1 2)	POLYGON ((0 1.5, 2 1.5, 2 2.5, 0 2.5, 0 1.5))
CB1	Commercial Building 1	commercial_building	POINT (1.7 4)	POLYGON ((1 3, 2.4 3, 2.4 5, 1 5, 1 3))
CB2	Commercial Building 2	commercial_building	POINT (6.5 6.8)	POLYGON ((5.5 6.1, 7.5 6.1, 7.5 7.5, 5.5 7.5, 5.5 6.1))
CB3	Commercial Building 3	commercial_building	POINT (9 8.5)	POLYGON ((8.5 7.5, 9.5 7.5, 9.5 9.5, 8.5 9.5, 8.5 7.5))

Table 4.1: Fictitious dataset of POIs in 2-D Euclidian space

For each edge in the search pattern, the algorithm identifies promising pairs of nodes at every depth level within the quadtrees. For the edge e_{CG}, a pair of nodes (N_i, N_j) from the "commercial building" and "gym" quadtrees, respectively, is considered promising if their bounding boxes intersect. Similarly, a pair of nodes from the "commercial building" and "school" quadtrees is deemed promising for e_{CS} if the minimum distance between their bounding boxes is less than 5.5 and the maximum distance is at least 2.5, in accordance with Definition 4.

For this example, the root nodes of the quadtrees for the objects "commercial building", "gym", and "school" are denoted as N_C, N_G, and N_S, respectively. Nodes descending from these root nodes are assigned subscripts indicating their path from the root. Specifically, the

southwest child of a node is labeled as child 0, the southeast as child 1, the northwest as child 2, and the northeast as child 3. For example, the southwest child of the northeast child of the root node for the "commercial building" would be denoted as $N_{C:30}$.

At the beginning of the processing of each depth level of the quadtrees, the edges of the search pattern are reordered to prioritize the processing of exclusive edges first. Consequently, the promising pairs of nodes for the edge e_{CS} are computed first. At the root depth level, the pair of root nodes (N_C, N_S) is the only promising pair for the edge e_{CS}. These nodes have identical bounding boxes, resulting in a minimum distance of 0 and a maximum distance of 14.14. Therefore, at the root level, the edges e_{CS} and e_{CG} each have only one promising pair of nodes.

At the first non-root depth level, the subdivisions of the quadtrees remain equivalent. All pairs of nodes at this level continue to be promising for edge e_{CG} as their bounding boxes intersect, and the minimum and maximum distances between the bounding boxes are promising for the distance thresholds of the edge e_{CS}. With four nodes at this level for each quadtree, both edges e_{CG} and e_{CS} will each have $4 \cdot 4 = 16$ promising pairs of nodes at the first level.

At the second depth level, the subdivisions of the quadtrees diverge, and not all pairs of nodes from the "commercial building" and "gym" quadtrees continue to intersect. Table 4.2 illustrates the bounding rectangles for each node in the "commercial building", "gym", and "school" quadtrees. Nodes that are not subdivided at this level are also represented in their entirety at the second depth level. Consequently, the quadtree for "commercial building" contains seven nodes at the second level, whereas the quadtree for "school" has four nodes. This results in $7 \cdot 4 = 28$ potential node pairs for the edge e_{CS} ("commercial building - school") at this level. Since all these node pairs are descendants of the promising pairs from the first level, each must be evaluated. After verifying the distances between the bounding boxes of these nodes, all 28 node pairs are considered promising for identifying solutions for the edge "commercial building - school".

Node	**Bounding Box (x_min, y_min, x_max, y_max)**
N_C	$[0.0, 0.0, 10.0, 10.0]$
$N_{C:0}$	$[0.0, 0.0, 5.0, 5.0]$
$N_{C:1}$	$[5.0, 0.0, 10.0, 5.0]$
$N_{C:2}$	$[0.0, 5.0, 5.0, 10.0]$
$N_{C:3}$	$[5.0, 5.0, 10.0, 10.0]$
$N_{C:30}$	$[5.0, 5.0, 7.5, 7.5]$
$N_{C:31}$	$[7.5, 5.0, 10.0, 7.5]$
$N_{C:32}$	$[5.0, 7.5, 7.5, 10.0]$
$N_{C:33}$	$[7.5, 7.5, 10.0, 10.0]$
N_G	$[0.0, 0.0, 10.0, 10.0]$
$N_{G:0}$	$[0.0, 0.0, 5.0, 5.0]$
$N_{G:00}$	$[0.0, 0.0, 2.5, 2.5]$
$N_{G:01}$	$[2.5, 0.0, 5.0, 2.5]$
$N_{G:02}$	$[0.0, 2.5, 2.5, 5.0]$
$N_{G:03}$	$[2.5, 2.5, 5.0, 5.0]$
$N_{G:1}$	$[5.0, 0.0, 10.0, 5.0]$
$N_{G:2}$	$[0.0, 5.0, 5.0, 10.0]$
$N_{G:3}$	$[5.0, 5.0, 10.0, 10.0]$
N_S	$[0.0, 0.0, 10.0, 10.0]$
$N_{S:0}$	$[0.0, 0.0, 5.0, 5.0]$
$N_{S:1}$	$[5.0, 0.0, 10.0, 5.0]$
$N_{S:2}$	$[0.0, 5.0, 5.0, 10.0]$
$N_{S:3}$	$[5.0, 5.0, 10.0, 10.0]$

Table 4.2: Bounding boxes of the nodes in the quadtrees “commercial building”, “gym” and “school”

For the edge e_{CG} ("commercial building - gym"), there are $7 \cdot 7 = 49$ potential node pairs at the second level. However, after verifying the intersections between the nodes' bounding boxes, only 26 pairs are considered promising for containing inner objects that satisfy the edge e_{CG}. Specifically, the algorithm identifies the following promising node pairs: $(N_{C:30}, N_{G:03})$, $(N_{C:31}, N_{G:1})$, $(N_{C:30}, N_{G:1})$, $(N_{C:32}, N_{G:2})$, $(N_{C:30}, N_{G:2})$, $(N_{C:31}, N_{G:3})$, $(N_{C:33}, N_{G:3})$, $(N_{C:32}, N_{G:3})$, $(N_{C:30}, N_{G:3})$, $(N_{C:1}, N_{G:01})$, $(N_{C:1}, N_{G:03})$, $(N_{C:1}, N_{G:1})$, $(N_{C:1}, N_{G:2})$, $(N_{C:1}, N_{G:3})$, $(N_{C:0}, N_{G:00})$, $(N_{C:0}, N_{G:01})$, $(N_{C:0}, N_{G:02})$, $(N_{C:0}, N_{G:03})$, $(N_{C:0}, N_{G:1})$, $(N_{C:0}, N_{G:2})$, $(N_{C:0}, N_{G:3})$, $(N_{C:2}, N_{G:02})$, $(N_{C:2}, N_{G:03})$, $(N_{C:2}, N_{G:1})$, $(N_{C:2}, N_{G:2})$ and $(N_{C:2}, N_{G:3})$.

Given that the second depth level constitutes the leaf level, the algorithm shifts its focus to processing objects within these promising leaf nodes to identify promising object pairs for each edge. The subsequent step in the search process involves examining the objects within the promising node pairs corresponding to the search pattern's edges. A pair of objects is considered promising for edge e_{13} if their locations meet the edge's distance constraints. In turn, a pair of objects is promising for edge e_{12} if they reside within the promising node pairs of this edge and the bounding boxes of their polygonal boundaries intersect. Since edge e_{12} involves a qualitative topological constraint defined by the "contains" relationship, the bounding boxes of objects satisfying this constraint necessarily intersect. Consequently, this stage eliminates unpromising object pairs concerning the topological constraint. At this phase, the algorithm identifies $(CB1, S1)$ as the only promising object pair for edge e_{CS} and the object pair $(CB1, G1)$ for edge e_{CG}. Notably, the commercial building $CB2$ contains a gym, but it fails to meet the exclusion constraint of edge e_{CS} due to its proximity to school $S2$, rendering this POI unable to satisfy the edge's exclusion requirements.

In some instances, a subset of edges within the search pattern involves all vertices. When this occurs, confirming promising object pairs for these edges results in a set of candidate objects for all vertices. In such cases, it becomes unnecessary to verify the objects within promising nodes for certain edges that share common endpoints with adjacent edges already examined. These edges, termed skip-edges, allow the algorithm to derive promising object pairs by evaluating the already gathered candidate objects for the vertices of these edges. These candidates are obtained from the object pairs that satisfy the adjacent edges previously processed. In this example, the search pattern does not contain skip-edges. However, if

a third edge were to establish a spatial constraint between the searched "school" and the "gym", it could be classified as a skip-edge since the promising objects for their vertices would already have been identified after computing the promising object pairs for the first two edges. Consequently, verifying the inner objects of its promising node pairs would be redundant.

The final step involves combining the solutions of adjacent edges to form tuples that represent the ultimate solutions for the entire search graph. By examining shared objects between adjacent edges, only the solutions that persist through the joins between the promising object pairs of adjacent edge are retained. The algorithm then performs a final verification to eliminate the solutions not meeting the spatial constraints of skip-edges or the exact topological relationship requirements, before outputting the final results.

In this example, the joining of object pairs for the "commercial building - gym" and "commercial building - school" edges yields a single complete solution: $(CB1, G1, S1)$. This outcome represents the sole solution for the spatial search pattern shown in Figure 4.1 (A) within the POI dataset depicted in Figure 4.1 (B).

4.2.3 QQESPM-Quadtree Solution

The QQESPM-Quadtree solution is designed as a module that employs the Algorithm 1 to run QQ-SPM queries. It also manages large datasets with an ad hoc scalable implementation of the IL-Quadtree indexing stored on disk as binary files, for faster data access. The proposed QQESPM-Quadtree solution is an ad hoc solution for QQ-SPM searching, not requiring any geospatial storage technology, since it processes datasets of spatial objects (e.g., POIs) using its proper IL-Quadtree implemented on disk.

The proposed QQESPM-Quadtre allows a sliced data reading strategy, ensuring that only the necessary IL-Quadtree nodes and objects are read from disk into memory during queries. This allows for querying large databases without overwhelming RAM. This process provides scalability and enables the processing of queries against large datasets.

4.2.4 Choosing an Order for Joining Edges

After computing the matching pairs of objects for each edge, the algorithm must join these candidate solutions edge by edge to find the solutions for the entire search graph. The order in which edges are processed during this joining step can significantly affect the computational cost.

Consider a search graph with three edges ($e_{1,2}$, $e_{2,3}$, $e_{3,4}$), where each edge $e_{i,j}$ connects vertices v_i and v_j. Suppose edge $e_{1,2}$ has 1,000 candidate object pairs, edge $e_{2,3}$ has 10,000 candidate object pairs, and edge $e_{3,4}$ has 40 candidate object pairs. If the edges are joined in the order ($e_{1,2}$, $e_{2,3}$, $e_{3,4}$), the first step will join the solutions of $e_{1,2}$ and $e_{2,3}$, requiring $1,000 \times 10,000 = 10,000,000$ pair-wise tests, which may cause a significant overhead. If this join yields 20,000 solutions, these partial solutions must then be joined with the 40 solutions of edge $e_{3,4}$, requiring $20,000 \times 40 = 800,000$ additional computations. Thus, the total operations performed would be $10,000,000 + 800,000 = 10,800,000$.

Alternatively, if the edge solutions are joined in the order ($e_{3,4}$, $e_{2,3}$, $e_{1,2}$), the first step will join the 40 solutions of edge $e_{3,4}$ with the 10,000 solutions of edge $e_{2,3}$, requiring 400,000 operations. If this step yields 500 partial solutions, the next step will join these 500 partial solutions with the 1,000 solutions of edge $e_{1,2}$, costing 500,000 operations. For this strategy, the total operations performed would be $400,000 + 500,000 = 900,000$, which

is significantly fewer than the 10,800,000 required by the previous strategy.

This example demonstrates that the choice of edge order in the join phase significantly impacts the computational cost of solving the problem. In practice, it is impossible to predict the exact number of operations for a specific edge order before the joining process, as the total number of partial solutions after each join step is unknown until the join is performed. Based on these intuitions, the QQESPM-Quadtree algorithm applies a greedy alternating strategy, which ultimately provides a median performance in terms of operation cost. This strategy reduces the computational cost of joins by controlling the magnitude of the total partially constructed solutions during the join phase.

The strategy for ordering edges in the join phase follows this rule: the first edge is the one with the fewest solutions. Once the n-th edge is chosen, the $(n+1)$-th edge is preferably adjacent to the n-th edge. If there are adjacent edges that have not yet been chosen, and if the last chosen edge minimized the number of solutions, the next edge will be the one with the most solutions among the adjacent edges. Conversely, if the last chosen edge maximized the number of solutions, the next edge will be the one with the fewest solutions among the adjacent edges. If no adjacent edges remain, select the next edge from those that share at least one vertex with an already chosen edge. This alternation between selecting the edge with the fewest solutions and the edge with the most solutions, among the available adjacent edges at each step, is the reason why this ordering method is called alternated greedy edge ordering. This ordering is described in Algorithm 3.

The ordering of edges for the join phase is a connected ordering, where each next edge is adjacent to some of the already processed edges. This allows using promising pairs of objects of processed edges to eliminate many unpromising pairs of objects for subsequent edges, as the next edge always shares a vertex with an already processed edge, and the matched spatial object associated with the common vertex of two adjacent edges must be the same object for the matching pairs of objects of both edges.

4.3 QQESPM-Elastic

This section presents a second, independent approach for solving the QQ-SPM problem, based solely on queries executed against an Elasticsearch[1] node. The details on the design of such solution, presented in this section, provide a positive answer for the first half of the Research Question RQ_3.

Elasticsearch is a search engine software that provides a distributed data storage system with reverse indexing methods focused on textual search efficiency. It features an HTTP web interface and stores documents in JSON format. Elasticsearch offers many capabilities for spatial data processing[2], including native spatial indexing, spatial data types, and functions for searching by distance and connectivity. These features enable Elasticsearch to efficiently handle QQ-SPM queries. Therefore, the QQESPM-Elastic module is proposed as a pipeline that automatically converts QQ-SPM searches into a series of Elasticsearch requests, ultimately yielding the final solutions to the spatial search pattern.

By consulting the official documentation, the following valuable resources for designing the QQESPM-Elastic module were identified:

- Two spatial data types:
  - geopoint: Stores latitude and longitude values.
  - geoshape: Stores polygons and various shapes using textual encoding methods.
- *geo_distance* query: Matches documents containing a geo_point or geo_shape within a specified distance from a specific geopoint.
- *geo_shape* query: Matches documents containing a geo_point or geo_shape that has a specified connectivity relation with a given geo_shape. Possible relations include intersects, contains, within, and disjoint.

Utilizing these inherent functionalities of Elasticsearch, we design three fundamental spatial query operations to form the foundation of a novel QQ-SPM solution approach. These elementary operations of the QQESPM-Elastic module are following outlined.

[1] https://www.elastic.co/

[2] https://www.elastic.co/guide/en/elasticsearch/reference/current/geo-queries.html

4.3.1 Elementary Operations

The data objects stored on Elasticsearch are referred to as documents. The proposed QQESPM-Elastic module executes QQ-SPM queries through repeated calls to only three elementary operations. These operations are outlined below.

- EO_1 (**search_by_keyword**): Retrieves all documents containing a specified keyword.
- EO_2 (**search_by_distance_and_keyword**): Retrieves all documents containing a specified keyword and whose associated spatial shape's centroid is within a specified distance from a given latitude and longitude.
- EO_3 (**search_by_relation_and_keyword**): Retrieves all documents containing a specified keyword and whose spatial shape has a specific connectivity relation with the shape of a particular document.

The function depicted in Source Code 4.1 constructs the Elasticsearch query for the elementary query EO_3. Subsequently, an explanation is described on how to employ these three basic operations to devise a comprehensive solution for addressing the QQ-SPM problem leveraging Elasticsearch.

Source Code 4.1: Function that generates the elementary operation EO_3 for Elasticsearch

```
def search_by_keyword_and_relation_elastic(es, index_name, kw:str, hit:
    dict, relation, size = 100000, count = False):
    query = {
        "query": {
            "bool": {
                "filter": [
                    {"match": {"properties.keywords.keyword": kw}},
                    {
                        "geo_shape": {
                        "geometry": {
                            "indexed_shape": {
                            "index": index_name,
                            "id": hit['_id'],
                            "path": "geometry"
                            },
```

```
                        "relation": relation # intersects, contains,
                            within
                    }
                    }
                }
            ]
        }
    }
}
if count:
    return es.count(index = index_name, body = query)['count']
else:
    query["fields"] = ["geometry", "properties.osm_id"]
    query["_source"] = False
    query["size"] = size
    return query
```

4.3.2 QQESPM-Elastic Search Procedure

The search procedure of QQESPM-Elastic is detailed in Algorithm 4 (QQESPM-EO), which leverages the elementary operations EO_1, EO_2 and EO_3. The algorithm accepts a dataset D of geo-textual objects and a spatial pattern graph $G(V, E)$ as input, and outputs ψ, representing all matches of G within D. The procedure begins by calculating the frequency of each keyword within the search pattern (Lines 2 and 3 of Algorithm 4). If any keyword is absent from the dataset, the search pattern has no solutions (Lines 4 and 5 of Algorithm 4). The algorithm then selects the initial vertex v based on the lowest keyword frequency in the dataset to start the search (Line 6 of Algorithm 4). Using EO_1, it identifies all objects containing this keyword (Line 7). The variable Θ tracks the edges already processed, while the variable B stores the visited vertices with remaining edges.

The algorithm employs a `While` loop to iterate through the edges of the search pattern as long as there are remaining edges (Line 10 of Algorithm 4). The next vertex in the path is preferably selected from the neighboring vertices of the current one. If there are any neighboring vertices with remaining edges, the next vertex v' is chosen from among the neighbors of the current vertex v, prioritizing the vertex with the minimum frequency in

the dataset. The promising object pairs for the edge $e(v, v')$ are then computed using the `Get_Object_Pairs_EO` subroutine described in Algorithm 5. For each candidate object o_i corresponding to vertex v, the related objects o_j for vertex v' are identified as those pairs (o_i, o_j) that satisfy the spatial constraints of edge e. If the edge is qualitative, the algorithm uses the elementary operation EO_3 (filter by topological relation with o_i). If the edge is quantitative, it utilizes operation EO_2 (filter by distance from o_i). This procedure is outlined in Lines 1 through 8 of Algorithm 5. The promising object pairs for edge e are then returned to Algorithm 4 in the variable S_e (Line 16 of Algorithm 4).

The edge e is then marked as solved (Line 17), and the candidate objects for the endpoint vertices of e are updated by filtering to include only those present in S_e. Likewise, the frequencies of the keywords associated with the vertices of e are updated to reflect the total counts of candidate objects for these endpoint vertices. If the current vertex v has no neighboring vertices with remaining edges, the next vertex is selected from the visited vertices with remaining edges, stored in variable B, based on the one with the minimum keyword frequency (Lines 21 through 24 of Algorithm 4).

The edges are then reordered in a new sequence to reduce the cost of the upcoming joining phase, which consolidates the promising object pairs for all edges into joined solutions for the full search pattern graph. This edge reordering is identical to the reordering process in QQESPM-Quadtree for the joining phase (Line 25 of Algorithm 4). It employs a connected greedy alternating strategy, generating an edge order that minimizes heavy computations during the joining step. The joining of edge solutions follows a process similar to that in QQESPM-Quadtree, filtering out promising object pairs that do not share a common object at the common vertex of adjacent edges (Lines 26 through 28 of Algorithm 4). The fully joined solutions are then returned in the variable ψ (Line 29).

Algorithm 4: QQESPM-EO

Input: D: dataset of geo-textual objects in a Elasticsearch index, $G(V, E)$: spatial pattern

Output: ψ: the matches of the spatial pattern G

$K \leftarrow$ // the set of keywords of all vertices in V
for *each keyword* $w \in K$ **do**
    $f_w \leftarrow$ the frequency of keyword w in the dataset D
if $\exists w \in K : f_w = 0$ **then**
    return $\emptyset$
$v \leftarrow$ the vertex $v \in V$ that minimizes $f_{v.keyword}$
$C_v \leftarrow$ RUN EO_1 for $v.keyword$ in dataset D // the candidate objects for the vertex v
$\Theta \leftarrow \emptyset$ // processed edges
$B \leftarrow \{v\}$ //keep visited vertices with remaining edges
while $E \backslash \Theta \neq \emptyset \wedge B \neq \emptyset$ *(there are edges left)* **do**
    $N \leftarrow \{v' \in v.\text{neighbors: } e(v, v') \notin \Theta\}$
    if $N \neq \emptyset$ **then**
        B.add(v)
        $v' \leftarrow$ the vertex $v_i \in N$ that minimizes $f_{v_i.keyword}$
        $e \leftarrow$ the edge linking v to v'
        $S_e \leftarrow$ RUN Get_Object_Pairs_EO($e(v, v'), C_v$) // the promising object pairs for edge e
        Θ.add(e)
        Filter C_v and $C_{v'}$ to only the objects appearing at S_e
        Update the frequencies of keywords $f_{v.keyword}$ and $f_{v'.keyword}$ using S_e
        $v \leftarrow v'$
    else
        $B \leftarrow B \backslash \{v\}$
        if $B \neq \emptyset$ **then**
            $v \leftarrow$ the vertex $v \in B$ that minimizes $f_{v.keyword}$
$E \leftarrow$ RUN Generate_Connected_Edges_Ordering(S, $G(V, E)$, true)// reorder the edges for the join phase
$\psi \leftarrow \emptyset$ // keep the partially constructed matches of the search pattern
for *each edge* $e \in E$ **do**
    $\psi \leftarrow \psi$.join(S_e)
return ψ

Algorithm 5: GET_OBJECT_PAIRS_EO

Input: $e(v, v')$: an edge of a spatial pattern, C_v: the candidate objects for the vertex v

Output: S_e: the promising object pairs for edge e

```
1 S_e ← ∅
2 for o_i ∈ C_v do
3     if edge e is qualitative with relation ℜ ≠ "disjoint" then
4         O_j ← RUN EO_3 for (o_i, v'.keyword, ℜ) in D // the set of objects o_j s.t. the pair
              (o_i, o_j) is promising for edge e
5     else
6         O_j ← RUN EO_2 for (o_i, v'.keyword, e.l_ij, e.u_ij) in D // the set of objects o_j s.t.
              the pair (o_i, o_j) is promising for edge e
7     S'_e ← the set of pairs (o_i, o_j) with o_j ∈ O_j
8     S_e ← S_e ∪ S'_e
9 return S_e
```

4.3.3 Implementation Decisions for QQESPM-Elastic

Several implementation decisions were applied to enhance the execution time of the QQESPM-Elastic module. Firstly, the design of the QQESPM-Elastic opts for "*filter*" queries over "*must*" queries. While both types of search queries are available in Elasticsearch and are based on boolean conditions, "*must*" queries sort the results based on computed scores, whereas "*filter*" queries solely select documents that meet the boolean conditions without computing scores. Given that the QQ-SPM search objective is to retrieve all matches of the search pattern without prioritizing or ranking the relevance, "*filter*" queries prove more efficient and aligned with this purpose.

To enhance the performance of the proposed QQESPM-Elastic module, the MultiSearch API is employed. This feature of Elasticsearch enables the parallel processing of multiple independent queries. For example, when dealing with a vertex v_i containing $10,000$ candidate objects, and processing the subsequent vertex, its neighbor v_j, the QQESPM-Elastic procedure needs to execute an elementary query operation for each of these $10,000$ candidate objects of v_i, is based on the keyword of v_j and the constraints of the edge linking v_i and v_j. By leveraging the MultiSearch feature of Elasticsearch, these $10,000$ are executed in parallel. Ultimately, this approach eliminates the need to submit Elasticsearch requests within code repetition loops, resulting in a significant performance improvement.

The third feature for enhancing performance was the implementation of the same connected greedy alternating edges ordering strategy utilized in QQESPM-Quadtree within the joining process of QQESPM-Elastic. This approach can reduce computational overhead during the composition of partial solutions for the search pattern, by controling the magnitude of the total number of partial solutions after each join of two edges. This strategy avoids excessively large Cartesian products of edges solutions in the join phase.

4.4 QQESPM-SQL

This section describes the third approach for solving the QQ-SPM search problem. Such approach relies exclusively on SQL queries executed on a PostgreSQL[3] database server with the PostGIS[4] spatial extension enabled. This section provides detailed insights into the design of this approach, presenting a positive response for the second half of the Research Question RQ_3.

4.4.1 PostGIS Spatial Queries

The PostGIS extension extends the functionality of the PostgreSQL relational database by incorporating support for storing, indexing, and querying geospatial data, along with offering a wide range of spatial functions and operations[5]. Consequently, databases with PostGIS

[3]https://www.postgresql.org/

[4]https://postgis.net/

[5]https://postgis.net/docs/using_postgis_query.html

capabilities serve as an appropriate platform for executing QQ-SPM queries. To leverage this environment effectively, the QQESPM-SQL solution is developed. This solution automatically converts QQ-SPM query requirements into a single SQL query, leveraging the PostGIS spatial functions. The generated SQL query efficiently retrieves all groups of objects that meet the criteria specified in the spatial pattern graph.

The QQESPM-SQL solution's efficient design draws from recommendations found in the official documentation and insights gained from experimental executions. This investigation yielded valuable information that guided a set of informed implementation decisions, leading to performance improvements.

Upon reviewing the official documentation, the following valuable resources for designing the QQESPM-SQL solution were identified:

- Two important spatial data types: *ST_Point* and *ST_Polygon* which are generally stored as Well-known binary (WKB) format in a database table.
- Two important distance functions *ST_DistanceSphere* and *ST_DWithin*, which return the distance in meters between two geometries.
- Topological relation functions such as *ST_Intersects*, *ST_Covers* and *ST_Disjoint* which enable the verification of whether two geometries satisfy a given named spatial predicates, based on the DE-9IM model formalized in[33, 34, 47].

4.4.2 QQESPM-SQL Approach

The proposed QQESPM-SQL solution accepts a spatial pattern graph as input, which can be provided in either JSON format or as a binary variable format. This module adequately translates the spatial pattern requirements into a comprehensive SQL query, encompassing all the criteria of the spatial search pattern and capable of retrieving all of its matches. In composing this SQL query from the spatial pattern, two primary strategies can be employed. The first strategy utilizes an implicit join, while the second entails designing an efficient greedy and explicit join order. Remarkably, both approaches yielded similar query performance in experimental executions.

QQESPM-SQL implicit join approach

In this approach, the query is constructed using the following strategy: First, a temporary table for each keyword of the search pattern is declared using the WITH clause. The SELECT clause specifies the return of the IDs column of the objects, maintaining the same order as the vertices listed in the search pattern. The FROM clause employs an implicit JOIN linking all the temporary tables of the keywords declared in the WITH clause. The WHERE clause then consolidates all the quantitative and qualitative requirements of all edges in the spatial pattern graph. The order of the requirements follows the order of the edges list in the search pattern. Source Code 4.2 provides an example of a spatial pattern JSON, and Source Code 4.3 illustrates its corresponding SQL query generated by this approach.

Source Code 4.2: A spatial pattern in JSON

```
{
  "vertices": [
    {
      "id": 0, "keyword": "waste_basket"
    },
    {
      "id": 1, "keyword": "bicycle"
    },
    {
      "id": 2, "keyword": "travel_agency"
    }
  ],
  "edges": [
    {
      "id": 0,
      "vi": 0,
      "vj": 1,
      "lij": 0,
      "uij": Infinity,
      "sign": "-",
      "relation": "within"
    },
    {
```

```
      "id": 1,
      "vi": 1,
      "vj": 2,
      "lij": 81.36447702059968,
      "uij": 367.12676197537536,
      "sign": "<",
      "relation": null
    }
  ]
}
```

Source Code 4.3: SQL query with implicit join

```
WITH
    tb_waste_basket AS
    (SELECT * FROM pois WHERE amenity = 'waste_basket' OR building = '
        waste_basket' OR landuse = 'waste_basket' OR leisure = '
        waste_basket' OR shop = 'waste_basket' OR tourism = 'waste_basket'
         ),
    tb_bicycle AS
    (SELECT * FROM pois WHERE amenity = 'bicycle' OR building = 'bicycle'
         OR landuse = 'bicycle' OR leisure = 'bicycle' OR shop = 'bicycle'
         OR tourism = 'bicycle' ),
    tb_travel_agency AS
    (SELECT * FROM pois WHERE amenity = 'travel_agency' OR building = '
        travel_agency' OR landuse = 'travel_agency' OR leisure = '
        travel_agency' OR shop = 'travel_agency' OR tourism = '
        travel_agency' )
SELECT tb_waste_basket.osm_id AS tb_waste_basket_id, tb_bicycle.osm_id AS
    tb_bicycle_id, tb_travel_agency.osm_id AS tb_travel_agency_id
FROM tb_waste_basket, tb_bicycle, tb_travel_agency
WHERE
    ST_CoveredBy(tb_waste_basket.geometry, tb_bicycle.geometry) AND
    ST_DistanceSphere(tb_bicycle.centroid, tb_travel_agency.centroid)
       BETWEEN 81.36447702059968 AND 367.12676197537536  AND
    NOT EXISTS (SELECT 1 FROM tb_bicycle aux WHERE ST_DWithin(
        tb_travel_agency.centroid::geography, aux.centroid::geography,
        81.36447702059968, false))
```

QQESPM-SQL explicit greedy join approach

In this approach, the query is structured as follows: Similar to the first method, temporary tables for the keywords in the search pattern are declared using the WITH clause. The SELECT clause retrieves the IDs of matching objects based on the vertex order in the search pattern. However, there is a difference in the FROM clause. Instead of the FROM+WHERE strategy, an explicit INNER JOIN operation is used.

To determine the order of temporary tables in the JOINs, this explicit greedy join approach employs a greedy strategy to order the vertices. This order alternates based on the frequency in dataset of each keyword of the search, similar to the greedy alternated edges ordering utilized in QQESPM-Quadtree. The initial temporary table for the JOIN is the one associated with the least frequent keyword of the search pattern. Subsequently, the keyword for the JOIN is chosen among the neighboring vertices, either minimizing or maximizing the frequency of the keyword, following the alternated strategy utilized in QQESPM-Quadtree and QQESPM-Elastic. After the inclusion of a specific vertex in the JOIN clause, the boolean condition for the next ON clause is composed with the requirements of all edges linking the currently included vertex with all the previously included vertices.

Consider a spatial pattern with vertices v_1, v_2, v_3, associated with keywords having the frequencies of $100, 1000, 300$, respectively. Suppose this search pattern includes edges $e_{1,2}, e_{1,3}, e_{2,3}$. The order of vertices for the INNER JOIN will be v_1, v_3, v_2. Let's denote the temporary tables for the keywords of v_1, v_2, v_3 as tb_1, tb_2, and tb_3, respectively. The condition for the tb_1 INNER JOIN tb_3 will encompass all the constraints of the edge $e_{1,3}$. Subsequently, the condition for the subsequent INNER JOIN tb_2 will include all the constraints of the edges $e_{1,2}$ and $e_{2,3}$. These edges link the current vertex v_2 with all the vertices previously included in the JOIN, namely v_1 and v_3. In the example provided in Source Code 4.2, let's assume the frequencies of the keywords "*waste_basket*", "*bicycle*", and "*travel_agency*" are $1280, 120$ and 118, respectively. The resulting SQL query produced by this approach is shown in Source Code 4.4 and follow the order of temporary tables: *tb_travel_agency*, *tb_bicycle*, *tb_waste_basket*.

Source Code 4.4: SQL query with explicit join

```
WITH
    tb_travel_agency AS
```

```
    (SELECT * FROM pois WHERE amenity = 'travel_agency' OR building = '
        travel_agency' OR landuse = 'travel_agency' OR leisure = '
        travel_agency' OR shop = 'travel_agency' OR tourism = '
        travel_agency' ),
    tb_bicycle AS
    (SELECT * FROM pois WHERE amenity = 'bicycle' OR building = 'bicycle'
         OR landuse = 'bicycle' OR leisure = 'bicycle' OR shop = 'bicycle'
         OR tourism = 'bicycle' ),
    tb_waste_basket AS
    (SELECT * FROM pois WHERE amenity = 'waste_basket' OR building = '
        waste_basket' OR landuse = 'waste_basket' OR leisure = '
        waste_basket' OR shop = 'waste_basket' OR tourism = 'waste_basket'
         )
SELECT tb_waste_basket.osm_id AS tb_waste_basket_id, tb_bicycle.osm_id AS
    tb_bicycle_id, tb_travel_agency.osm_id AS tb_travel_agency_id
FROM tb_travel_agency
INNER JOIN tb_bicycle
ON
    ST_DistanceSphere(tb_bicycle.centroid, tb_travel_agency.centroid)
       BETWEEN 81.36447702059968 AND 367.12676197537536  AND
    NOT EXISTS (SELECT 1 FROM tb_bicycle aux WHERE ST_DWithin(
        tb_travel_agency.centroid::geography, aux.centroid::geography,
        81.36447702059968, false))
INNER JOIN tb_waste_basket
ON
    ST_Within(tb_waste_basket.geometry, tb_bicycle.geometry)
```

The explicit greedy join approach achieved performance comparable to the implicit join in various query execution tests. Notably, the queries constructed in Source Codes 4.3 and 4.4 generated identical query plans when analyzed with the EXPLAIN command. Consequently, the implicit join method was retained as the default JOIN approach for QQESPM-SQL due to its simpler query syntax.

4.4.3 Implementation Decisions for QQESPM-SQL

Several decisions were chosen to enhance the performance of the spatial SQL queries. Firstly, QQESPM-SQL uses two spatial indexes for the objects' geometries: a GiST-based index and an SP-GiST-based index. The GiST index is widely used and versatile, offering excellent query performance. However, when a GIS data table exceeds a few thousand rows, an SP-GiST index becomes more efficient for speeding up spatial searches, as suggested in the PostgreSQL documentation.

The second implementation decision involves using the EXISTS operator in PostgreSQL to model proximity avoidance constraints. This approach significantly improves query performance compared to using the COUNT function, which has shown poor performance in handling such constraints.

Additionally, an appropriate use of the ST_DistanceSphere and ST_DWithin functions is employed. Query executions indicate that when the query involves two varying geometries, the ST_DistanceSphere function performs faster in condition clauses. Conversely, when searching for a pair of geometries where one is fixed and only the other varies, the ST_DWithin function is the most efficient choice. This observation aligns with PostgreSQL's official documentation.

The QQESPM-SQL module leverages all the above mentioned implementation decisions. When dealing with proximity avoidance constraints, where the central object remains fixed, the library generates a query with the EXISTS filter in combination with the ST_DWithin function. In contrast, for distance interval constraints where both objects vary, the optimal function to use is ST_DistanceSphere.

The last performance enhancement involves optimizing the database parameter settings. Specifically, in PostgreSQL various parameters govern machine resource usage, which can be configured in the *postgresql.conf* file. According to official documentation, setting the shared_buffers parameter to 25% of the existing physical RAM is optimal. While other parameters also exist, the documentation lacks clear recommendations for those. To address this, a widely-used developer tool called PGTune[6] was utilized. By inputting machine configuration details, PGTune provides optimal settings for over ten internal PostgreSQL parameters. In this context, the query environment is optimally setup using

[6]https://pgtune.leopard.in.ua/#/

the recommendations of this tool.

4.5 Generalizing the QQ-SPM Query

This master's thesis introduces the QQ-SPM search as a solution specifically motivated by POI search scenarios. Appendix A demonstrates a POI search application prototype using the QQ-SPM search format. Nevertheless, this search concept can be extended to encompass various geo-textual searches beyond POI search. Essentially, it can be applied to any dataset containing geo-textual objects with spatial shape geometries defining the objects' spatial scopes or boundaries, along with associated textual content. For instance, the QQ-SPM query could be employed to explore entire cities, web documents, or any other form of geo-textual data.

Moreover, this research incorporated as default the geodesic distance to fulfill the distance criteria of the search patterns. However, the proposed solution is agnostic to implementation details, allowing for the interchangeability of distance metrics between objects. For instance, alternatives like road network distance, which calculates the shortest path along existing thoroughfares such as streets and avenues, and Euclidean distance can be seamlessly chosen. Additionally, the indexing approach adopted by the QQESPM algorithm was the IL-quadtree. Nevertheless, various other geo-textual indexing strategies can be applied to the dataset, only requiring adjustments on the QQ-SPM search algorithms to enable the handling of different indexes and distance metrics while executing QQ-SPM queries.

4.6 Final Considerations

This chapter introduced the formal definition of the QQ-SPM search problem and presented two theorems foundational to an algorithmic solution. It also described the design of three libraries to efficiently address the search problem. The first solution, QQESPM-Quadtree, leverages these theorems to exploit the IL-Quadtree geo-textual index. The second, QQESPM-Elastic, converts spatial requirements into native Elasticsearch geo-queries. Lastly, QQESPM-SQL translates spatial search requirements into efficient SQL

queries employing PostGIS. The upcoming chapter presents the methodology and results of performance experiments comparing these three solutions, highlighting their strengths and weaknesses.

Chapter 5

Performance Experiments

This chapter presents a performance comparison for the consolidated versions of the three QQ-SPM solutions presented in Chapter 4. Each solution was implemented as a Python library. This chapter outlines the experimental methodology used for the comparison and discusses the obtained results.

5.1 Datasets

The experiments utilized two datasets of POIs extracted from OpenStreetMap[1]. Table 5.1 presents the POI statistics for each dataset. These POIs are situated within a bounding box centered around London, UK. The datasets include various POI types, such as the OSM tags *amenity*, *shop*, *tourism*, *landuse*, *leisure*, and *building*. Dataset 1 is a subset of Dataset 2 and is used to compare QQESPM-Quadtree with a baseline search method that has proven inefficient for QQ-SPM queries. In turn, Dataset 2 is a bigger dataset, employed for more intensive searches, comparing the three proposed QQ-SPM solutions: QQESPM-Quadtree, QQESPM-Elastic, and QQESPM-SQL.

Dataset	#POIs	Extension	Total keywords	Distinct keywords
Dataset 1	$38,000$	$5.5\text{km} \times 5.5\text{km}$	$39,378$	514
Dataset 2	$127,975$	$12\text{km} \times 12\text{km}$	$131,041$	760

Table 5.1: Datasets Statistics

[1] https://www.openstreetmap.org/

The design of spatial pattern search systems necessitates clean and comprehensive data sources in practical applications. While some researches address these challenges[45, 55], this master's thesis does not delve into this direction, as the primary focus is to evaluate the performance of efficient solutions for the QQ-SPM query. Thus, this master's thesis serves as foundational research for investigating the performance of QQ-SPM queries and precedes practical considerations.

The proposed solution leverages polygonal boundaries for target objects, employing a buffering pipeline to increase the likelihood of finding intersecting geometries within the datasets. This process ultimately allows more results for qualitative topological-constrained queries. Initially, the original POI geometries are replaced with their respective convex hulls and undergo a polygon simplification process. This pipeline utilizes the *Shapely*[2] Python library. The use of convex hulls and polygon simplifications aims to correct invalid polygons with corrupted data and create simpler boundaries for the POIs. This approach generates boundaries that tolerate inaccuracies in the annotated geometry data, ultimately facilitating faster retrieval of intersecting geometries due to the simplification process. Moreover, it enhances the frequency of such retrievals, as the boundaries of the POIs are represented by simpler and expanded areas, increasing the potential for intersections between neighboring POIs.

Figure 5.1 illustrates the execution of this buffering pipeline. Initially, the polygon geometries typically consist of shapes with numerous sides. However, such complex shapes are often unnecessary for determining whether two neighboring geometries should be considered intersecting in a query. Figure 5.1 (C) shows the geometry after the convex hull and simplification procedures. This form is further expanded by the subsequent buffering process, which increases the areas associated with the POIs. The simplification tolerance parameter for the *shapely.simplify* function was experimentally set to 0.05. Higher tolerances produce more simplified polygons, resulting in fewer vertices.

The buffer size for the geometries is computed as follows. The average side length for each polygon geometry in the datasets is calculated. The median value of the average lengths of all polygons is then set as the default buffer size for geometries with point shape. Each polygon shape, in turn, receives a buffer with a size equal to 10% of its respective

[2]https://shapely.readthedocs.io/

average length. As a result of this pipeline, the total area occupied by geometries in the datasets increased by 54%. Figure 5.2 examplifies how this buffering pipeline increased the chance of intersecting neighboring geometries. The polygons in orange represent the original geometries of POIs, and the polygons in pink represent the post-processed geometries of POIs. Noticeably the number of intersecting geometries increased with the buffering pipeline. The smaller pink squares without orange subregions represent buffers of point geometries.

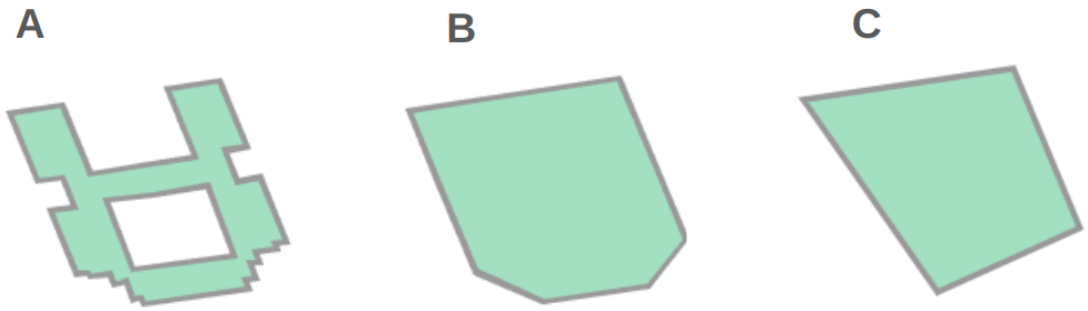

Figure 5.1: Example of an original geometry (A), its convex hull (B) and its simplification (C)

Figure 5.2: POIs geometries before (orange) and after (pink) buffering pipeline

An extract of the Dataset 2 is shown in Table 5.2. During a search, the proposed

QQESPM solutions scan objects in the dataset, looking for keywords from the search pattern. These keywords should appear in the keyword columns (e.g., “amenity”, “shop”, “tourism”, etc.). For instance, if a query includes the keyword “cafe”, the POI with osm_id 249217389 in the Table 5.2 (line 5) would be a candidate matching object.

osm_id	name	amenity	shop	tourism	landuse	leisure	building	geometry	centroid
991607788	Kensington West						apartments	POLYGON ((-0.21319 51.49516, ...	POINT (-0.21296 51.49553)
154951930							residential	POLYGON ((-0.18174 51.52468, ...	POINT (-0.18177 51.52474)
669557409							residential	POLYGON ((-0.14745 51.52154, ...	POINT (-0.14742 51.52158)
249217389	Moscos Cafe	cafe						POLYGON ((-0.15393 51.51315, ...	POINT (-0.15378 51.51322)
149065743	No 74 Hair Beauty		hairdresser					POLYGON ((-0.10270 51.52439, ...	POINT (-0.10277 51.52443)

Table 5.2: Extract of the Datasets

5.2 Solution Approaches Used in the Experiments

The experiments involve conducting spatial pattern searches using four distinct solution approaches for QQ-SPM queries. Specifically, a baseline approach called ESPM+TV is utilized for solving QQ-SPM queries, alongside the three proposed approaches: QQESPM-Quadtree, QQESPM-Elastic, and QQESPM-SQL.

The baseline approach, ESPM+TV, involves using the ESPM algorithm[24] to address the distance constraints of the search pattern, followed by a final topological verification. This verification filters out the results from ESPM that do not meet the topological requirements of the QQ-SPM search graph. Thus, ESPM+TV can be considered a straightforward or basic solution for QQ-SPM queries, without any optimization for query performance. For edges with topological constraints but without specific distance requirements, the ESPM+TV approach assumes a default distance threshold of $1km$, as the ESPM algorithm is designed to handle only distance constraints.

Two sets of query executions were conducted to evaluate the different solution approaches for QQ-SPM queries:

- **Experiment 1**: A small dataset (Dataset 1) and simple search patterns were used to compare the performance of one of the proposed solutions, specifically QQESPM-Quadtree, against the baseline approach for QQ-SPM queries (ESPM+TV).

- **Experiment 2**: A larger dataset (Dataset 2) and more complex search patterns were employed to compare the performance of the three proposed solutions for QQ-SPM queries: QQESPM-Quadtree, QQESPM-Elastic, and QQESPM-SQL.

5.3 Search Patterns

The search patterns for Experiment 1 (comparing QQESPM-Quadtree with the baseline approach ESPM+TV) consist of 64 randomly generated spatial patterns for the queries. For Experiment 2 (comparing the three proposed approaches), 128 search patterns were generated. These patterns include the most frequent keywords in the dataset, prioritizing queries that yield a higher number of results to simulate real-world search scenarios. Specifically, the keywords in the search patterns were selected from those with a frequency greater than 30 in the datasets. A total of 100 frequent keywords were used to create the search patterns for Experiment 1, while 186 keywords were employed for Experiment 2. Figure 5.3 displays the top 25 most frequent keywords in Dataset 2 along with their respective frequencies. Notably, the three most frequent keywords in the larger dataset are "residential", "bicycle_parking", and "apartments", reflecting common characteristics of a residential area in London, UK.

Eight distinct graph architectures were used to generate the spatial patterns, as illustrated in Figure 5.4. Four different qualitative probabilities were considered, representing the likelihood of an edge having a qualitative topological requirement: $1, 1/2, 1/3$, and $1/4$. An equal number of spatial patterns were generated for each of these qualitative probabilities and for each search graph architecture. The search patterns were created by varying the keywords for the vertices, as well as the distance, topological relations, and exclusion constraints for the edges of the spatial pattern graph.

To define the distance constraints, the minimum distances l_{ij} were randomly selected between 0 and 1000 meters using a uniform distribution. Additionally, the maximum distance constraints were set between $l_{ij} + 200$ and $l_{ij} + 2000$ meters, ensuring a maximum search radius of 3km between the queried POIs. These values were chosen to represent typical POI search scenarios within a single city, where the user seeks to find nearby POIs, ideally within the same neighborhood or district.

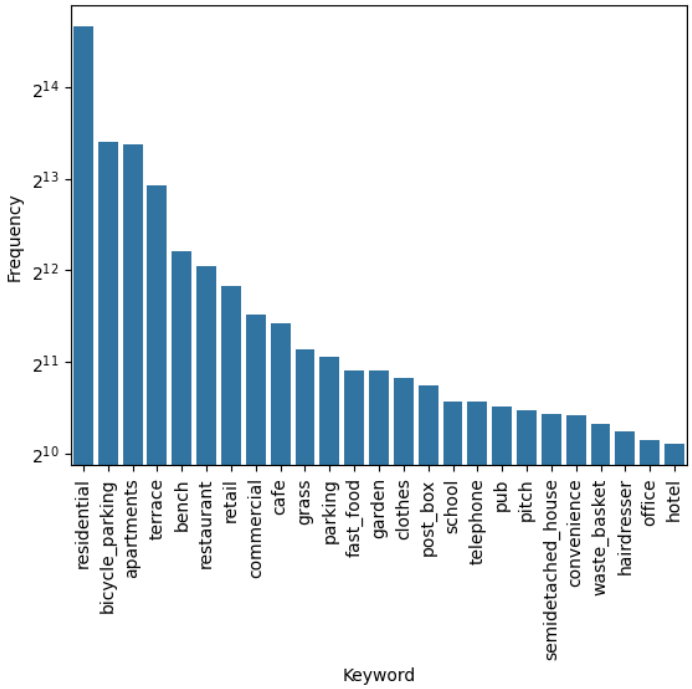

Figure 5.3: Top-25 keywords with highest frequencies in the dataset

For qualitative edges, the topological constraints were randomly selected from "contains", "within", "intersects" and "disjoint" using a uniform distribution. Figure 5.5 illustrates an example of a spatial search pattern generated for the third graph architecture shown in Figure 5.4. This specific search pattern includes the keywords "books", "dentist", and "clinic," the topological constraint "within" and additional distance constraints.

5.4 Experimental Setup

Spatial searches were performed on the London POI datasets using the baseline approach ESPM+TV and the three proposed methods for solving QQ-SPM queries: QQESPM-Quadtree, QQESPM-Elastic, and QQESPM-SQL, detailed in Chapter 4. The primary objectives of these experiments are to evaluate and compare the performance of each method in terms of execution time and memory efficiency. The queries varied in dataset size and search parameters.

To evaluate the performance and scalability of the search approaches (ESPM+TV, QQESPM-Quadtree, QQESPM-Elastic, and QQESPM-SQL), smaller subsets were extracted

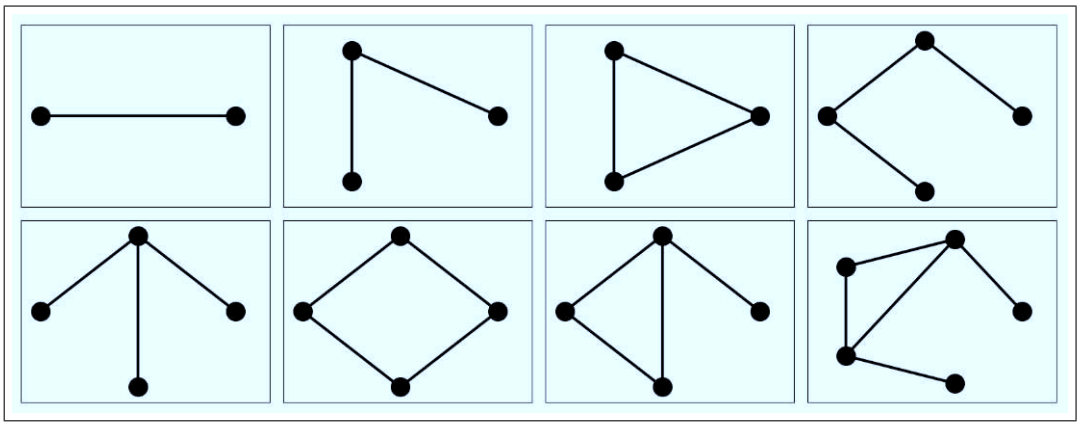

Figure 5.4: Graph architectures of the search patterns

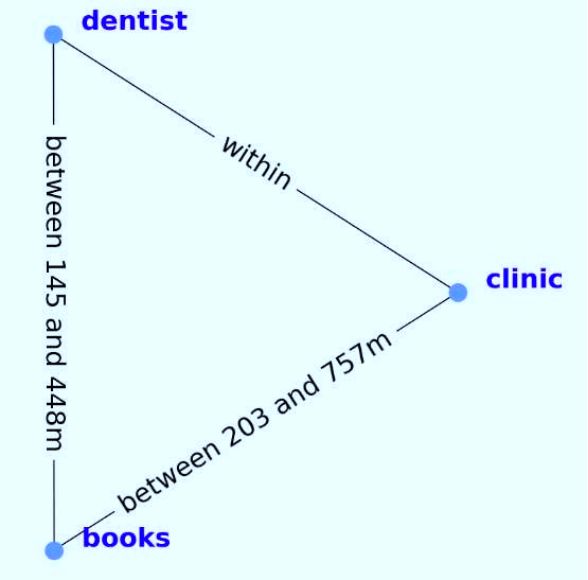

Figure 5.5: Example of generated search pattern

from the full datasets. The analysis focuses on how each method performs as the dataset size increases. Five subsets of varying sizes were created, each representing a different percentage of the total POIs: 20%, 40%, 60%, 80%, and 100%. The largest subset includes the entire dataset, randomly reordered. The sizes of these subsets are detailed in Table 5.3.

Fractional Dataset	# POIs in subsets of Dataset 1	# POIs in subsets of Dataset 2
20%	7,600	25,595
40%	15,200	51,190
60%	22,800	76,785
80%	30,400	102,380
100%	38,000	127,975

Table 5.3: Sizes of the fractional datasets

Experiment 1 involved 640 query executions, utilizing the 64 search patterns across each of the five fractions of Dataset 1 for both the QQESPM-Quadtree and ESPM+TV approaches. In Experiment 2, a total of 5,760 query executions were performed, applying 128 search patterns for three repetitions of each QQESPM solution approach across the five fractions of Dataset 2. For each search configuration, query execution time and memory usage were averaged over the three identical executions. These experiments were conducted on an Ubuntu OS machine with an Intel Core i7-12700F CPU operating at 4.90 GHz and 64 GB of RAM.

5.5 Results

In Experiment 1, the average execution times were 0.09s for QQESPM-Quadtree and 6.14s for ESPM+TV. The maximum execution times recorded were 5.24s for QQESPM-Quadtree and 717.89s for ESPM+TV. Figure 5.6 illustrates the performance of QQESPM-Quadtree and ESPM+TV across various dataset sizes. Notably, ESPM+TV exhibits exponentially worse performance than QQESPM-Quadtree, both in terms of average and maximum execution times.

The limited scalability of ESPM+TV rendered it unsuitable for larger datasets. Consequently, Experiment 2 excludes the ESPM+TV approach, as it involves larger dataset sizes and more complex search patterns. Instead, Experiment 2 evaluates and compares the

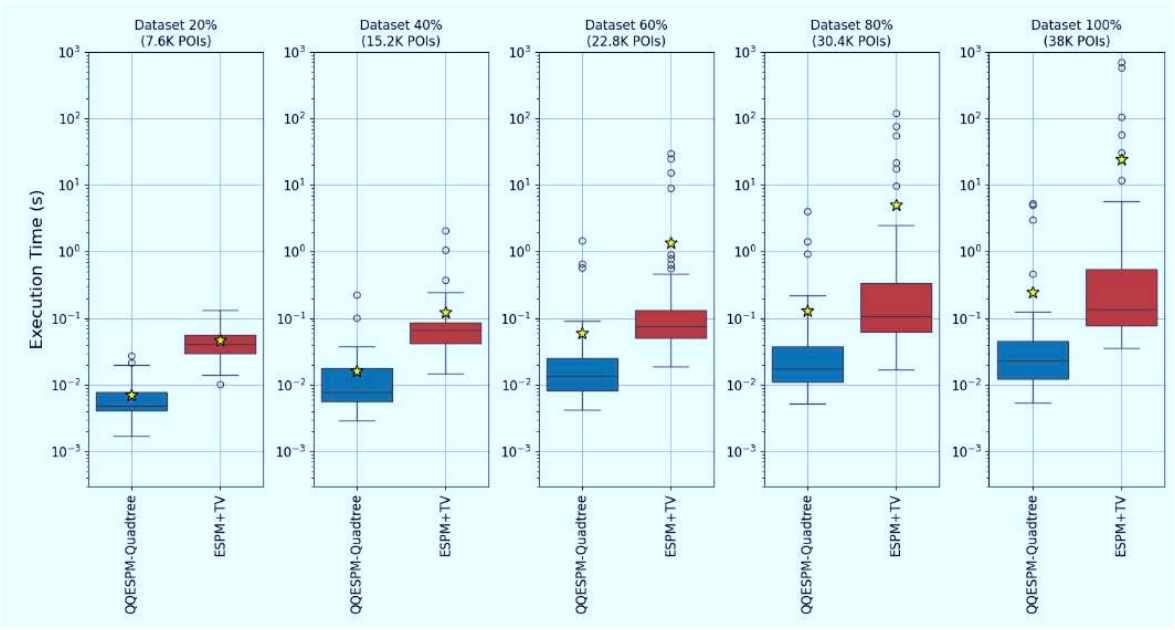

Figure 5.6: Statistics of execution time by dataset size for QQESPM-Quadtree and ESPM+TV approaches on subsets of Dataset 1 (averages as yellow star points)

performance of the three proposed approachesQQESPM-Quadtree, QQESPM-Elastic, and QQESPM-SQL. This experiment utilizes 128 spatial patterns applied to Dataset 2 and its subsets, which contain up to $127,975$ geo-textual objects. This dataset size is over three times larger than the maximum size manageable by ESPM+TV, which barely handled up to $38,000$ POIs.

In Experiment 2, the average execution times for all queries were 2.71 seconds for QQESPM-Quadtree, 2.09 seconds for QQESPM-Elastic, and 0.39 seconds for QQESPM-SQL. The maximum execution times recorded were 505.84 seconds for QQESPM-Quadtree, 444.34 seconds for QQESPM-Elastic, and 9.19 seconds for QQESPM-SQL. QQESPM-Quadtree and QQESPM-Elastic exhibited suboptimal performance on a minority queries, which significantly influenced both their average and maximum execution times. Interestingly, the median execution times were 0.01 seconds for QQESPM-Quadtree, 0.13 seconds for QQESPM-Elastic, and 0.32 seconds for QQESPM-SQL. This indicates that while QQESPM-Quadtree and QQESPM-Elastic generally performed faster, a few costly queries disproportionately affected their average performance.

The equivalence of the three QQ-SPM solution approaches were primarily verified by

the total number of solutions (results) each QQESPM library returned for identical queries, across all $5,760$ query executions. Notably, the maximum percentage divergence in total results between different libraries for identical search patterns was only 0.67%. This finding underscores that all three libraries perform the same search task equivalently and yield nearly identical results. The slight divergence observed may be attributed to distinct numeric approximations employed by each QQESPM library during query processing.

The scalability analysis measured query execution times for various dataset sizes to evaluate how the performance of each proposed solution is affected as the dataset size increases. The collected data from query executions were categorized by the size of the dataset against which the queries were performed.

Figure 5.7 summarizes the statistics of query elapsed times for each library as the dataset size increases. Notably, QQESPM-Quadtree consistently outperforms the other libraries in terms of median execution time, with this advantage growing as dataset sizes increase. A deeper analysis reveals that the 90th percentile of execution times also favors QQESPM-Quadtree, indicating that the majority of queries are more efficiently processed by the QQESPM-Quadtree solution.

The yellow stars in Figure 5.7 indicate the average execution times. Notably, the QQESPM-Quadtree and QQESPM-Elastic solutions experienced a significant decline in performance regarding average execution times as the dataset size increased. A similar trend is observed for the maximum query execution time. This behavior occurs because, although these libraries solve most queries in a fraction of a second, a minority of executions incur substantial costs, leading to poor performance in both average and maximum execution times. In summary, the QQESPM-SQL solution demonstrates better scalability for maximum and average execution times as the dataset size increases, as indicated by the plots.

The following analysis aims to evaluate the impact of the number of constraints in the search patterns on query execution time across different libraries. Figure 5.8 displays the statistics of query execution time by the number of vertices (keywords) in the search graph, separated by library. Conversely, Figure 5.9 shows the query execution time by number of edges in the search graph. Each edge defines a pair-wise relation between two searched objects and can contain both distance and topological constraints. It is observable that the

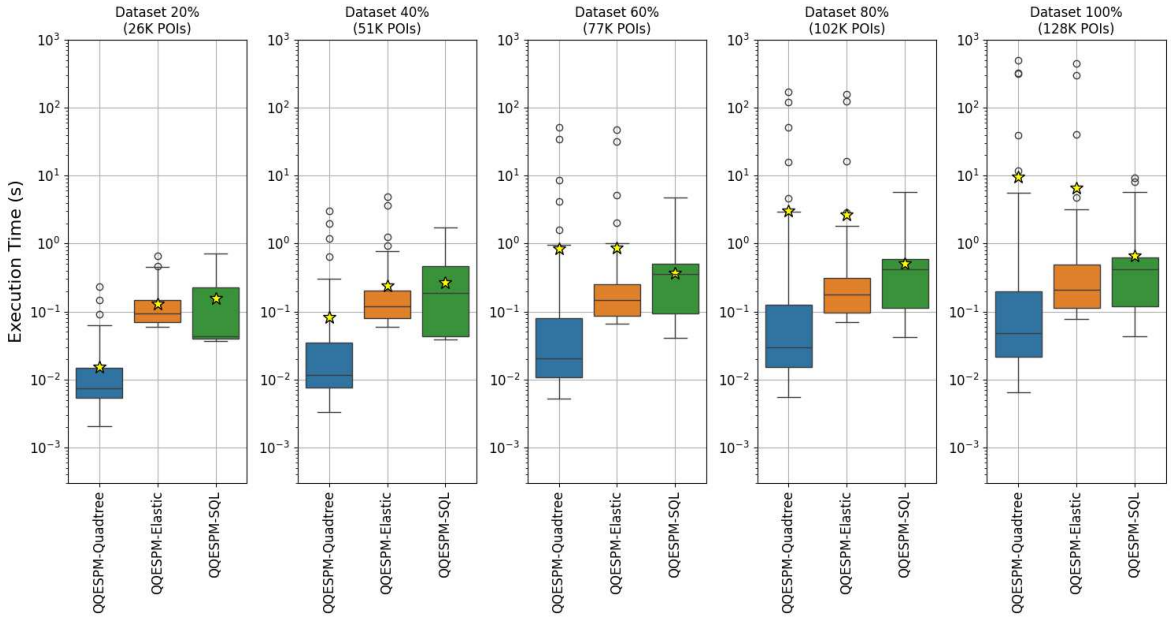

Figure 5.7: Statistics of execution time by dataset size for the three libraries on subsets of Dataset 2 (averages as yellow star points)

number of vertices is strongly related to the number of edges.

Notably, QQESPM-Quadtree performed better for patterns with up to 3 vertices and 2 edges and maintained a better median and 75th percentile execution time across all different numbers of vertices and edges. However, QQESPM-SQL achieved better maximum execution times for search patterns with 4 or more vertices. Interestingly, QQESPM-Elastic demonstrated slightly faster execution times compared to QQESPM-SQL for searches with 5 vertices and 5 edges. This discrepancy may be attributed to suboptimal query plans taken by the implicit JOIN approach in PostgreSQL. Since it is challenging to design a specific JOIN order optimal for all scenarios, the implicit JOIN was employed by default. Nevertheless, these results encourage further exploration of alternative explicit JOIN ordering algorithms to improve the performance of QQESPM-SQL in queries with many keywords.

Several search patterns with 4 vertices and 3 or more edges resulted in significant execution times for the libraries QQESPM-Quadtree and QQESPM-Elastic, notably affecting their average query performance. These prolonged query times stemmed from a costly and non-parallel joining process inherent to these libraries. Certain search patterns accumulated numerous partially constructed solutions during joining loops, thereby

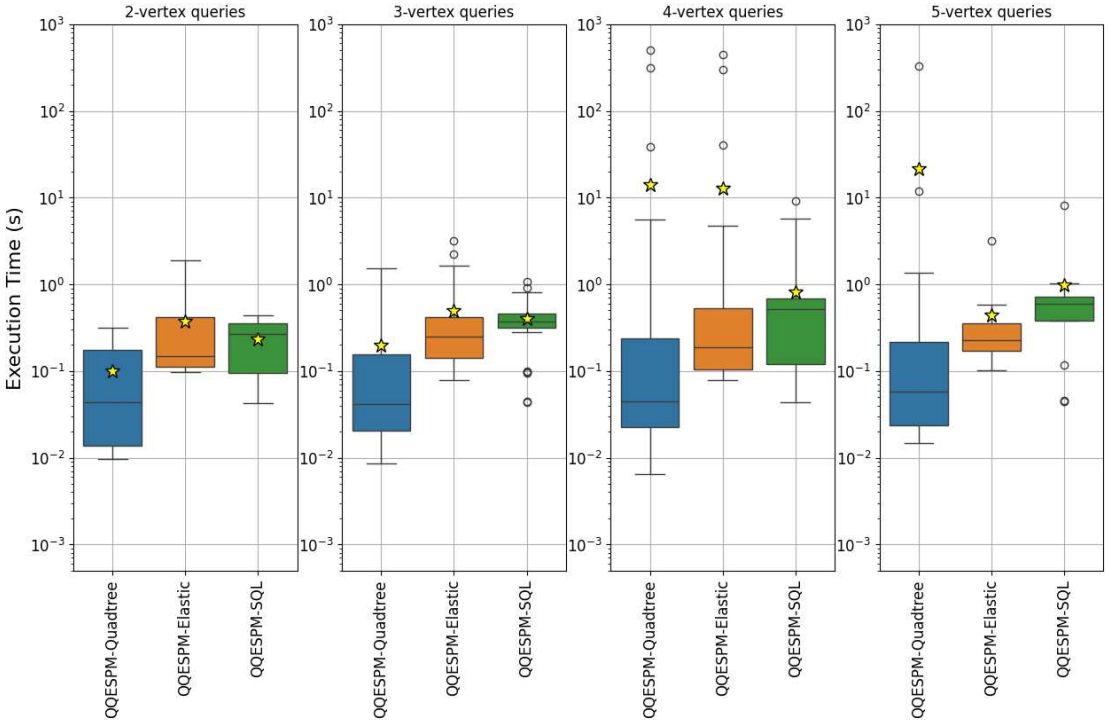

Figure 5.8: Statistics of execution time by number of vertices in the search graph for each library

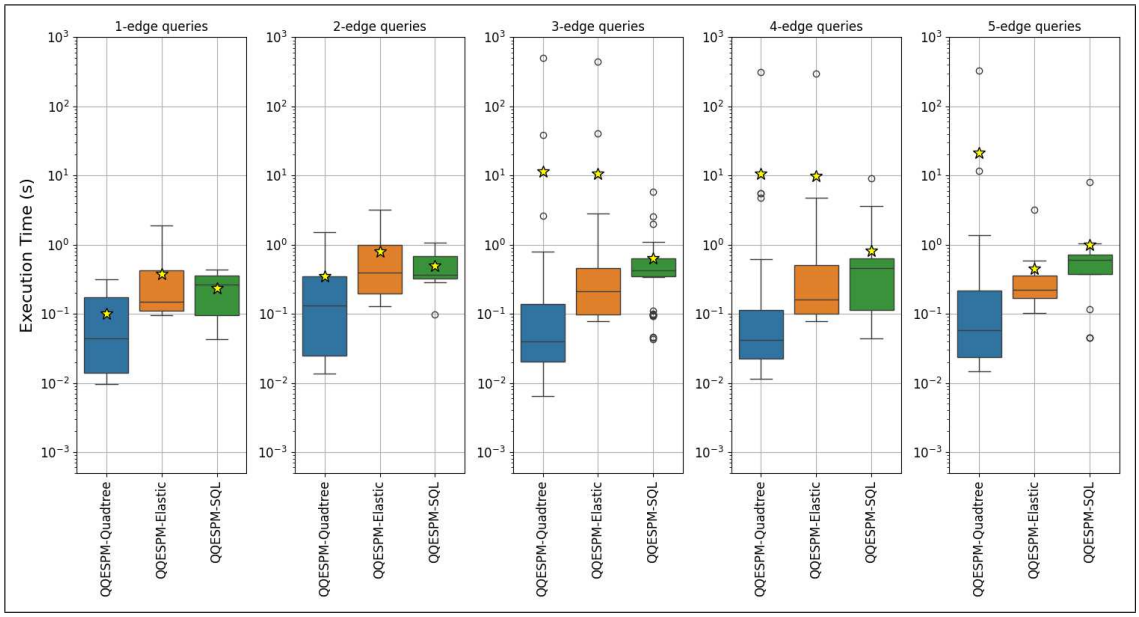

Figure 5.9: Statistics of execution time by number of edges in the search graph for each library

impacting query performance for these two libraries. Such costly queries primarily occurred with patterns yielding a high number of results. However, there are scenarios where both QQESPM-Quadtree and QQESPM-Elastic outperform QQESPM-SQL. This difference may be attributed to their underlying search strategies. QQESPM-Quadtree and QQESPM-Elastic employ imperative search strategies that incorporate various conditions for early query termination. In contrast, QQESPM-SQL utilizes a complete SQL spatial query, representing a more declarative search approach.

Figure 5.10 displays query execution times categorized by the qualitative probability of the search patterns. Patterns with lower qualitative probabilities predominantly feature distance constraints, whereas those with higher qualitative probabilities include more topological constraints. Interestingly, qualitative probability does not show a direct correlation with query execution time. QQESPM-Quadtree exhibits the best average execution time specifically for search patterns with a qualitative probability of 1.0 (fully qualitative patterns), highlighting its efficient handling of purely topological queries. In contrast, QQESPM-SQL consistently achieves relatively fast execution times compared to the occasional spikes observed in other solutions.

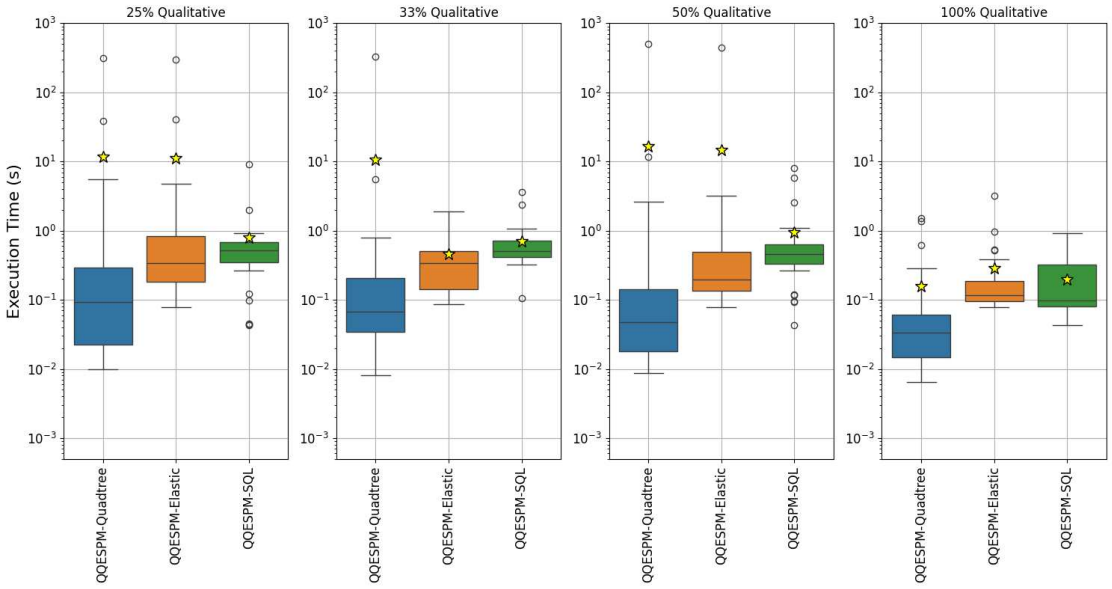

Figure 5.10: Statistics of execution time by qualitative probability in the search graph for each library

Figure 5.11 illustrates the variations in execution times across libraries based on the

number of exclusion constraints in the search patterns. In the context of the search graph, an edge marked with the exclusion signs "$\rightarrow$" or "$\leftarrow$" indicates a unidirectional exclusion constraint between two objects. Conversely, the sign "$\leftrightarrow$" denotes a bidirectional exclusion constraint, equivalent to two separate exclusion constraints. The total count of exclusion constraints within a search graph is determined by summing the occurrences across all its edges, ranging from none (0 exclusion constraints) to a maximum of 6 used in the experiments.

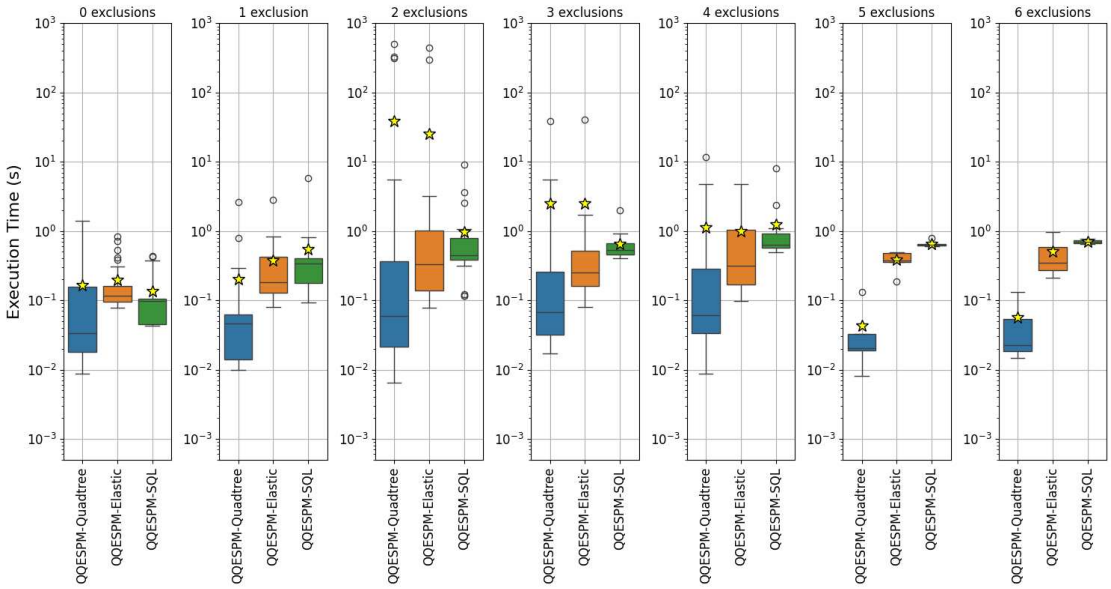

Figure 5.11: Statistics of execution time by total exclusion constraints in the search graph for each library

Figure 5.11 highlights that QQESPM-Quadtree demonstrates superior performance compared to other libraries when handling queries with more than 3 exclusion constraints. Its efficient computation and reuse of computations during the filtering of promising node pairs and object matches contribute significantly to its effectiveness in managing patterns with multiple exclusion constraints. However, there is a notable increase in execution time observed specifically for patterns containing exactly 2 exclusion constraints. This increase in executime time can be explained by the fact that in the sample of search patterns used, there was a high occurrence of queries with 2 exclusions, and also such patterns yielded more results. This caused costly searches for QQESPM-Quadtree and QQESPM-Elastic, stemming from their non-parallel joining process for queries that yield a substantial number

of search results.

In summary, the plots depicting execution time based on different numbers of search constraints indicate that parameters such as “Number of Vertices”, “Number of Edges”, “Qualitative Probability”, and “Number of Exclusion Constraints” do not show a direct correlation with query execution time. Figure 5.12 (A) illustrates the correlations among totals of search constraints, total query solutions (results), dataset size, and query execution time for QQESPM-Quadtree queries. Similarly, Figure 5.12 (B) presents these relationships for QQESPM-Elastic, and Figure 5.12 (C) for QQESPM-SQL query executions.

It is evident that only the variable “Number of Solutions” exhibited a notable correlation with query execution time, although “Dataset Size” presented a weak correlation with the execution time mainly for QQESPM-SQL. This analysis answers Research Question RQ_4. Specifically, the total number of query results demonstrates the strongest correlation with query execution time. This outcome clarifies the absence of a consistent upward or downward trend in execution time across different numbers of search constraints.

5.5.1 Memory Consumption

Memory allocation measurements were conducted for the three libraries under investigation. Figure 5.13 depicts the variation of total allocated memory across all experiments for each library. The white triangles indicate the averages. QQESPM-Elastic and QQESPM-SQL consistently maintain stable memory usage, which suggests that these solutions allocate all the necessary memory from application startup. In contrast, QQESPM-Quadtree shows more variability in memory allocation during queries, yet consistently consumes significantly less memory compared to the other libraries throughout the experiments.

As a Java application, Elasticsearch allocates logical memory (heap) from the system’s physical memory. This allocation should ideally be limited by half of the physical RAM, and limited to 32 GB. Consequently, on a 64 GB RAM machine, the QQESPM-Elastic module consistently consumes around 32 GB of memory. For PostgreSQL, the recomendation for the *shared_buffers* parameter setting, which is 25% of the RAM, was adhered. As a result, the QQESPM-SQL module consistently utilizes approximately 16 GB of memory.

In contrast, the QQESPM-Quadtree module employs a lazy loading strategy, fetching data slices from disk to RAM as needed. Unlike the other approaches, QQESPM-Quadtree

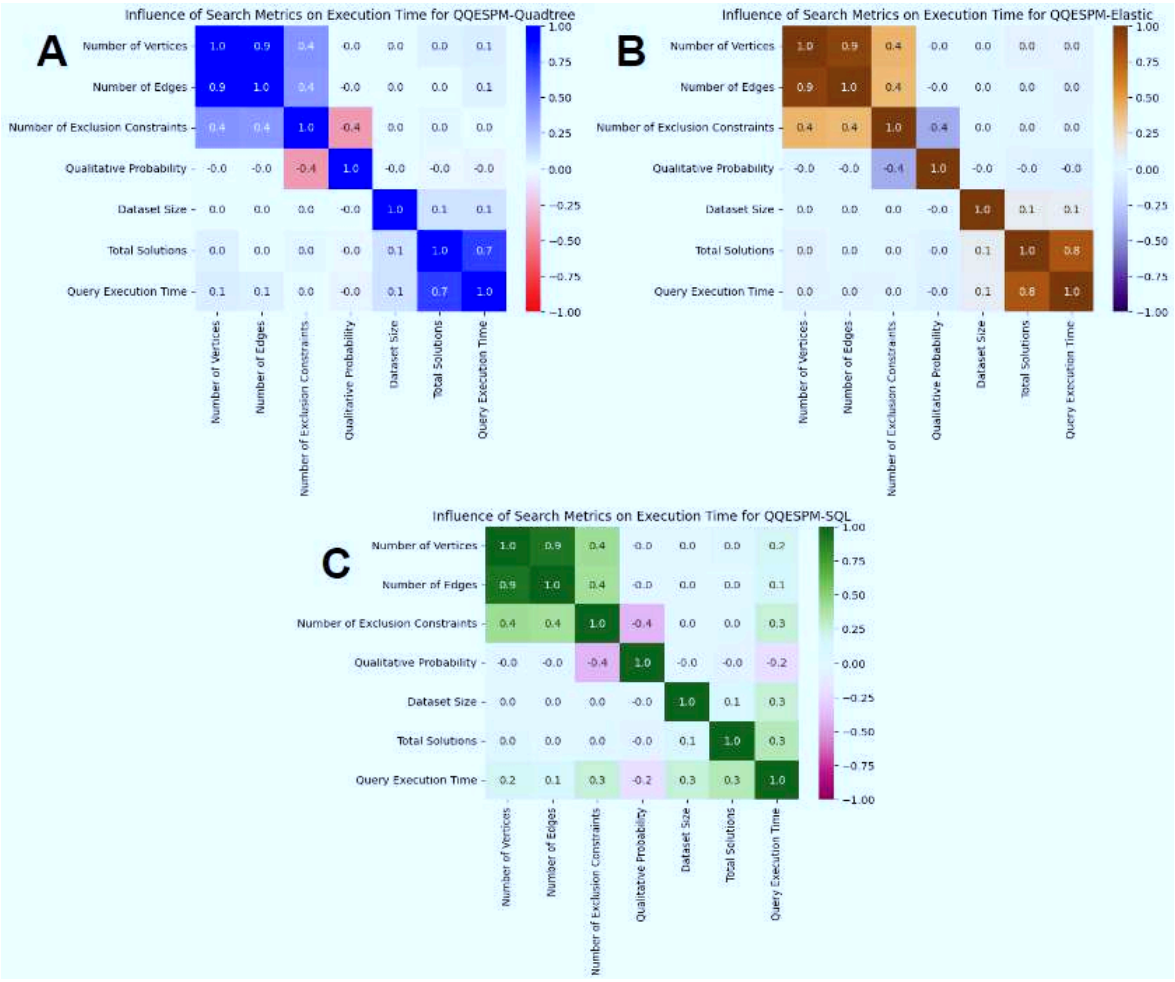

Figure 5.12: Correlation between the numbers of search constraints, dataset size and total solutions with the query execution time for QQESPM-Quadtree (A), QQESPM-Elastic (B) and QQESPM-SQL (C) query executions

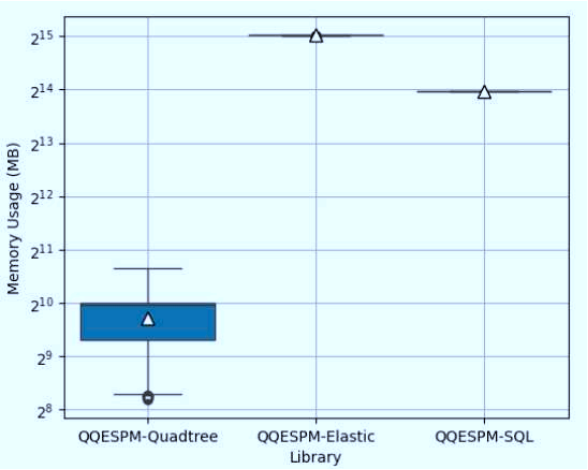

Figure 5.13: Allocated Memory for each solution during queries (averages as white triangle points)

does not maintain any persistent cache or rely on operating system data caching. When the application ends, all cached data is discarded, and subsequent runs of the library start caching anew. While this initial caching may slightly slow down the first queries after QQESPM-Quadtree initialization, it offers a memory usage advantage, consistently using significantly less memory compared to the other approaches.

Although, this observation cannot definitively distinguish QQESPM-Quadtree from the other approaches, since the other libraries can also be configured to use less memory by manually adjusting parameters. The maximum measured memory allocated by QQESPM-Quadtree was approximately 1.6 GB. As queries are submitted, QQESPM-Quadtree progressively loads more objects from disk to RAM, resulting in an almost continuous increase in allocated memory during its usage. However, this growth is constrained by a maximum memory usage prevention mechanism, which is similar to the circuit breaker mechanism from Elasticsearch.

5.6 Practical Implications and Considerations

This study utilized purely algorithmic implementations for conducting experiments. However, in real-world applications, practical strategies based on heuristics can be employed. One effective approach involves caching results from frequent queries and

dynamically updating the index structure. Moreover, to enhance query performance, a initial distance threshold heuristic, such as 500 meters, can be implemented to discard early POIs unlikely to have intersecting geometries (for topological queries). Nevertheless, it is crucial to consider the actual possible range of distances between the centroids of intersecting objects within the dataset. While 500 meters may seem a reasonable distance limit for the central location of neighbor POIs, it may become meaningless for queries to retrieve other types of geo-textual objects. In this sense, such heuristics would heavily depend on the application scenario.

For these reasons the solutions proposed in this master's thesis does not involve any such heuristic. The primary focus of this research was not to integrate heuristics and practical strategies for optimizing everyday queries for specific practical application scenarios. Instead, the emphasis was on providing a pure implementation of the algorithms, avoiding application-specific heuristics. The research aimed to assess the algorithmic solutions themselves, comparing the efficiency of different exact solutions for the QQ-SPM search problem.

Moreover, practical search applications inherently involve query time constraints and maximum limits on the number of search results. These aspects were overlooked in the experiments, as the searches could potentially retrieve exceedingly large result sets, equivalent to the total matches of the search pattern within the dataset. Consequently, this can lead to excessively long response times, rendering it non-optimized for practical real-world scenarios.

Accurate and relevant ranking of results is also a crucial practical aspect in IR systems. Furthermore, effective search systems are expected to return a set of results even if the exact search pattern does not yield any matches. This is achieved through query expansion mechanisms, which allow partial matching to provide users with relevant results, even when an exact match is not found for the search pattern. Such aspects are disregarded in the current implementation, which primarily focuses on the performance comparison of different solution approaches and environment technologies for processing the QQ-SPM queries. Hence, these implementations would necessitate minimal adaptation to meet the requirements of real search scenarios.

Another crucial implementation decision for a practical search system of QQ-SPM

queries is the choice of buffer size applied to the polygonal boundaries of the POIs. In this work, a standard buffer of 10% of the average side length was applied to the boundary of each POI in the dataset. However, this estimation was not thoroughly informed and was chosen as a default without extensive consideration. In practice, measurements of false positives and false negatives for real intersecting geometries should be conducted to determine the most appropriate buffer size for a specific search application scenario. Larger buffers may increase false positives of intersecting geometries, while smaller buffers may lead to false negatives, failing to retrieve important intersecting geometries in the queries.

Furthermore, search procedures that allow the retrieval of semantically similar keywords, rather than requiring exact matches, and slight acceptance thresholds for satisfying the distance and topological constraints, are important to avoid missing relevant results. In this context, partial matches of the search pattern are as important as exact matches in real IR systems.

5.7 Final Considerations

This chapter presented the methodology and results for a series of performance experiments conducted to compare the three proposed solutions for the QQ-SPM search problem. The findings highlight the robustness of the QQESPM-SQL solution, which effectively manages resource-intensive queries while demonstrating scalability and consistency. For large datasets, QQESPM-SQL remained the most suitable option. Conversely, QQESPM-Quadtree and QQESPM-Elastic outperformed in terms of execution time for over 90% of the queries, although exhibiting suboptimal performance in a minority of cases, resulting in higher average execution times. Conversely, search patterns with large numbers of exclusion or topological constraints presented few performances advantages for QQESPM-Quadtree and QQESPM-Elastic.

Chapter 6

Conclusion and Future Directions

The literature extensively explores querying geo-textual data using keywords and spatial proximity, with various studies proposing algorithms and indexing methods to handle spatio-textual queries efficiently. While most research focuses on basic item-wise queries, recent advancements have introduced methods for group queries that retrieve multiple spatio-textual objects with specific keyword and spatial restrictions. However, most research does not address complex spatial searches, particularly those involving qualitative topological constraints. This master's thesis fills this gap by investigating the QQ-SPM query, a flexible and generic type of spatial pattern search that generalizes the SPM query. The QQ-SPM query efficiently processes distance and topological constraints and is particularly useful for searches requirements like finding residential buildings in central London within 3 km of a gym-equipped shopping mall, while avoiding proximity to cemeteries.

The conducted investigation formalized the QQ-SPM and established two theorems that led to the development of the QQESPM-Quadtree algorithm that uses the IL-Quadtree index. Additionally, the QQESPM-Elastic solution was designed to efficiently leverage native spatial operations from Elasticsearch for QQ-SPM queries. The third achieved solution involved creating a pipeline to convert QQ-SPM search requirements into spatial SQL queries for use in relational databases. The research contributed the complete design of a framework for handling QQ-SPM queries across various geospatial technologies, including an ad hoc solution independent of GIS backends (QQESPM-Quadtree). The inclusion of these strategies within spatial databases incur optimized query plans for QQ-SPM queries. The proposed solutions may be embedded in backend APIs for query technologies like

PostgreSQL and Elasticsearch, and facilitate customized spatial pattern searches in web-based POI search applications like Google Maps.

A series of performance experiments were conducted to measure the execution time of three proposed solutions, along with a baseline approach, for solving QQ-SPM queries. The baseline approach, termed ESPM+TV, involves using the ESPM algorithm from previous research to retrieve groups of objects that satisfy the distance constraints of a QQ-SPM search pattern. The results are then filtered based on topological constraints through an exhaustive search. The investigation primarily focused on the scalability of these approaches as the target dataset for the search grew. Additionally, the research examined various factors contributing to the computational cost of the search, such as the impact of the number of exclusion constraints on query execution time.

The analysis revealed that the baseline approach, ESPM+TV, experienced a sharp exponential increase in execution time with linear increases in dataset size. Consequently, this approach proved infeasible for datasets exceeding 38,000 geo-textual objects, as execution times surpassed 700 seconds. In contrast, the three proposed solutions demonstrated near-linear behavior in execution time relative to dataset size. The three proposed solutions were further evaluated by executing QQ-SPM queries on datasets containing up to 128,000 geo-textual objects.

The scalability assessment results indicate that, for queries on datasets with fewer than 51,000 objects, the average performance of QQESPM-Quadtree and QQESPM-Elastic surpassed that of QQESPM-SQL. The ad hoc solution, QQESPM-Quadtree, also executed queries remarkably quickly, within thousandths of a second, and exhibited better median and 75th percentile execution times compared to the other solutions. However, overall, the QQESPM-SQL solution proved to be much more scalable, demonstrating superior performance in terms of both average and maximum query execution times. Additionally, QQESPM-SQL showed smaller increases in query time when handling larger datasets.

The categorization of queries based on the number of vertices or edges in the search pattern graph revealed that QQESPM-SQL is generally faster for queries with more than 3 vertices and 2 edges. A costly joining procedure in the QQESPM-Quadtree and QQESPM-Elastic solutions incurred significant overhead in queries that returned large numbers of results. This directly impacted the average performance of these solutions, particularly for

queries with more than 3 vertices or 2 edges, as such queries require more join operations.

The experiments also evaluated the execution time of the QQ-SPM search solutions relative to the number of exclusion constraints in the search pattern. The analysis revealed that the performance of QQESPM-SQL is directly impacted by the number of exclusion constraints in the query. In contrast, QQESPM-Quadtree and QQESPM-Elastic handled queries with a large number of exclusion constraints more effectively. Notably, QQESPM-Quadtree excelled in performance for queries with more than 4 exclusion constraints, delivering query execution times up to 10 times faster than the other solutions.

In summary, QQESPM-SQL has proven more efficient for handling QQ-SPM queries on large datasets. In contrast, QQESPM-Quadtree and QQESPM-Elastic performed better for queries on smaller datasets, queries with fewer keywords, or those involving more topological or exclusion constraints. QQESPM-SQL achieved an overall average query execution time approximately six times faster than the other two solutions, and its maximum execution time was up to 50 times faster. Additionally, the QQESPM-SQL library remained stable and robust even for resource-intensive queries that returned millions of results. These outcomes are largely attributable to a non-parallel and costly joining routine in QQESPM-Quadtree and QQESPM-Elastic. Future implementations could potentially enhance performance by introducing parallelization for this phase, thereby improving the efficiency of costly queries in these two solutions.

6.1 Limitations

One limitation of the proposed approach is its inability to semantically interpret query keywords, resulting in filtering to exact matches only. However, this limitation was intended to assess exact solutions for the QQ-SPM query, efficiently returning all exact matches of the search pattern. In this context, a few adjustments can be made to the open-source implementation to include partial matches, allowing non-exact search results with approximate keywords and distances.

Moreover, the sample of spatial patterns investigated may not adequately represent typical real-world search scenarios. Further investigations are necessary to assess the performance of the search approaches in practical settings. Additionally, additional

requirements could be incorporated to tailor current implementations to practical scenarios. These may include result ranking, setting a theoretical time limit for query execution, and retrieving a subset of matches when the full result set is excessively large and costly.

Another limitation is the gap between the types of constraints supported by QQ-SPM queries and real-world natural language search requirements, which often involve complex qualitative spatial predicates like "Near", "Between" and "Connected". Previous research has explored experimental methods based on geometric conventions and volunteered geographic information to interpret these expressions. However, there has been no integration of these approaches with the QQ-SPM query framework. Therefore, the proposed query type assumes that each qualitative spatial requirement can be formally and unambiguously represented computationally. Additionally, this analysis considered only four fundamental topological relationships (Contains, Within, Intersects, and Disjoint) between the queried objects. Despite these limitations, the direct employment of QQ-SPM queries into end-to-end applications is possible, as shown in Appendix A, by restricting the possible qualitative topological requirements to a predefined set.

6.2 Future Directions

In advancing this research, several avenues for future development can be explored. The following examples outline potential areas for further investigation.

- **Experimenting different spatial indexing**: Exploring alternative spatial indexes beyond the IL-Quadtree, such as IR-Trees, S2, and H3, along with the development of distinct algorithms tailored to leverage these indexes effectively. Evaluating multiple indexes and their corresponding algorithms could provide insights into optimal strategies for addressing QQ-SPM queries, considering their diverse constraints. Comparisons among these approaches can offer guidelines on their respective strengths and optimal contexts of applicability.

- **Designing highly-scalable solutions for big data**: The proposed search approaches do not explore distributed computing or big data principles and lack optimized designs for parallel and distributed processing. Future research could adapt the

proposed algorithms and implementations to leverage such capabilities for queries. Additionally, there is a need for guidelines on transforming these solutions into efficient and scalable distributed computing solutions. For instance, when dealing with large datasets distributed across multiple cluster machines, a central procedure could coordinate the distributed and parallel processing of complex QQ-SPM queries. This would involve aggregating search results from different clusters in subsequent stages of the search process.

- **Enriching search with semantics, partial matching and results ranking**: Another future direction involves integrating NLP and LLMs to improve the relevance of search results, enabling partial matching with semantically equivalent keywords and approximate spatial configurations among the retrieved search results. Moreover, incorporating ranking mechanisms, for example, prioritizing highly-rated POIs, is crucial for meeting real-world search needs in IR systems. Experimental validation of several tolerance search parameters to enhance results relevance could be conducted by volunteers to assess the accuracy and relevance of query results. Additionally, there is a growing interest in spatial searches based on natural language spatio-textual descriptions and qualitative requirements. Future research could explore automated methods for translating natural language specifications into appropriate QQ-SPM queries, leveraging a versatile spatial pattern search approach that accommodates both quantitative and qualitative constraints expressed in natural language requirements. Finally, the retrieval of heterogeneous types of geo-textual objects can be investigated, for example, for enabling the efficient processing of search scenarios such as finding POIs located along the same street (POIs and streets) or POIs closely located to a city's commercial district (POIs and administrative areas).

Bibliography

[1] Niloofar Aflaki, Kristin Stock, Christopher B Jones, Hans Guesgen, Jeremy Morley, and Yukio Fukuzawa. What do you mean you're in trafalgar square? comparing distance thresholds for geospatial prepositions. In *15th International Conference on Spatial Information Theory (COSIT 2022)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2022.

[2] Ritesh Ahuja, Nikos Armenatzoglou, Dimitris Papadias, and George J Fakas. Geo-social keyword search. In *Advances in Spatial and Temporal Databases: 14th International Symposium, SSTD 2015, Hong Kong, China, August 26-28, 2015. Proceedings 14*, pages 431–450. Springer, 2015.

[3] Walid G Aref and Ihab F Ilyas. Sp-gist: An extensible database index for supporting space partitioning trees. *Journal of Intelligent Information Systems*, 17:215–240, 2001.

[4] W.G. Aref and I.F. Ilyas. An extensible index for spatial databases. In *Proceedings Thirteenth International Conference on Scientific and Statistical Database Management. SSDBM 2001*, pages 49–58, 2001.

[5] Pasquale Balsebre, Weiming Huang, and Gao Cong. Lamp: A language model on the map. *arXiv preprint arXiv:2403.09059*, 2024.

[6] Felice L Bedford. Perceptual and cognitive spatial learning. *Journal of experimental psychology: Human perception and performance*, 19(3):517, 1993.

[7] Shaik Abdul Nusrath Begum and KP Supreethi. A survey on spatial indexing. *Journal of Web Development and Web Designing*, 3(1), 2018.

[8] Isabelle Bloch, Olivier Colliot, and Roberto M Cesar. On the ternary spatial relation"" between"". *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 36(2):312–327, 2006.

[9] Panagiotis Bouros, Shen Ge, and Nikos Mamoulis. Spatio-textual similarity joins. *Proceedings of the VLDB Endowment*, 6(1):1–12, 2012.

[10] Panagiotis Bouros and Nikos Mamoulis. Spatial joins: what's next? *SIGSPATIAL Special*, 11(1):1321, aug 2019.

[11] Tom Bruns and Max Egenhofer. Similarity of spatial scenes. In *Seventh international symposium on spatial data handling*, pages 31–42. Delft, The Netherlands, 1996.

[12] Sabina Buczkowska and Matthieu de Lapparent. Location choices of newly created establishments: Spatial patterns at the aggregate level. *Regional Science and Urban Economics*, 48:68–81, 2014.

[13] Xin Cao, Gao Cong, Tao Guo, Christian S Jensen, and Beng Chin Ooi. Efficient processing of spatial group keyword queries. *ACM Transactions on Database Systems (TODS)*, 40(2):1–48, 2015.

[14] Xin Cao, Gao Cong, Tao Guo, Christian S. Jensen, and Beng Chin Ooi. Efficient processing of spatial group keyword queries. *ACM Trans. Database Syst.*, 40(2), jun 2015.

[15] Xin Cao, Gao Cong, and Christian S. Jensen. Retrieving top-k prestige-based relevant spatial web objects. *Proc. VLDB Endow.*, 3(12):373384, sep 2010.

[16] Xin Cao, Gao Cong, Christian S Jensen, and Beng Chin Ooi. Collective spatial keyword querying. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, pages 373–384, 2011.

[17] Anderson C. Carniel, Ricardo R. Ciferri, and Cristina D.A. Ciferri. Festival: A versatile framework for conducting experimental evaluations of spatial indices. *MethodsX*, 7:100695, 2020.

[18] Anderson Chaves Carniel. Defining and designing spatial queries: the role of spatial relationships. *Geo-spatial Information Science*, 0(0):1–25, 2023.

[19] Ariel Cary, Ouri Wolfson, and Naphtali Rishe. Efficient and scalable method for processing top-k spatial boolean queries. In *International Conference on Scientific and Statistical Database Management*, pages 87–95. Springer, 2010.

[20] Harry Kai-Ho Chan, Cheng Long, and Raymond Chi-Wing Wong. Inherent-cost aware collective spatial keyword queries. In *International Symposium on Spatial and Temporal Databases*, pages 357–375. Springer, 2017.

[21] Harry Kai-Ho Chan, Cheng Long, and Raymond Chi-Wing Wong. On generalizing collective spatial keyword queries. *IEEE Transactions on Knowledge and Data Engineering*, 30(9):1712–1726, 2018.

[22] Anderson Chaves Carniel. Defining and designing spatial queries: the role of spatial relationships. *Geo-spatial Information Science*, pages 1–25, 2023.

[23] Gang Chen, Jingwen Zhao, Yunjun Gao, Lei Chen, and Rui Chen. Time-aware boolean spatial keyword queries. *IEEE Transactions on Knowledge and Data Engineering*, 29(11):2601–2614, 2017.

[24] Hongmei Chen, Yixiang Fang, Ying Zhang, Wenjie Zhang, and Lizhen Wang. Espm: Efficient spatial pattern matching. *IEEE Transactions on Knowledge and Data Engineering*, 32(6):1227–1233, 2019.

[25] Lisi Chen, Gao Cong, Christian S Jensen, and Dingming Wu. Spatial keyword query processing: An experimental evaluation. *Proceedings of the VLDB Endowment*, 6(3):217–228, 2013.

[26] Lisi Chen, Shuo Shang, Chengcheng Yang, and Jing Li. Spatial keyword search: a survey. *GeoInformatica*, 24:85–106, 2020.

[27] Yue Chen, Kaiyu Feng, Gao Cong, and Han Mao Kiah. Example-based spatial pattern matching. *Proceedings of the VLDB Endowment*, 15(11):2572–2584, 2022.

[28] Zhida Chen, Lisi Chen, Gao Cong, and Christian S Jensen. Location-and keyword-based querying of geo-textual data: a survey. *The VLDB Journal*, 30:603–640, 2021.

[29] Dong-Wan Choi, Jian Pei, and Xuemin Lin. Finding the minimum spatial keyword cover. In *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*, pages 685–696, 2016.

[30] Dong-Wan Choi, Jian Pei, and Xuemin Lin. On spatial keyword covering. *Knowledge and Information Systems*, 62(7):2577–2612, 2020.

[31] Maria Christoforaki, Jinru He, Constantinos Dimopoulos, Alexander Markowetz, and Torsten Suel. Text vs. space: efficient geo-search query processing. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 423–432, 2011.

[32] Eliseo Clementini and Paolino Di Felice. A comparison of methods for representing topological relationships. *Information sciences-applications*, 3(3):149–178, 1995.

[33] Eliseo Clementini, Paolino Di Felice, and Peter Van Oosterom. A small set of formal topological relationships suitable for end-user interaction. In *International symposium on spatial databases*, pages 277–295. Springer, 1993.

[34] Eliseo Clementini, Jayant Sharma, and Max J Egenhofer. Modelling topological spatial relations: Strategies for query processing. *Computers & graphics*, 18(6):815–822, 1994.

[35] Anthony G Cohn, Brandon Bennett, John Gooday, and Nicholas Mark Gotts. Qualitative spatial representation and reasoning with the region connection calculus. *geoinformatica*, 1:275–316, 1997.

[36] Anthony G Cohn and Jochen Renz. Qualitative spatial representation and reasoning. *Foundations of Artificial Intelligence*, 3:551–596, 2008.

[37] Gao Cong and Christian S. Jensen. *Spatio-textual Data*, pages 1580–1587. Springer International Publishing, Cham, 2019.

[38] Gao Cong, Christian S Jensen, and Dingming Wu. Efficient retrieval of the top-k most relevant spatial web objects. *Proceedings of the VLDB Endowment*, 2(1):337–348, 2009.

[39] Rone Ilídio Da Silva, Daniel Fernandes Macedo, and José Marcos S Nogueira. Spatial query processing in wireless sensor networks–a survey. *Information Fusion*, 15:32–43, 2014.

[40] João Paulo Dias de Almeida and João B Rocha-Junior. Top-k spatial keyword preference query. *Journal of Information and Data Management*, 6(3):162–162, 2015.

[41] Ian De Felipe, Vagelis Hristidis, and Naphtali Rishe. Keyword search on spatial databases. In *2008 IEEE 24th International conference on data engineering*, pages 656–665. IEEE, 2008.

[42] Ian De Felipe, Vagelis Hristidis, and Naphtali Rishe. Keyword search on spatial databases. In *2008 IEEE 24th International Conference on Data Engineering*, pages 656–665, 2008.

[43] Ke Deng, Xin Li, Jiaheng Lu, and Xiaofang Zhou. Best keyword cover search. *IEEE Transactions on Knowledge and Data Engineering*, 27(1):61–73, 2015.

[44] Ke Deng, Huanliang Sun, Yu Ge, Xiaofang Zhou, Christian Søndergaard Jensen, et al. Clue-based spatio-textual query. *Proceedings of the VLDB Endowment*, 10(5):529–540, 2017.

[45] Germano B dos Santos, Leonardo JAS Figueiredo, Fabrício A Silva, and Antonio AF Loureiro. Moredata: Enriquecimento semântico para grandes volumes de dados geolocalizados. In *Anais Estendidos do XXXVIII Simpósio Brasileiro de Bancos de Dados*, pages 126–131. SBC, 2023.

[46] Max Egenhofer. A mathematical framework for the definition of topological relations. In *Proc. the fourth international symposium on spatial data handing*, pages 803–813, 1990.

[47] Max J Egenhofer and John Herring. Categorizing binary topological relations between regions, lines, and points in geographic databases. *The*, 9(94-1):76, 1990.

[48] M.Y. Eltabakh, R. Eltarras, and W.G. Aref. Space-partitioning trees in postgresql: Realization and performance. In *22nd International Conference on Data Engineering (ICDE'06)*, pages 100–100, 2006.

[49] Ju Fan, Guoliang Li, Lizhu Zhou, Shanshan Chen, and Jun Hu. Seal: Spatio-textual similarity search. *arXiv preprint arXiv:1205.6694*, 2012.

[50] Yixiang Fang, Reynold Cheng, Gao Cong, Nikos Mamoulis, and Yun Li. On spatial pattern matching. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, pages 293–304. IEEE, 2018.

[51] Yixiang Fang, Reynold Cheng, Jikun Wang, Lukito Budiman, Gao Cong, and Nikos Mamoulis. Spacekey: Exploring patterns in spatial databases. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, pages 1577–1580, 2018.

[52] Yixiang Fang, Yun Li, Reynold Cheng, Nikos Mamoulis, and Gao Cong. Evaluating pattern matching queries for spatial databases. *The VLDB Journal*, 28:649–673, 2019.

[53] Maria Engracinda dos Santos Ferreira and Luciene Stamato Delazari. The use of spatial terms near,very near,next to,side by side and nearby in the descriptions of spatial configurations. *Boletim de Ciências Geodésicas*, 25:e2019008, 2019.

[54] Athanasios Fevgas and Panayiotis Bozanis. Lb-grid: An ssd efficient grid file. *Data & Knowledge Engineering*, 121:18–41, 2019.

[55] Leonardo JAS Figueiredo, Germano B dos Santos, Raissa PPM Souza, Fabrício A Silva, and Thais RM Braga Silva. Moredata: A geospatial data enrichment framework. In *Proceedings of the 29th International Conference on Advances in Geographic Information Systems*, pages 419–422, 2021.

[56] Raphael A Finkel and Jon Louis Bentley. Quad trees a data structure for retrieval on composite keys. *Acta informatica*, 4:1–9, 1974.

[57] Peter F Fisher, Roland Billen, and Eliseo Clementini. Introducing a reasoning system based on ternary projective relations. In *Developments in Spatial Data Handling: 11 th International Symposium on Spatial Data Handling*, pages 381–394. Springer, 2005.

[58] Paolo Fogliaroni. *Qualitative Spatial Configuration Queries Towards Next Generation Access Methods for GIS*. PhD thesis, Universität Bremen, 2012.

[59] Paolo Fogliaroni and Eliseo Clementini. Modeling visibility in 3d space: a qualitative frame of reference. In *3D Geoinformation Science: The Selected Papers of the 3D GeoInfo 2014*, pages 243–258. Springer, 2014.

[60] Paolo Fogliaroni, Jan Oliver Wallgrün, Eliseo Clementini, Francesco Tarquini, and Diedrich Wolter. A qualitative approach to localization and navigation based on visibility information. In *Spatial Information Theory: 9th International Conference, COSIT 2009 Aber Wrach, France, September 21-25, 2009 Proceedings 9*, pages 312–329. Springer, 2009.

[61] Paolo Fogliaroni, Paul Weiser, and Heidelinde Hobel. Qualitative spatial configuration search. *Spatial Cognition & Computation*, 16(4):272–300, 2016.

[62] Christian Freksa. Qualitative spatial reasoning. In *Cognitive and linguistic aspects of geographic space*, pages 361–372. Springer, 1991.

[63] Volker Gaede and Oliver G¨unther. Multidimensional access methods. *ACM Comput. Surv.*, 30(2):170231, jun 1998.

[64] Yunpeng Gao, Yao Wang, and Shengwei Yi. Preference-aware top-k spatio-textual queries. In Shaoxu Song and Yongxin Tong, editors, *Web-Age Information Management*, pages 186–197, Cham, 2016. Springer International Publishing.

[65] Tao Guo, Xin Cao, and Gao Cong. Efficient algorithms for answering the m-closest keywords query. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, SIGMOD '15, page 405418, New York, NY, USA, 2015. Association for Computing Machinery.

[66] Xi Guo and Xiaochun Yang. Direction-aware nearest neighbor query. *IEEE Access*, 7:30285–30301, 2019.

[67] Ying Guo, Lianzhen Zheng, Yuhan Zhang, and Guanfeng Liu. Mcops-spm: Multi-constrained optimized path selection based spatial pattern matching in

social networks. In *Cloud Computing, Smart Grid and Innovative Frontiers in Telecommunications: 9th EAI International Conference, CloudComp 2019, and 4th EAI International Conference, SmartGIFT 2019, Beijing, China, December 4-5, 2019, and December 21-22, 2019 9*, pages 3–19. Springer, 2020.

[68] Antonin Guttman. R-trees: A dynamic index structure for spatial searching. In *Proceedings of the 1984 ACM SIGMOD international conference on Management of data*, pages 47–57, 1984.

[69] Ramaswamy Hariharan, Bijit Hore, Chen Li, and Sharad Mehrotra. Processing spatial-keyword (sk) queries in geographic information retrieval (gir) systems. In *19th International Conference on Scientific and Statistical Database Management (SSDBM 2007)*, pages 16–16, 2007.

[70] Ramón Hermoso, Sergio Ilarri, and Raquel Trillo Lado. Re-coskq: Towards pois recommendation using collective spatial keyword queries. In *RecTour@RecSys*, 2019.

[71] Daniel Hernández, Eliseo Clementini, and Paolino Di Felice. Qualitative distances. In *Spatial Information Theory A Theoretical Basis for GIS: International Conference COSIT'95 Semmering, Austria, September 21–23, 1995 Proceedings 2*, pages 45–57. Springer, 1995.

[72] Huiqi Hu, Guoliang Li, Zhifeng Bao, Jianhua Feng, Yongwei Wu, Zhiguo Gong, and Yaoqiang Xu. Top-k spatio-textual similarity join. *IEEE Transactions on Knowledge and Data Engineering*, 28(2):551–565, 2015.

[73] Edwin H. Jacox and Hanan Samet. Spatial join techniques. *ACM Trans. Database Syst.*, 32(1):7es, mar 2007.

[74] Georgios Kalamatianos, Georgios J Fakas, and Nikos Mamoulis. Proportionality in spatial keyword search. In *Proceedings of the 2021 International Conference on Management of Data*, pages 885–897, 2021.

[75] Ali Khodaei, Cyrus Shahabi, and Chen Li. Hybrid indexing and seamless ranking of spatial and textual features of web documents. In *Database and Expert Systems*

Applications: 21st International Conference, DEXA 2010, Bilbao, Spain, August 30-September 3, 2010, Proceedings, Part I 21, pages 450–466. Springer, 2010.

[76] Ravi K Kothuri and Siva Ravada. Efficient processing of large spatial queries using interior approximations. In *International Symposium on Spatial and Temporal Databases*, pages 404–421. Springer, 2001.

[77] Ravi Kanth V Kothuri, Siva Ravada, and Daniel Abugov. Quadtree and r-tree indexes in oracle spatial: a comparison using gis data. In *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, pages 546–557, 2002.

[78] Taesung Lee, Jin-woo Park, Sanghoon Lee, Seung-won Hwang, Sameh Elnikety, and Yuxiong He. Processing and optimizing main memory spatial-keyword queries. *Proceedings of the VLDB Endowment*, 9(3):132–143, 2015.

[79] Guoliang Li, Jianhua Feng, and Jing Xu. Desks: Direction-aware spatial keyword search. In *2012 IEEE 28th International Conference on Data Engineering*, pages 474–485, 2012.

[80] Miao Li, Lisi Chen, Gao Cong, Yu Gu, and Ge Yu. Efficient processing of location-aware group preference queries. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, page 559568, New York, NY, USA, 2016. Association for Computing Machinery.

[81] Yun Li, Yixiang Fang, Reynold Cheng, and Wenjie Zhang. Spatial pattern matching: A new direction for finding spatial objects. *SIGSPATIAL Special*, 11(1):312, aug 2019.

[82] Zhisheng Li, Ken C.K. Lee, Baihua Zheng, Wang-Chien Lee, Dik Lee, and Xufa Wang. Ir-tree: An efficient index for geographic document search. *IEEE Transactions on Knowledge and Data Engineering*, 23(4):585–599, 2011.

[83] Sitong Liu, Guoliang Li, and Jianhua Feng. A prefix-filter based method for spatio-textual similarity join. *IEEE Transactions on Knowledge and Data Engineering*, 26(10):2354–2367, 2013.

[84] Cheng Long, Raymond Chi-Wing Wong, Ke Wang, and Ada Wai-Chee Fu. Collective spatial keyword queries: a distance owner-driven approach. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, SIGMOD '13, page 689700, New York, NY, USA, 2013. Association for Computing Machinery.

[85] Zhiguo Long, Matt Duckham, Sanjiang Li, and Steven Schockaert. Indexing large geographic datasets with compact qualitative representation. *International Journal of Geographical Information Science*, 30(6):1072–1094, 2016.

[86] Yun Lu, Mingjin Zhang, Shonda Witherspoon, Yelena Yesha, Yaacov Yesha, and Naphtali Rishe. Sksopen: Efficient indexing, querying, and visualization of geo-spatial big data. In *2013 12th International Conference on Machine Learning and Applications*, volume 2, pages 495–500, 2013.

[87] Ahmed R Mahmood, Walid G Aref, Ahmed M Aly, and Mingjie Tang. Atlas: on the expression of spatial-keyword group queries using extended relational constructs. In *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 1–10, 2016.

[88] Paras Mehta, Dimitrios Skoutas, Dimitris Sacharidis, and Agnès Voisard. Coverage and diversity aware top-k query for spatio-temporal posts. In *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPACIAL '16, New York, NY, USA, 2016. Association for Computing Machinery.

[89] Paras Mehta, Dimitrios Skoutas, and Agnès Voisard. Spatio-temporal keyword queries for moving objects. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '15, New York, NY, USA, 2015. Association for Computing Machinery.

[90] Carlos Minervino, Claudio Campelo, Maxwell Oliveira, and Salatiel Silva. Qqespm: A quantitative and qualitative spatial pattern matching algorithm. *ArXiv*, abs/2312.08992, 2023.

[91] Reinhard Moratz and Marco Ragni. Qualitative spatial reasoning about relative point position. *Journal of Visual Languages & Computing*, 19(1):75–98, 2008. Spatial and Image-based Information Systems.

[92] Sergey Nepomnyachiy, Bluma Gelley, Wei Jiang, and Tehila Minkus. What, where, and when: keyword search with spatio-temporal ranges. In *Proceedings of the 8th Workshop on Geographic Information Retrieval*, GIR '14, New York, NY, USA, 2014. Association for Computing Machinery.

[93] Jürg Nievergelt and Peter Widmayer. Spatial data structures: Concepts and design choices. In *Handbook of Computational Geometry*, pages 725–764. Elsevier, 2000.

[94] Open GIS Consortium, Inc. *OpenGIS Simple Features Specification for SQL*, 3 1998. Revision 1.0.

[95] Open GIS Consortium, Inc. *OpenGIS Simple Features Specification for SQL*, 5 1999. Revision 1.1.

[96] Zhihu Qian, Jiajie Xu, Kai Zheng, Pengpeng Zhao, and Xiaofang Zhou. Semantic-aware top-k spatial keyword queries. *World Wide Web*, 21:573–594, 2018.

[97] Gabriel Joseph Ramos Rafael. Busca por grupos de pontos de interesse usando processamento qualitativo de regiões espaciais. Master's thesis, Universidade Federal de Campina Grande, Centro de Engenharia Elétrica e Informática, Programa de Pós-Graduação em Ciência da Computação, Campina Grande, Paraíba, Brasil, 2021.

[98] David A Randell, Zhan Cui, and Anthony G Cohn. A spatial logic based on regions and connection. *KR*, 92:165–176, 1992.

[99] Jinfeng Rao, Jimmy Lin, and Hanan Samet. Partitioning strategies for spatio-textual similarity join. In *Proceedings of the 3rd ACM SIGSPATIAL International Workshop on Analytics for Big Geospatial Data*, pages 40–49, 2014.

[100] Jochen Renz, editor. *The Region Connection Calculus*, pages 41–50. Springer Berlin Heidelberg, Berlin, Heidelberg, 2002.

[101] Joao B Rocha-Junior, Orestis Gkorgkas, Simon Jonassen, and Kjetil Nørvåg. Efficient processing of top-k spatial keyword queries. In *Advances in Spatial and Temporal Databases: 12th International Symposium, SSTD 2011, Minneapolis, MN, USA, August 24-26, 2011, Proceedings 12*, pages 205–222. Springer, 2011.

[102] George Roumelis, Polychronis Velentzas, Michael Vassilakopoulos, Antonio Corral, Athanasios Fevgas, and Yannis Manolopoulos. Parallel processing of spatial batch-queries using xbr+-trees in solid-state drives. *Cluster Computing*, 23(3):1555–1575, 2020.

[103] Steven Schockaert, Chris Cornelis, Martine De Cock, and Etienne E. Kerre. Fuzzy spatial relations between vague regions. In *2006 3rd International IEEE Conference Intelligent Systems*, pages 221–226, 2006.

[104] Du Shihong, Wang Qiao, and Qin Qiming. Definitions of natural-language spatial relations: Combining topology and directions. *Geo-Spatial Information Science*, 9(1):55–64, 2006.

[105] Anders Skovsgaard and Christian S Jensen. Finding top-k relevant groups of spatial web objects. *The VLDB Journal*, 24:537–555, 2015.

[106] Jiabao Sun, Jiajie Xu, Kai Zheng, and Chengfei Liu. Interactive spatial keyword querying with semantics. CIKM '17, page 17271736, New York, NY, USA, 2017. Association for Computing Machinery.

[107] Panagiotis Tampakis, Dimitris Spyrellis, Christos Doulkeridis, Nikos Pelekis, Christos Kalyvas, and Akrivi Vlachou. A novel indexing method for spatial-keyword range queries. In *Proceedings of the 17th International Symposium on Spatial and Temporal Databases*, pages 54–63, 2021.

[108] David Taniar and Wenny Rahayu. A taxonomy for region queries in spatial databases. *Journal of Computer and System Sciences*, 81(8):1508–1531, 2015.

[109] Yufei Tao and Cheng Sheng. Fast nearest neighbor search with keywords. *IEEE Transactions on Knowledge and Data Engineering*, 26(4):878–888, 2014.

[110] George Tsatsanifos and Akrivi Vlachou. On processing top-k spatio-textual preference queries. In *EDBT*, pages 433–444, 2015.

[111] Subodh Vaid, Christopher B Jones, Hideo Joho, and Mark Sanderson. Spatio-textual indexing for geographical search on the web. In *Advances in Spatial and Temporal Databases: 9th International Symposium, SSTD 2005, Angra dos Reis, Brazil, August 22-24, 2005. Proceedings 9*, pages 218–235. Springer, 2005.

[112] Jan Oliver Wallgr¨un, Diedrich Wolter, and Kai-Florian Richter. Qualitative matching of spatial information. In *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '10, page 300309, New York, NY, USA, 2010. Association for Computing Machinery.

[113] Dingming Wu, Gao Cong, and Christian S Jensen. A framework for efficient spatial web object retrieval. *The VLDB Journal*, 21:797–822, 2012.

[114] Dingming Wu, Yafei Li, Byron Choi, and Jianliang Xu. Social-aware top-k spatial keyword search. In *2014 IEEE 15th International Conference on Mobile Data Management*, volume 1, pages 235–244, 2014.

[115] Dingming Wu, Man Lung Yiu, Gao Cong, and Christian S Jensen. Joint top-k spatial keyword query processing. *IEEE Transactions on Knowledge and Data Engineering*, 24(10):1889–1903, 2011.

[116] Tao Xu, Aopeng Xu, Joseph Mango, Pengfei Liu, Xiaqing Ma, and Lei Zhang. Efficient processing of top-k frequent spatial keyword queries. *Scientific Reports*, 12(1):7352, 2022.

[117] Junye Yang, Yong Zhang, Xiaofang Zhou, Jin Wang, Huiqi Hu, and Chunxiao Xing. A hierarchical framework for top-k location-aware error-tolerant keyword search. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pages 986–997, 2019.

[118] Mingyang Yang, Long Zheng, Yanchao Lu, Minyi Guo, and Jie Li. Cloud-assisted spatio-textual k nearest neighbor joins in sensor networks. In *2015 1st International*

Conference on Industrial Networks and Intelligent Systems (INISCom), pages 12–17, 2015.

[119] Man Lung Yiu, Xiangyuan Dai, Nikos Mamoulis, and Michail Vaitis. Top-k spatial preference queries. In *2007 IEEE 23rd International Conference on Data Engineering*, pages 1076–1085, 2007.

[120] Chengyuan Zhang, Ying Zhang, Wenjie Zhang, and Xuemin Lin. Inverted linear quadtree: Efficient top k spatial keyword search. *IEEE Transactions on Knowledge and Data Engineering*, 28(7):1706–1721, 2016.

[121] Dongxiang Zhang, Yeow Meng Chee, Anirban Mondal, Anthony KH Tung, and Masaru Kitsuregawa. Keyword search in spatial databases: Towards searching by document. In *2009 IEEE 25th international conference on data engineering*, pages 688–699. IEEE, 2009.

[122] Dongxiang Zhang, Beng Chin Ooi, and Anthony K. H. Tung. Locating mapped resources in web 2.0. In *2010 IEEE 26th International Conference on Data Engineering (ICDE 2010)*, pages 521–532, 2010.

[123] Dongxiang Zhang, Kian-Lee Tan, and Anthony K. H. Tung. Scalable top-k spatial keyword search. EDBT '13, page 359370, New York, NY, USA, 2013. Association for Computing Machinery.

[124] Jun Zhang, Manli Zhu, Dimitris Papadias, Yufei Tao, and Dik Lun Lee. Location-based spatial queries. SIGMOD '03, page 443454, New York, NY, USA, 2003. Association for Computing Machinery.

[125] Li Zhang, Xiaoping Sun, and Hai Zhuge. Density-based spatial keyword querying. *Future Generation Computer Systems*, 32:211–221, 2014.

[126] Pengfei Zhang, Huaizhong Lin, Bin Yao, and Dongming Lu. Level-aware collective spatial keyword queries. *Information Sciences*, 378:194–214, 2017.

[127] Yu Zhang, Youzhong Ma, and Xiaofeng Meng. Efficient spatio-textual similarity join using mapreduce. In *2014 IEEE/WIC/ACM International Joint Conferences on Web*

Intelligence (WI) and Intelligent Agent Technologies (IAT), volume 1, pages 52–59. IEEE, 2014.

[128] Li Zhigang, Liangtian, and Yang Wunian. Research of gis-based urban disaster emergency management information system. In *2010 International Conference on Computer and Communication Technologies in Agriculture Engineering*, volume 2, pages 484–487, 2010.

[129] Yinghua Zhou, Xing Xie, Chuang Wang, Yuchang Gong, and Wei-Ying Ma. Hybrid index structures for location-based web search. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 155–162, 2005.

Appendix A

QQ-SPM Application Prototype

This appendix presents a prototype application designed to showcase the main features of a QQ-SPM search through a POI search example. We developed a basic web application using the Flask Python framework to demonstrate a possible implementation of a QQ-SPM query focused on POI searches. The application consists of a single screen, depicted in Figure A.1 via a screenshot.

The search example involves a user searching for a residential building to rent an apartment. The user has specific preferences: the building should be located within 1000 meters of a shopping mall. Additionally, the mall should have an on-site fitness center. The user also prefers not to live near any cemeteries.

We marked colored rectangles in Figure A.1 to indicate the five important regions of the search screen. Region A (in red) is where the user specifies query requirements by selecting keywords, distances, connectivity, and exclusion constraints to construct a specific spatial pattern to be searched. Region B (in blue) provides a real-time graphical representation of the spatial pattern being created by the user. Each time a new rule is added in the requirements panel, the drawing in this region is updated, reflecting the current spatial pattern that will be submitted to the search backend when the "Search Pattern" button is pressed. Proximity distance constraints are shown with black edges, exclusion constraints with red edges, and qualitative topological constraints with blue edges in the spatial pattern drawing.

After pressing the "Search Pattern" button, the first result, i.e., the initial group of POIs matching the search pattern, is displayed below in the same screen. Region C (in orange) within Figure A.1 lists the names associated with the POIs in the first group shown as a

result. Their locations on the map appear to the right, in Region D (green rectangle), where the user can manipulate the map to view details of the output results' locations. The user can click the buttons in Region E (pink rectangle) to view other query results. Each time the user clicks on a different page number, Regions C and D update with the names of the POIs in the next group and their corresponding map locations.

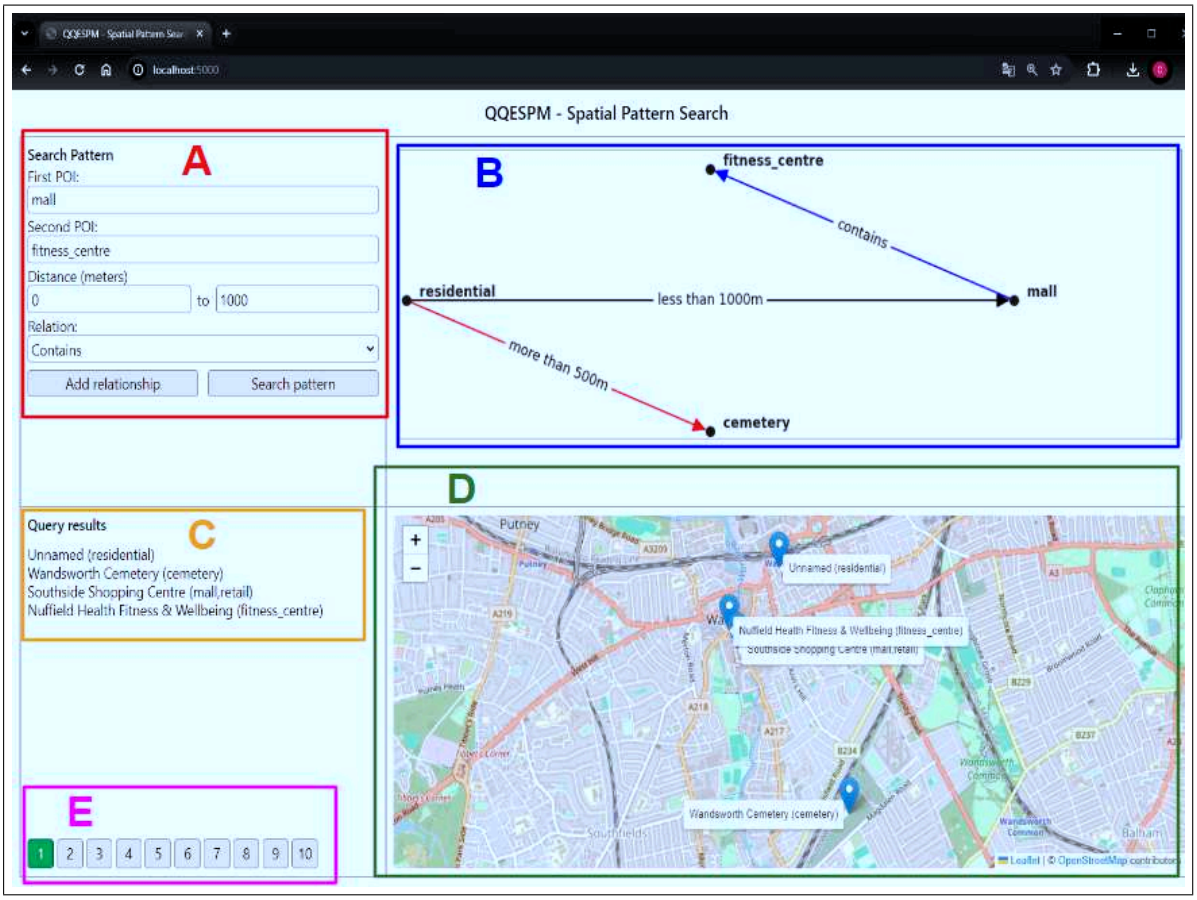

Figure A.1: Overview of the QQ-SPM search tool

Figure A.2 provides a detailed explanation of how to input requirements in such an application. These steps are divided into four screenshots, each showing different stages of inserting pattern requirements. Figure A.2 (A) illustrates how the user begins the search, either by typing or selecting two POIs keywords for creating a specific relationship (pair-wise constraint)

Figure A.2 (B) shows a stage where the user has already inserted the POIs keywords for the relationship requirement and can now choose the distance requirements. We exemplified this with the creation of an exclusion constraint. In this case, the user is looking for a

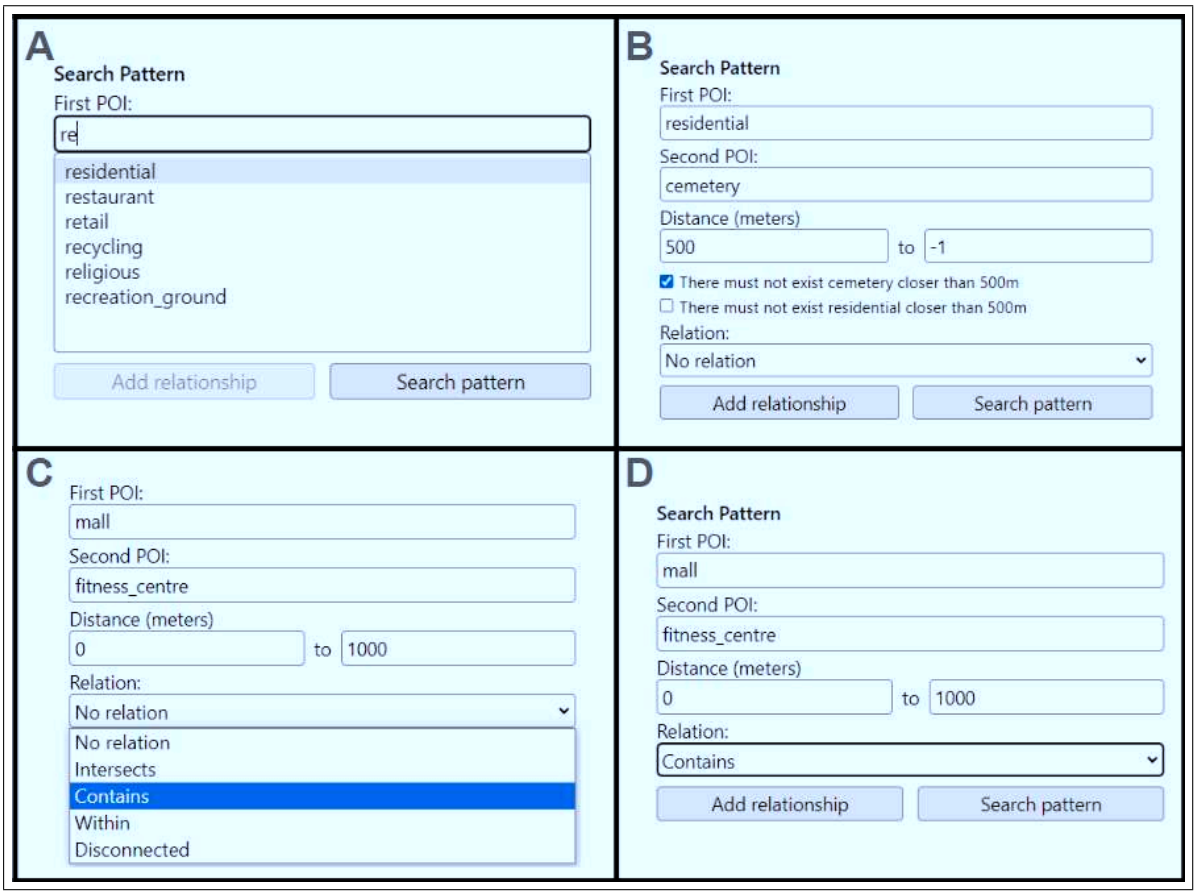

Figure A.2: Input data for searching spatial pattern

residential building that must be at least 500 meters far from cemeteries. After creating the constraint between two POIs, the user clicks the “Add Relationship” button. Subsequently, an update will be visible in the drawing region of the spatial pattern (Region B of Figure A.1).

After creating one relationship, which generates an edge in the search pattern graph, the user can submit additional constrained relationships to compose the search pattern. Figure A.2 (C) demonstrates how the user can select a specific topological constraint, such as choosing a shopping mall that must have a fitness center within its facilities. Once all necessary relationships have been added and the spatial pattern graph meets the user’s expectations, the user clicks on the “Search Pattern” button, as shown in Figure A.2 (D). The query is then processed, and the user can find locations with POIs matching their criteria and enjoy their choices.

Such an application could be utilized in scenarios beyond finding suitable residential areas. As discussed in Chapter 1, a spatial pattern search tool could be invaluable for trip planning, urban planning, scene recognition, and many other applications. These applications can include POI searches, geo-tagged document retrieval, and any other geo-textual searches compatible with the QQ-SPM query format.