一种面向大规模空间文本数据的空间结构匹配算法

刘志丹 林维鑫 伍楷舜
(深圳大学计算机与软件学院 广东 深圳 518060)

摘 要 为了支持各类基于位置的服务,人们提出了各种查询和搜索空间文本数据的方法和技术.传统的空间关键字查询和近期提出的空间模式匹配不支持用户定义查询关键字对象以及对象之间细致的空间结构关系,使得查询结果集庞大但无效结果偏多,不能满足用户高效且精确的查询需求.本文因此提出了一种新的查询模式——空间结构匹配查询(Spatial Structure Matching,SSM),允许用户定义一组查询关键字对象并指定任意两个对象之间的距离和方向约束.为了解决SSM查询问题,本文首先提出了一种基于多路连接的基准方法,将SSM查询问题分解为单个对象的关键字匹配,两个对象的边匹配和多个对象的聚合匹配.为了提高SSM查询效率,本文提出了基于扫描线算法的边匹配计算,利用对象的地理位置信息来降低边匹配计算开销.本文利用同时满足查询关键字,距离和方向约束的空间对象构造对象连接图,从而将SSM查询问题转换为在对象连接图上搜索与SSM查询结构同构的子图匹配问题,并且利用经典的子图同构匹配算法求解获得最终的查询结果.在四个大规模空间文本数据集上的实验结果表明,本文所提算法的查询效率远高于对比算法,返回的查询结果集精简有效且在查询时间上提升至少3倍.

关键词 空间文本数据;空间查询;空间结构;扫描线;子图同构匹配
中图法分类号 TP18 **DOI号** 10.11897/SP.J.1016.2022.01261

A Spatial Structure Matching Algorithm for Large Spatial-Textual Datasets

LIU Zhi-Dan LIN Wei-Xin WU Kai-Shun
(*College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, Guangdong* 518060)

Abstract Many techniques have been proposed to query and search on the massive spatial-textual data for supporting various location-based services. Traditional spatial keyword query (SKQ) and the recent spatial pattern matching (SPM), however, cannot accurately capture users' query intention on the interested objects and their spatial relations, resulting in a huge number of irrelevant results. As a result, existing techniques still cannot well satisfy users' query requirements. Therefore, this paper proposes a novel query problem-Spatial Structure Matching (SSM), which allows the user to provide a set of keyword objects and meanwhile define the constraints on both distance and direction for any two concerned objects. To answer an SSM query, this paper firstly presents a multi-way join based baseline approach, which decomposes the SSM query into the keyword matching for each individual object, edge matching for a pair of objects, and aggregation matching for a set of objects. To improve the efficiency, we further propose a sweep-line based edge matching algorithm by exploiting the geographic locations of objects to filter out the pairs of objects that cannot meet distance constraints. In addition, we build an object-connection graph using the objects that meet all constraints, and transform the SSM query into a subgraph isomorphism problem, which searches the subgraphs with similar structure as SSM query over

收稿日期:2021-07-19;在线发布日期:2022-01-17.本课题得到国家自然科学基金项目(62172284,61872248,U2001207)、广东省自然科学基金项目(2020A1515011502,2017A030312008)资助.**刘志丹**,博士,助理教授,硕士生导师,中国计算机学会(CCF)会员,主要研究领域为城市计算、物联网等.E-mail:liuzhidan@szu.edu.cn.**林维鑫**,硕士研究生,主要研究方向为空间数据处理与分析.**伍楷舜**(通信作者),博士,特聘教授,中国计算机学会(CCF)会员,主要研究领域为物联网、移动计算等.E-mail:wu@szu.edu.cn.

the object-connection graph. This problem is well solved by adopting a fast subgraph matching algorithm. Experiments based on four large real-world spatial-textual datasets demonstrate that the proposed approach significantly outperforms the compared approaches by providing refined and effective results, while reducing the query processing time by at least 3 times.

Keywords spatial-textual data; spatial query; spatial structure; sweep-line; subgraph isomorphism

1 引 言

随着移动互联网,全球定位等技术的蓬勃发展和移动智能终端设备的日益普及,基于位置的服务(Location-based Services,LBS)日益流行,进而加速了地理位置数据和文本数据的融合,产生了海量的空间文本数据[1].例如,人们可以通过大众点评等LBS应用对兴趣点(Point of Interest,POI)(如餐厅、酒店、旅游景点等),进行文字描述或评价,使得兴趣点的地理位置与特定的文本数据产生了密切的关联.对海量空间文本数据的处理分析可进一步衍生出各类智慧城市应用,如城市区域搜索[2]、POI推荐[3]、旅游规划[4]、人类流动分析[5]等.

为了有效支撑各类LBS和智慧城市应用,人们提出了各种方法来查询和搜索空间文本数据库[6].其中,空间关键字查询(Spatial Keyword Query,SKQ)[7]问题被广泛研究.SKQ根据用户给定的一组查询关键字,空间约束或查询位置在空间文本数据库中检索返回与查询关键字相关且满足空间距离约束的空间文本对象集[8].特别地,一个空间文本对象通常由两部分组成:对象的地理位置和描述该对象的关键字集合[9].例如,用户通过SKQ可以查询获得距离其所在位置500 m以内的所有餐厅的信息.近年来,SKQ问题已经演化出不同类型的SKQ查询技术,例如top-k近邻空间关键字查询[10-12]、m-最近关键字查询[13-14]和集体关键字查询[15-16]等.

尽管SKQ技术可以从海量空间文本数据中搜索返回查询结果,但是由于其不能准确地捕获用户的实际需求,往往返回大量无效的查询结果[17].假定用户甲想搜寻一处住宅,并希望该住宅离餐厅较近(如小于500 m)而离公园距离适中(如介于500 m和1000 m之间).一个典型的SKQ查询将以{“住宅”,“公园”,“餐厅”}作为查询关键字,并且以1000 m作为空间约束使得3个空间对象的相互距离符合用户的距离约束.近期,研究者们提出了一种新的面向空间文本数据库的查询模式:空间模式匹配(Spatial Pattern Matching,SPM)[17-19].在SKQ的基础上,SPM允许用户进一步定义任意两个空间对象之间的距离约束.相比于SKQ,SPM使得用户能更准确地表述其查询需求.SPM以用户提供的模式作为输入,在空间文本数据库中搜索同时满足任意对象之间的距离约束和关键字约束的匹配实例.针对上述用户甲的查询需求,SPM可以表述为{“住宅”-“餐厅”:(0 m,500 m);“住宅”-“公园”:(500 m,1000 m)},并且将获得更为精确的查询结果.

在大规模的空间文本数据库中,SPM仅仅利用关键字和对象之间距离的约束来进行查询,仍然会返回很多无用结果,造成用户的筛选困扰.例如,用户乙曾经在北京某餐厅用餐,体验很好想再次前去用餐,但是她忘记了餐厅的具体名称和位置,只记得距离餐厅100 m~200 m处有家美发店以及距离餐厅100 m以内有家银行.SPM可以将该查询表示为{“餐厅”-“美发店”:(100 m,200 m);“餐厅”-“银行”:(0 m,100 m)}.在北京POI数据集上运行此SPM查询将获得1708个查询结果(请参考第6.1节以了解更多POI数据集的介绍).如此多的查询结果显然不能让用户乙快速定位到自己想找的餐厅信息.事实上,用户乙还记得美发店在餐厅的正东方向而银行在餐厅的正南方向附近.利用对象的方位信息对SPM的查询结果进行过滤,最终将获得11个有效查询结果.在实际生活中,用户对所查询的空间对象往往拥有更多信息和需求.除了关键字和空间距离的约束之外,对象之间的方位信息也是重要的查询线索[20-21].支持用户灵活定制查询关键字,空间距离和方位等方面约束的查询,可以精确表示用户的真实查询需求,进而获得更为有效和精简的查询结果.

针对SKQ和SPM不能精确捕获用户查询需求及查询低效的问题,本文研究了一种新颖的空间文本数据查询问题:空间结构匹配(Spatial Structure Matching,SSM)查询问题.SSM允许用户提供特定的关键字对象,并对任意一对关键字对象赋予空间距离和方向的结构约束,从而形成一个具有特定空

间结构的查询模式. 相较于 SKQ 和 SPM，SSM 可以更为灵活地定制空间文本对象在关键字，距离和方向三方面的约束，根据指定查询结构在大型空间文本数据库中查询获得精简而有效的匹配结果. 此外，SSM 查询问题的研究有助于进一步理解地理区域的空间形态结构，有利于城市区域相似性[22-23]和城市规划[24]等方面的研究.

然而在大型空间文本数据库中高效执行 SSM 查询面临巨大挑战. 由于在空间文本数据库中每个对象通常以独立的形式存在，而空间结构约束是对多个对象之间的关系的多方面约束，所以 SSM 需要核查空间文本数据库中任意两个对象之间是否满足关键字，空间距离和方向的关系约束，然后将符合约束条件的对象对聚合以形成最终的查询结果实例. 实际的空间文本数据库往往包含海量的空间文本对象，因此其匹配计算复杂度非常高. 虽然可利用 SPM 查询技术[17-19]首先获得符合关键字和距离约束的查询实例，然后再利用对象之间的方向约束进一步过滤匹配实例以获得最终的查询结果. 然而，这种方法不能有效利用各类约束条件提前过滤无关的空间文本对象，其计算效率仍然低下. 此外，Fang 等人[18-19]提出的 SPM 查询技术需要提前建立静态对象索引，不能适应空间文本数据库的动态更新，将造成额外的索引更新和维护开销.

为了有效解决 SSM 查询问题，本文提出了一种无需索引的高效空间结构匹配算法. 本文算法首先扫描空间文本数据库获得满足查询关键字约束的空间文本对象集，然后为各个对象构造匹配范围圆. 对象圆相交即表示两个对象满足一定的距离约束. 为了加快对象间相交关系的判断，本文提出扫描线算法，将所有对象圆投影到纵轴，然后依据圆投影的相交关系来判断对象之间的相交情况. 为了减少圆投影相交而对象实际不相交而导致的误判，本文提出了分区策略. 算法将进一步核查实际相交对象在关键字、距离和方向三方面的约束. 本文将所有满足查询约束要求的相交对象构建对象连接图，并将 SSM 查询问题转化为在对象连接图上的子图同构问题，利用经典的子图同构匹配算法[25]求得符合查询约束的匹配实例. 本文的算法无需预先建立复杂的索引结构，适用于在动态变化的空间文本数据库中进行高效的 SSM 查询. 本文的主要贡献如下：

(1) 首次提出并定义了面向大规模空间文本数据的空间结构匹配(SSM)查询问题.

(2) 提出了空间文本对象匹配范围圆的概念，设计了基于扫描线算法和子图同构匹配的空间结构匹配算法，可以高效解决 SSM 查询问题. 同时对本文算法进行了深入的时间和空间复杂度分析.

(3) 设计了丰富的对比实验，在真实的大规模空间文本数据集上进行了大量实验，验证了本文所提算法的查询准确性和执行效率.

本文第 2 节介绍面向空间文本数据查询的相关工作；第 3 节形式化定义 SSM 查询问题；第 4 节提出解决 SSM 查询问题的基准方法；第 5 节针对基准方法提出一系列的优化策略；第 6 节介绍对比实验并对实验结果进行了分析；第 7 节总结并展望本文的工作.

2　相关工作

空间关键字查询问题(Spatial Keyword Query，SKQ)是空间查询中常见的问题，旨在空间文本数据库中检索满足空间与文本约束的对象[7]. 按照文献[9，26]，SKQ 查询方法依据是否给定查询位置可以分为近邻 SKQ 查询[7，10-12，20，27-30]和范围 SKQ 查询[13-16]. 近邻 SKQ 查询以一个查询位置和若干关键字作为参数，检索返回距离与查询位置相近且满足文本约束的空间对象[9]. 例如，top-k SKQ 查询[7，10-12]返回包含所有查询关键字且与查询位置最近的前 k 个空间对象. 近邻 SKQ 查询一般考虑空间距离和文本相关性来对查询结果进行排序[8]. 相比于静态查询，近期研究工作[27-30]关注在空间文本数据上的动态近邻 SKQ 查询，以解决用户在移动过程中也能找到满足约束条件的空间对象结果. 例如，动态近邻 SKQ 查询可以为共享出行司机推荐行驶路线附近的潜在乘客[29-30]. 特别地，Li 等人[20]研究了方向感知的近邻 SKQ 查询，该查询不仅包含了查询位置和关键字约束，还包含了方向区间的约束. 符合查询的空间对象需处在查询位置的指定方向区间，同时覆盖给定的关键字并离查询位置较近. 本文提出的 SSM 问题与文献[20]存在诸多不同. SSM 问题定义了空间对象之间更为细致的方向约束，而非空间对象与查询位置的相对方向. 此外，SSM 问题没有指定查询位置，而是在较大的查询范围内检索满足特定空间结构的对象集.

与近邻 SKQ 查询不同，范围 SKQ 查询不依赖特定的查询位置，用于检索满足查询关键字约束且空间关系紧密的对象集[9]，相关研究包括 m-最近关键字查询(m-Closest Keywords Query，mCK)[13-14]

和集体关键字查询(Collective Spatial Keyword Query,CoSKQ)[15-16]等.具体地,mCK[13-14]检索覆盖所有查询关键字且直径最小的空间对象集,其中直径定义为对象集中相距最远的两个空间对象的距离.CoSKQ[15-16]在空间文本数据库中搜索以最小的成本共同覆盖一组给定关键字的空间对象集.对SKQ问题感兴趣的读者可参考综述文献[8,31].

传统的SKQ问题没有考虑任意两个对象之间的距离约束,因此不能支持用户在空间约束方面细粒度的查询要求.为了弥补SKQ的不足,近期有学者提出了面向空间文本数据的空间模式匹配查询(Spatial Pattern Matching,SPM)[17].SPM允许用户制定拥有多个关键字对象和不同对象之间的距离约束的空间模式作为查询输入,并返回数据库中所有满足空间模式约束的匹配实例.为解决SPM问题,Fang等人[18]提出了多对连接算法(Multi-Pair-Join,MPJ)及改进算法多星连接算法(Multi-Star-Join,MSJ)[19].MPJ和MSJ算法利用IR树(Inverted R-tree)索引结构[10,32]实现空间文本对象的索引以加快检索,基于边连接(Pair-Join,PJ)操作来搜索合适的对象对,并基于多路连接(Multi-Way Join)[33]聚合得到最终的SPM查询结果.PJ操作的主要思想是:自顶向下地在IR树每层节点中寻找该层的匹配节点对;然后搜索这些匹配节点对的孩子节点,在其中找下一层的匹配节点对;重复向下搜索直到在IR树叶子节点层找到查询对象边的所有匹配.在获得所有需要的对象边以后,MPJ和MSJ根据一定的连接顺序将所有匹配边根据顶点进行连接以形成最终的查询结果.与MPJ不同,MSJ算法首先利用对象的空间关系对SPM查询进行优化,缩小对象之间的距离约束范围以强化在IR树上的剪枝效果.MPJ和MSJ算法需要针对空间文本数据库建立静态的IR树索引,然后在该索引结构上进行搜索.因此空间文本数据库的动态变化会带来频繁的索引建立与更新操作,降低算法的可用性.此外,MPJ和MSJ算法的剪枝策略没有充分利用对象的空间位置信息.

虽然SPM相比于SKQ增加了空间对象之间的距离约束,但其在大规模空间文本数据库中检索仍然会返回大量的无效结果,查询效果不理想.为进一步提高空间查询效率并满足用户定义精细查询条件的需求,本文提出了SSM查询问题,允许用户定义空间对象之间的关键字,距离和方向等方面的空间结构约束,从而获得更精简且有效的匹配实例.本文提出的SSM查询方法无需预先建立复杂的索引结构,适应于动态变化的空间文本数据库,并充分利用关键字,空间距离和方向等约束设计了优化策略.

3 问题定义

假设$\mathcal{D}$是空间文本对象集合.每个空间文本对象①记为$o_a=((x_a,y_a),doc(o_a))$,其中(x_a,y_a)表示o_a的地理位置(如经纬度),$doc(o_a)$表示描述o_a的关键字集合.$w\in doc(o_a)$表示o_a拥有关键字w.

定义1. 空间结构:空间结构表示为$S=\mathcal{G}(\mathcal{V},\mathcal{E})$.每个顶点$v_i\in\mathcal{V}$拥有一个关键字$w_i$,如果$w_i\in doc(o)$,则说明对象$o$匹配顶点$v_i$;每条边$(v_i,v_j)\in\mathcal{E}$连接两个顶点$v_i$和$v_j$,并指定它们在距离和方向上的空间关系.

为了容忍用户的不确定输入,每条边$(v_i,v_j)\in\mathcal{E}$定义了距离区间$[d^l_{i,j},d^u_{i,j}]$和方向区间$[\theta^l_{i,j},\theta^u_{i,j}]$.具体地,距离区间表示$\mathcal{D}$中匹配顶点的两个对象之间距离的下限(即$d^l_{i,j}$)和上限(即$d^u_{i,j}$),而方向区间表示匹配对象之间方向的下限(即$\theta^l_{i,j}$)和上限(即$\theta^u_{i,j}$).两个对象之间的方向定义为:以两个对象中纵坐标值较小的对象为参考点,将参考点与另一个对象的连线与水平正方向产生的夹角作为两个对象的方向.

定义2. 边匹配:对于空间结构S中的任意一条边$(v_i,v_j)\in\mathcal{E}$,如果两个对象o_a和o_b分别匹配顶点v_i和v_j(即v_i的关键字w_i和v_j的关键字w_j分别与o_a和o_b匹配),并且o_a和o_b满足(v_i,v_j),在距离和方向上的约束(即o_a和o_b的地理空间距离在$[d^l_{i,j},d^u_{i,j}]$范围内,方向在$[\theta^l_{i,j},\theta^u_{i,j}]$范围内),则称对象对$(o_a,o_b)$是边$(v_i,v_j)$的一个边匹配.

定义3. 聚合匹配:给定空间结构$S=\mathcal{G}(\mathcal{V},\mathcal{E})$,如果存在对象子集$\mathcal{T}\subset\mathcal{D}$,使得空间结构$S$与$\mathcal{T}$存在一个满射关系$\pi:\mathcal{V}\to\mathcal{T}$.即每条边$(v_i,v_j)\in\mathcal{E}$,对象对$(\pi(v_i),\pi(v_j))$是边$(v_i,v_j)$的一个边匹配,则称对象子集$\mathcal{T}$是空间结构$S$的一个聚合匹配.

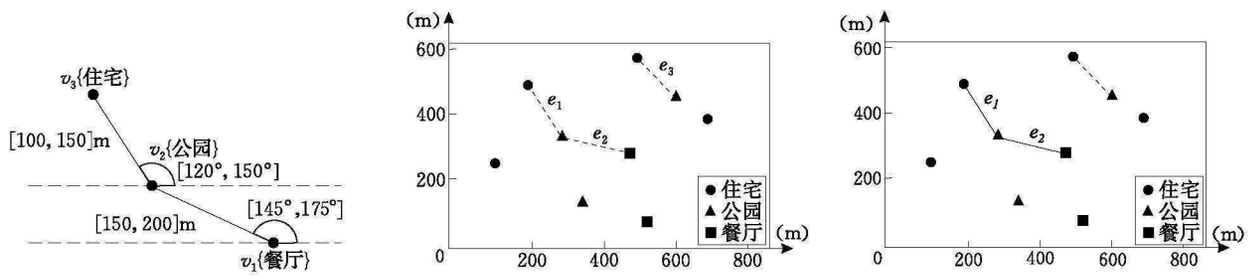
定义4. 空间结构匹配问题(Spatial Structure Matching problem,SSM):给定对象集合$\mathcal{D}$和用户输入的空间结构S,SSM查询问题旨在从$\mathcal{D}$中找出S的所有聚合匹配.

图1(a)示意了空间结构查询$S=\mathcal{G}(\mathcal{V},\mathcal{E})$.其中,顶点集合$\mathcal{V}=\{v_1,v_2,v_3\}$,顶点$v_1$,$v_2$和$v_3$分别拥

① 下文中,空间文本对象简写为对象.

有关键字“餐厅”、“公园”和“住宅”;边集合$\mathcal{E}=\{(v_1,v_2),(v_2,v_3)\}$,其中边$(v_1,v_2)$的距离区间和方向区间分别为[150,200]m 和[145°,175°],而边(v_2,v_3)的距离区间和方向区间分别为[100,150]m 和[120°,150°].图 1(b)示意了在对象集合$\mathcal{D}$中执行 SSM 查询 S 的边匹配结果,返回了 3 个匹配的对象组合边$\{e_1,e_2,e_3\}$.例如,对象边 e_2 匹配了 S 中的边(v_1,v_2),即 e_2 的两个对象同时满足边(v_1,v_2)的查询关键字,空间距离和方向的约束.将所有的匹配边进行聚合,最终得到图 1(c)所示的符合查询 S 的匹配实例,即由 e_1 和 e_2 构成的对象集是最符合用户空间查询需求的结果.

相较于传统的 SKQ[7,8,31] 和 SPM[17-19],本文关注的 SSM 问题在关键字和空间距离约束的基础上增加了对象之间的方向约束,支持更精确地满足特定空间结构的对象集查询.SPM 问题被证明是 NP 难问题[18-19],而 SSM 问题比 SPM 问题更为复杂,因此也是 NP 难问题.为了有效解决 SSM 问题,本文将首先提出基准方法,然后提出优化策略来提高计算效率.



(a) 空间结构S　　(b) 边匹配结果　　(c) 聚合匹配结果

图 1　一个简单的 SSM 查询例子

4　基准方法

如果以空间结构 S 整体来匹配对象集合$\mathcal{D}$中的任意对象子集,其查询计算量是十分巨大的.因此,基准方法首先筛选符合 S 关键字约束的对象集合,然后在对象集合中寻找满足 S 任意一条边在空间距离和方向约束的边匹配,最后将合适的边匹配聚合形成最终的查询结果.基准方法的主要步骤如下.

(1)筛选符合查询关键字的空间对象.对于 S 中的每个顶点 v_i 及查询关键字 w_i,遍历对象集合$\mathcal{D}$以构造关键字对象集 $K_i=\{o|w_i\in doc(o)\&o\in\mathcal{D}\}$,即 K_i 包含对象集合$\mathcal{D}$所有可以匹配顶点 v_i 的对象.

(2)边匹配计算.对于 S 中的每条边(v_1,v_2),构造边匹配集合 E_{ij}.具体地,遍历验证集合 K_i 和集合 K_j 的每一个对象对(o_a,o_b)是否满足边(v_i,v_j)的空间距离和方向约束,其中 $o_a\in K_i$ 且 $o_b\in K_j$.如果对象对(o_a,o_b)是边(v_i,v_j)的一个边匹配,那么则将(o_a,o_b)添加到集合 E_{ij};否则舍弃该对象对.

(3)聚合匹配计算.从每个边匹配集合 E_{ij} 抽取一个对象对,所有的对象对则形成一个查询匹配实例.如果该实例与 S 满足聚合匹配要求,则将其添加到查询结果集;否则舍弃.遍历检查所有边匹配集合形成的聚合组合即可获得最终的 SSM 查询结果.

基准方法主要遵循多路连接技术[33],因此比较容易实现.然而,基准方法仍然是暴力地枚举所有可能的边匹配和聚合匹配组合,未有效利用对象的地理分布信息和空间结构查询 S 的空间约束信息,不能很好地过滤不符合查询要求的对象或者对象对,因此其计算复杂度较高.

5　优化设计

本节将对基准方法进行优化.第 5.1 节介绍一种基于扫描线的快速边匹配计算方法;第 5.2 节介绍基于子图同构匹配的聚合匹配计算方法,从而获得最终的 SSM 查询结果;第 5.3 节深入分析了本文 SSM 查询方法的时间和空间复杂度.

5.1　基于扫描线的边匹配计算

基准方法在匹配边(v_i,v_j)的对象对(o_a,o_b)时没有考虑对象 o_a 和 o_b 的实际位置,可能导致两个空间相距很远的对象也形成一条边,从而产生没必要的核查计算.虽然可以采用一些空间索引结构(如四叉树索引)提前对对象集合$\mathcal{D}$中的所有对象进行索引,然后基于索引结构进行符合距离约束的对象边的构造.但是,动态更新的对象集合$\mathcal{D}$需要经常计算更新索引结构,将带来额外的计算量.因此,本文提出一种适用于数据库动态更新的边匹配计算方法.

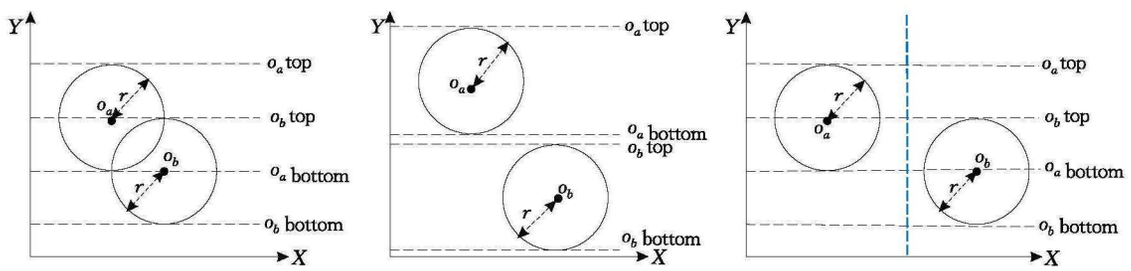
5.1.1　扫描线算法

优化方法仍然首先获得符合查询关键字约束的

对象子集.扫描对象集合$\mathcal{D}$,将能够匹配查询S的任意关键字的对象构造匹配对象集$\mathcal{K}$.对于任意对象$o_a\in\mathcal{D}$,构造以o_a的位置(x_a,y_a)为中心,半径为r的圆.如果对象o_a和对象o_b的圆相交,则o_a和o_b的空间距离将落在$(0,2r]$范围之内.因此,可以通过设计合适的半径r来为每个对象构造圆来约束其匹配对象的范围,并根据两个对象的圆是否相交来判断它们是否可以构成一条满足一定距离约束的匹配边.该方法可快速过滤掉大量不可能形成边的对象对组合,节省大量的计算成本.本文称以对象o_a为圆心,r为半径的圆为对象o_a的匹配范围圆.

定义 5. 对象的匹配范围圆:给定空间结构查询$S=\mathcal{G}(\mathcal{V},\mathcal{E})$,令$d_{\max}=\max\{d^u_{i,j}\mid(v_i,v_j)\in\mathcal{E}\}$,那么任意对象$o_a\in\mathcal{K}$针对$S$的匹配范围圆是以$o_a$的位置$(x_a,y_a)$为圆心,半径为$r=\frac{d_{\max}}{2}$的圆形区域.

对于$\mathcal{K}$中的任意对象,构造其匹配范围圆,然后通过判断任意两个圆是否相交即可保留最有可能满足查询S的边匹配条件的对象对.但是,这样的边匹配计算需要遍历所有可能的对象对并判断它们的圆是否相交,计算复杂度仍然较高.因此,本文提出将对象的圆投影到纵轴,并根据它们的投影是否有相交来判断两个圆是否相交.如图 2(a)所示,如果两个圆相交,则它们在纵轴上的投影也必然相交;而图 2(b)则显示两个不相交的圆,它们的投影也不相交.对于对象o_a的圆,记录其在纵轴的投影区间为$[o_a.\text{bottom},o_a.\text{top}]$,其中$o_a.\text{bottom}$表示$o_a$的圆的底部("bottom")扫描线,而$o_a.\text{top}$表示其顶部("top")扫描线.根据图 2(a)和图 2(b),如果两个圆相交,那么一个圆的顶部扫描线必然处于另外一个圆的顶部扫描线和底部扫描线之间.



(a) 圆及投影相交 (b) 圆及投影皆不相交 (c) 圆不相交但投影相交

图 2 两个空间对象的圆及投影关系示例

算法 1. 扫描线算法.

输入:对象子集$\mathcal{T}\subset\mathcal{D}$,匹配范围圆半径$r$
输出:对象相交集$\mathcal{R}$
1. 初始化结果集$\mathcal{R}\leftarrow\varnothing$;
2. 初始化键值映射表$M\leftarrow\varnothing$;
3. FOR $\mathcal{T}$中的每个对象o_a
4. 构造以o_a为圆心,半径为r的匹配范围圆;
5. WHILE 自顶向下扫描所有的圆
6. IF 扫描到对象o_a的圆的顶部
7. FOR M中每个键值对象κ
8. $M[\kappa]=M[\kappa]\cup\{o_a\}$;
9. 创建o_a的相交对象集$M[o_a]\leftarrow\varnothing$;
10. ELSE IF 扫描到对象o_a的圆的底部
11. //添加对象o_a的相交对象集到最终结果$\mathcal{R}$
12. $\mathcal{R}=\mathcal{R}\cup M[o_a]$;
13. //将对象o_a的相交对象集从M中删除
14. $M=M\backslash M[o_a]$;
15. RETURN $\mathcal{R}$;

算法 1 描述了用于记录对象圆相交情况的扫描线算法.该算法利用键值映射表M来记录算法执行过程中各个对象的相交对象集.例如,$M[o_a]$表示以对象o_a为键值的一个集合,包含与o_a的圆相交的其他对象.对于给定的对象子集$\mathcal{T}\subset\mathcal{D}$,首先为$\mathcal{T}$中的每个对象$o_a$构造一个以其位置为圆心,$r$为半径的圆(第 3、4 行).然后,自顶向下地扫描所有对象的圆.如果扫描到对象o_a的圆的顶部,那么将o_a添加到M中已有键值的各个相交集合中(第 7、8 行);然后在M中添加键值o_a,并赋值$M[o_a]$为空集(第 9 行).如果扫描到对象o_a的圆的底部,那么与o_a相交的所有其他对象已经找到,可以将集合$M[o_a]$添加到最终结果集$\mathcal{R}$(第 12 行),并将$M[o_a]$从M中移除(第 14 行).当算法扫描完所有圆之后,程序将返回$\mathcal{R}$.

例 1. 表 1 详细记录了在图 3 所示的对象集$\mathcal{T}=\{o_a,o_b,o_c,o_d,o_e\}$上执行扫描线算法时各个步骤的中间结果$M$和最终结果$\mathcal{R}$的值.算法 1 自顶向下地扫描图 3 中的所有圆,直至扫描到对象o_e圆的底部结束,最终结果保存在$\mathcal{R}$中(见表 1 的最后一行).结果集$\mathcal{R}$表示对象o_a与o_b及o_c相交,o_b与o_c相交,o_c与o_d相交,o_d与o_e相交.$\mathcal{R}$中键值为空的记录(如o_e:{})可以删除,因为此类对象不与其它对象相交,不能构成合理的查询结果.

表 1　在图 3 上执行扫描线算法的过程

扫描对象	中间结果 M	最终结果$\mathcal{R}$
o_a.top	$M[o_a]\leftarrow\varnothing$	$\varnothing$
o_b.top	$M[o_a]=\{o_b\}$, $M[o_b]\leftarrow\varnothing$	$\varnothing$
o_c.top	$M[o_a]=\{o_b,o_c\}$, $M[o_b]=\{o_c\}$, $M[o_c]\leftarrow\varnothing$	$\varnothing$
o_a.bottom	$M[o_b]=\{o_c\}$, $M[o_c]\leftarrow\varnothing$	$o_a:\{o_b,o_c\}$
o_b.bottom	$M[o_c]=\varnothing$	$o_a:\{o_b,o_c\}$, $o_b:\{o_c\}$
o_d.top	$M[o_c]=\{o_d\}$, $M[o_d]\leftarrow\varnothing$	$o_a:\{o_b,o_c\}$, $o_b:\{o_c\}$
o_c.bottom	$M[o_d]=\varnothing$	$o_a:\{o_b,o_c\}$, $o_b:\{o_c\}$, $o_c:\{o_d\}$
o_e.top	$M[o_d]=\{o_e\}$, $M[o_e]\leftarrow\varnothing$	$o_a:\{o_b,o_c\}$, $o_b:\{o_c\}$, $o_c:\{o_d\}$
o_d.bottom	$M[o_e]=\varnothing$	$o_a:\{o_b,o_c\}$, $o_b:\{o_c\}$, $o_c:\{o_d\}$, $o_d:\{o_e\}$
o_e.bottom	$\varnothing$	$o_a:\{o_b,o_c\}$, $o_b:\{o_c\}$, $o_c:\{o_d\}$, $o_d:\{o_e\}$, $o_e:\{\}$

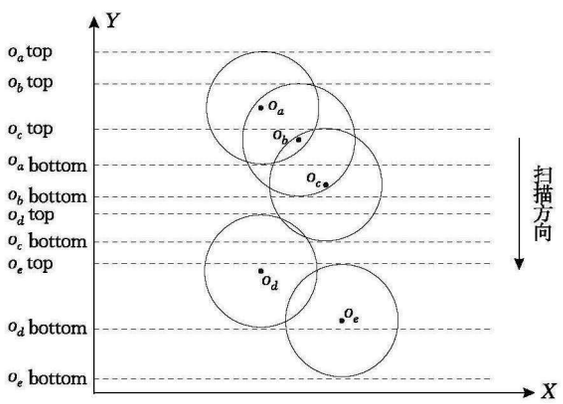


图 3　扫描线算法执行示例

5.1.2　分区策略

将二维空间的圆投影到一维空间的纵轴来判断两个圆是否相交容易导致误判.例如图 2(c)所示,对象 o_a的圆与对象 o_b的圆不相交,但是它们在纵轴的投影区域却相交.虽然此类误判的边可以通过后期的边匹配条件核查剔除掉,但是仍然会带来一些不必要的计算开销.为了尽可能避免此类误判,本文提出将空间区域进行分区,然后在各个分区分别执行扫描线算法.如图 2(c)所示,如果将对象 o_a和 o_b划分到不同的分区,分别执行扫描线算法,那么就不会出现 o_a与 o_b相交的误判.

分区的目的在于减少两对象圆相交的误判,同时需要减少处于分区边界的对象圆数目.如果一个对象圆处在分区边界,需要将该对象圆同时分配到两个分区以考察它与其他对象圆的相交情况.这样操作可以避免遗漏可能的两对象圆相交的情况,但是会增加存储和计算开销.理论上,可以设计一种分区方法根据对象圆的分布将它们分派到不同分区,使得较少甚至没有对象圆同处两个分区.但是,这样的分区方法设计较为复杂,将带来额外的计算开销.因此,本文拟采用一种简单的分区策略,即将查询空间划分成等大小的分区.具体地,分区算法将整个查询区域在横轴上划分成 z 个大小为 L 的分区 $\mathbb{P}=\{P_1,P_2,\cdots,P_z\}$,其中第 j 个分区的横轴范围为 $[x_0+(j-1)\times L, x_0+j\times L]$,其中 x_0为查询区域 X 轴的起始坐标.对于$\mathcal{K}$中的任意对象 o_a及其匹配范围圆,分区算法将该圆投影到横轴,获得 o_a在横轴的投影区域$[x_a-r,x_a+r]$.如果 o_a圆的横轴投影区域正好落在分区 P_j,那么将 o_a添加到分区 P_j的空间对象子集 $\mathcal{T}_{P_j}$;如果 o_a圆的横轴投影区域横跨两个区域,如 P_j和 P_{j+1},则将 o_a同时添加到$\mathcal{T}_{P_j}$和$\mathcal{T}_{P_{j+1}}$.合适的分区大小 L 可以减少对象相交与否的误判,同时可以避免将同一对象添加到两个子集而带来的冗余存储和额外计算的开销.本文将通过基于真实数据的实验来寻找合适的分区大小 L,从而平衡对象相交误判率与额外存储开销(见第 6.2.1 节实验).对于分区 P_j,它的分区对象子集$\mathcal{T}_{P_j}$将作为算法 1 的输入以求得分区内对象的相交情况$\mathcal{R}_{P_j}$.各个分区可并行执行扫描线算法以加快处理速度.

例 2.　在图 4 中,查询区域按照 $L=4\times r$ 被划分成 3 个分区 $\mathbb{P}=\{P_1,P_2,P_3\}$.查询区域内的 9 个对象被分配到 3 个分区,即$\mathcal{T}_{P_1}=\{o_a,o_b,o_c\}$,$\mathcal{T}_{P_2}=\{o_b,o_d,o_e,o_f,o_g\}$和$\mathcal{T}_{P_3}=\{o_e,o_f,o_g,o_h,o_i\}$.$\mathcal{T}_{P_1}$,$\mathcal{T}_{P_2}$和$\mathcal{T}_{P_3}$将作为算法 1 的输入以求得各个分区的对象相交情况,各个分区的输出为

$$\mathcal{R}_{P_1}=\{o_a:\{o_b,o_c\},o_b:\{o_c\}\};$$

$$\mathcal{R}_{P_2}=\{o_f:\{o_b\},o_g:\{o_f,o_e\},o_d:\{o_e\}\};$$

$$\mathcal{R}_{P_3}=\{o_h:\{o_f\},o_f:\{o_i\},o_i:\{o_g\},o_g:\{o_e\}\}.$$

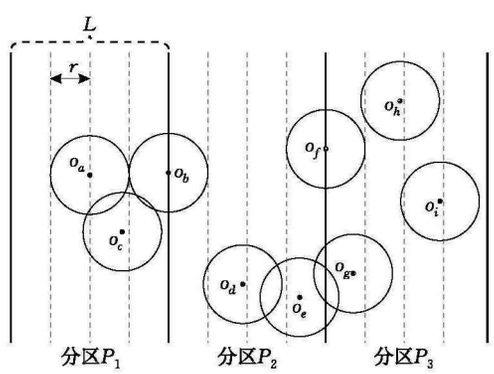


图 4　查询区域按照 L 划分成 3 个分区

5.1.3　边匹配约束检查

扫描线算法的输出只说明了两个对象圆的投影距离在$[0,d_{\max}]$之间,而且扫描线算法是将分区内所有符合查询关键字约束的对象一起作为输入,所

以仍需要对扫描线算法输出的相交对象对进行边匹配条件核查.给定查询 S,对于扫描线算法输出的相交对象对(o_a,o_b)依次进行以下的约束检查.

(1)关键字约束核查:扫描查询 S 中的边,如果存在$(v_i,v_j)\in\mathcal{E}$,使得 o_a 和 o_b 分别与顶点 v_i 和 v_j 匹配,那么 o_a 与 o_b 构成的边满足 S 中的一条边的关键字约束,继续后面的核查计算;否则 o_a 与 o_b 不能构成满足查询要求的边,将从结果集中删除.

(2)距离约束核查:计算 o_a 与 o_b 的距离 d_{ab},如果 d_{ab} 落在边(v_i,v_j)的距离约束$[d^l_{i,j},d^u_{i,j}]$内,则继续步(3);否则将 o_a 与 o_b 的相交记录从结果集中删除.

(3)方向约束核查:计算 o_a 与 o_b 的方向 θ_{ab},如果 θ_{ab} 落在边(v_i,v_j)的方向约束$[\theta^l_{i,j},\theta^u_{i,j}]$内,则 o_a 与 o_b 满足 S 的边匹配要求;否则将 o_a 与 o_b 的相交记录从结果集中删除.

最终查询区域的所有符合边匹配约束的对象对为所有分区结果的并集,即$\mathcal{R}=\bigcup_{i-1}^{x}\mathcal{R}_{P_i}$.图 4 中各个分区执行扫描线算法的结果$\mathcal{R}_{P_1}$,$\mathcal{R}_{P_2}$ 和$\mathcal{R}_{P_3}$,经过上述边匹配约束核查(假定相交对象满足关键字和方向约束),最终边匹配结果集$\mathcal{R}=\bigcup_{i-1}^{3}\mathcal{R}_{P_i}=\{o_a:\{o_b,o_c\},o_b:\{o_c\},o_g:\{o_e\},o_d:\{o_e\}\}$.

5.2 基于子图同构匹配的聚合匹配计算

基准方法通过枚举 S 所有边匹配集合的聚合匹配组合,并核查各个组合是否满足 S 的聚合匹配要求来获得最终的查询结果.该方法没有考虑对象边的实际空间连接,将会浪费较多的计算来排除根本不能形成合理结果的组合.查询 S 可表示为关键字对象及其关系的查询图$\mathcal{G}_s$;而集合$\mathcal{R}$则包含满足 S 查询约束的对象及其相交关系,可表示为对象连接图$\mathcal{G}_{obj}$.因此,可采用"过滤-验证"的策略来优化聚合匹配计算,即首先在对象连接图$\mathcal{G}_{obj}$上寻找与查询图$\mathcal{G}_s$结构相同的子图,然后验证各个子图是否满足 S 的关键字,距离和方向约束,最后所有通过验证的子图即为 SSM 查询结果.该方法考虑了对象的实际空间连接关系,可自动地过滤不合理的聚合匹配组合.

本文因此提出基于子图同构匹配的聚合匹配计算.特别地,"过滤"计算部分可建模为子图同构问题(Subgraph Isomorphism Problem)[34].子图同构查询是图数据查询中的基础查询,旨在目标图中寻找与查询图具备相同结构的子图[34-35].假设有查询图$\mathcal{G}_s=(V_S,E_S)$和目标图$\mathcal{G}_{obj}=(V,E)$,子图同构即从$\mathcal{G}_s$到$\mathcal{G}_{obj}$存在函数$f:V_S\rightarrow V$并且$(u,v)\in E_s$使得$(f(u),f(v))\in E$同样成立,函数 f 称为子图同构的一个映射.因此,聚合匹配的"过滤"计算可转化为在对象连接图$\mathcal{G}_{obj}$上搜寻与 S 的查询图$\mathcal{G}_S$结构相同的子图同构匹配问题.近期,Carletti 等人[35]研究者利用大量真实图数据对已有的子图同构匹配算法进行了性能对比测试.结果表明 VF3 算法[25]能够在匹配时间和内存开销之间实现较好的平衡,综合性能表现优异.VF3 算法包含大量的剪枝策略,在大且密集的图数据上仍能快速完成子图同构匹配查询[35].本文的对象连接图$\mathcal{G}_{obj}$可能包含众多的对象点及密集的连接关系(例如,用户输入的关键字较为常见且命中较多的空间文本对象),因此本文采用 VF3 算法来完成子图同构匹配查询.值得一提的是,本文的方法也可采用其他子图同构匹配算法[36]来完成"过滤"阶段的计算,并在其返回结果集上验证是否满足查询 S 的约束来获得最终的 SSM 查询结果.

优化的聚合匹配计算具体执行过程如下:

(1)构造对象连接图$\mathcal{G}_{obj}$:给定对象相交结果集$\mathcal{R}$,以$\mathcal{R}$中出现的所有对象为顶点,对象的相交情况构造边,从而获得全局的匹配对象连接图$\mathcal{G}_{obj}$.需要注意的是经过关键字,距离和方向等约束的过滤和核查,$\mathcal{G}_{obj}$很可能是非连通图,即包含一些子图.

(2)运行子图同构匹配算法:为了提高查询效率,过滤掉$\mathcal{G}_{obj}$中不能形成有效聚合匹配组合的子图(即包含的对象或边的数目不满足 S 的要求).将剩余的对象及其边构成的目标图$\mathcal{G}'$和查询图$\mathcal{G}_s$作为 VF3 算法[25]的输入来求解子图同构查询问题.

(3)验证子图匹配结果:由于图$\mathcal{G}'$保留的边是符合查询 S 在关键字,空间距离和方向三方面的约束,所以"验证"计算只需要核对 VF3 算法返回的子图的每条边能够正确且唯一地与 S 的查询图$\mathcal{G}_s$的边一一映射即可.经过"验证"的所有子图即构成满足空间结构约束 S 的 SSM 查询结果集.

5.3 算法分析

假设对象集合$\mathcal{D}$拥有 $n=|\mathcal{D}|$个空间对象,且所有对象在空间均匀分布.$\mathcal{D}$包含 m 个关键字,而且对象的关键字分布呈均匀分布.此外,假设用户给定的空间结构查询 S 拥有 $k=|\mathcal{V}|$个顶点和 $g=|\mathcal{E}|$条边.基准方法的时间复杂度分析如下:(1)查询 S 的关键字筛选需要扫描所有对象,时间复杂度为$\mathcal{O}(kn)$;(2)边匹配计算枚举所有可能的对象组合,时间复杂度为$\mathcal{O}\left(g\left(\frac{n}{m}\right)^2\right)$,其中$\frac{n}{m}$表示每个关键字的平均匹配对象数目;(3)聚合匹配计算枚举并核查任意 g 个边匹配构成的边组合,由于每个边匹配集大小约

为$\left(\frac{n}{m}\right)^2$以及核查每个聚合匹配组合是否满足查询$S$的计算量为$\mathcal{O}(g^2)$，故聚合匹配计算的时间复杂度为$\mathcal{O}\left(g^2\left(\frac{n}{m}\right)^{2g}\right)$. 因此，基准方法的总时间复杂度约为$\mathcal{O}\left(kn+g\left(\frac{n}{m}\right)+g^2\left(\frac{n}{m}\right)^{2g}\right)$.

本文优化方法的时间复杂度分析如下：(1) 进行基于查询关键字的对象筛选的过程中可同时将所有命中的对象分配到不同分区，时间复杂度为$\mathcal{O}(kn)$；(2) 基于分区的扫描线算法仅需要扫描该分区内所有对象的匹配范围圆在纵轴的投影，在各个分区并行执行扫描线算法的时间复杂度为$\mathcal{O}\left(\frac{kn}{zm}\right)$，其中$z$为分区数. 假定分区长度为$L$且宽度为$W$，则分区内可形成的边匹配集合大小为$\frac{2r}{W}\left(\frac{kn}{zm}\right)^2$，那么分区的边匹配核查的时间复杂度为$\mathcal{O}\left(\frac{2gr}{W}\left(\frac{kn}{zm}\right)^2\right)$，其中$r(\ll W)$为对象匹配范围圆的半径；(3) 基于子图同构匹配的聚合匹配计算构造的对象连接图$\mathcal{G}_{obj}$最多包含$\frac{kn}{m}$个顶点，则运行 VF3 算法最坏情况的时间复杂度为$\mathcal{O}\left(\left(\frac{kn}{m}\right)^3\right)$[25]. 对 VF3 算法的匹配子图进行验证的计算量$\mathcal{O}(g^2)$为常数，可忽略. 综上，优化方法的总时间复杂度约为$\mathcal{O}\left(kn+\frac{kn}{zm}+\frac{2gr}{W}\left(\frac{kn}{zm}\right)^2+\left(\frac{kn}{m}\right)^3\right)$. 显然，优化方法的时间复杂度相比基准方法有显著改善，即从$\mathcal{O}(n^{2g})$降低为最坏情况的$\mathcal{O}(n^3)$.

本文优化方法的空间复杂度分析如下：边匹配计算的关键字筛选和对象分区的空间复杂度为$\mathcal{O}(n)$，扫描线算法的存储开销与对象圆相交数相关，即为$\mathcal{O}\left(\frac{2r}{zW}\left(\frac{kn}{m}\right)^2\right)$；聚合匹配计算的对象连接图$\mathcal{G}_{obj}$最多包含$\frac{kn}{m}$个顶点和$\frac{2r}{zW}\left(\frac{kn}{m}\right)^2$条边，故 VF3 算法的空间复杂度为$\mathcal{O}\left(\frac{kn}{m}+\frac{2r}{zW}\left(\frac{kn}{m}\right)^2\right)$[25]. 因此，优化方法总的空间复杂度为$\mathcal{O}\left(n+\frac{kn}{m}+\frac{4r}{zW}\left(\frac{kn}{m}\right)^2\right)$.

6　实验验证与结果分析

6.1　实验设置

本节实验将在真实的大规模空间文本数据集上测试对比分析不同算法的 SSM 查询性能.

对比算法. 本文首次提出并定义空间结构匹配(SSM)查询问题，当前尚不存在有效解决该问题的算法. 因此，本文提出的优化方法将与第 4 节的基准方法(记为 Baseline)进行对比. 为了提高计算效率，基准方法也采用分区策略来减少无效的边匹配计算. 此外，Fang 等人[17]为解决空间模式匹配(SPM)查询问题所提出的 MPJ 算法[18]和 MSJ 算法[19]经过适当调整也能用于解决 SSM 问题，即 MPJ 和 MSJ 算法在执行 SPM 查询的过程中增加对对象对(o_a,o_b)在方向约束层面的核查. 连接不同边以完成聚合匹配时，MPJ 算法优先聚合空间关键字中数量较少的边，因此本文新增对比方法 MPJR 算法采用随机的顺序聚合不同的边. MPJR，MPJ 和 MSJ 三个算法都采用 IR 树对数据库$\mathcal{D}$中的所有对象进行索引，然后在 IR 树上进行有效的剪枝以快速获得查询结果.

查询模式. 空间结构S主要由顶点数k及顶点之间的边关系确定，每条边附属的空间距离和方向约束可再由用户按需定义. 对于每种k的设置，变换顶点的边关系以枚举覆盖常见的拓扑结构. 本文将k设置为 3，4，5，6 或 7，并将这些顶点构成的常见 15 种拓扑关系图设为本文实验的查询模式，见图 5. 每一种查询模式包含指定数目的顶点(即关键字对象)，而每种查询模式里每条边的距离和方向范围都是随机产生. 假设参数$d_{\max}$为查询样例最大的距离约束，参数α为方向误差容忍度. 为了保证查询结果的准确性和可验证性，本节实验按以下步骤准备查询样例.

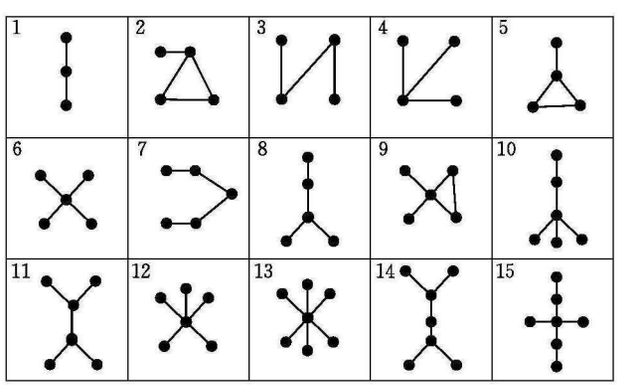


图 5　空间结构匹配查询模式

(1) 给定图 5 中的一种包含k个查询关键字对象的 SSM 查询模式及参数$d_{\max}$，从数据集的关键字集合中随机选择k个不同关键字并设定边的距离约束为$(0,d]$(其中d为$(0,d_{\max})$之间的随机数)，从而构造模式样例$\mathcal{Q}$；

(2) 利用 MSJ 算法[19]在数据集$\mathcal{D}$中查询样例$\mathcal{Q}$;

(3) 在查询结果中随机选择一个匹配结果 R 来构造 SSM 查询样例 S. 具体地,S 以 R 中对象的关键字作为查询关键字并保留关键字对象之间的连接关系;对于 R 中的每条边 $e=(o_a,o_b)$,假定其长度为 d_e,那么 S 中与该边对应的边 (v_i,v_j) 的距离约束设定为下限 $d^l_{i,j}=0.8\times d_e$,上限 $d^u_{i,j}=\{1.2\times d_e,d_{\max}\}$;以边 e 的对象 o_a 和 o_b 之间的方向 θ_e 来设定方向约束的下限 $\theta^l_{i,j}=\theta_e-\alpha$ 和方向约束上限 $\theta^u_{i,j}=\theta_e+\alpha$.

按上述方法构造的查询样例 S 能够保证数据集中确实存在正确的 SSM 查询结果,进而间接验证各个算法的准确性. 对于每个数据集,在给定 $d_{\max}$ 的情况下为图 5 所示的每个查询模式依据上述步骤各生成 10 个查询样例,因此每个数据集将产生 150 个查询样例. 特别地,对于查询模式最大距离约束 $d_{\max}$ 对算法查询性能影响的实验(见第 6.2.4 节),以图 5 中查询对象数为 $k=5$ 的 4 种查询模式(即编号6~9)的 5 种最大距离约束 $d_{\max}$(见表 2),各生成 10 个查询样例,为每个数据集共产生 200 个查询样例.

表 2 实验参数设置

参数	$d_{\max}$/m	$L/(\times r)$	k	$\alpha/(°)$
参数值	100,200,300,400,500	2,4,6,8,10,20,30,40,50	3,4,5,6,7	5,10,15,20,25
默认值	300	10	5	15

评价指标. 对于给定数据集,以完成所有查询样例的平均查询时间来评估算法的查询效率. 本文定义查询准确率来评估算法的查询有效性. 基准方法暴力枚举所有的边匹配和聚合匹配组合,因而不会遗漏任何有效结果. 本文因此将基准方法的查询结果集将视为查询 S 的完整且准确的结果集(即 ground truth),记为 G. 假设待评估算法的结果集为 R,那么该算法的查询准确率定义为 $acc=\frac{|G\cap R|}{|G\cup R|}\times 100\%$.

数据集. 本文采用 4 个国际大城市的 POI 数据集来测试各个算法的 SSM 查询性能. 这些数据集包含城市内 POI 对象的经纬度坐标以及类型说明. 特别地,北京和上海的 POI 数据集来源于北京大学城市与环境学院地理数据平台①,数据集中每个 POI 对象包含 3 个级别的类型描述信息. 由于 POI 对象的一级类型描述过于宽泛而三级类型描述粒度过细(如具体到 POI 实际名称),故本文采用 POI 对象的二级类型描述作为其关键字标签. 东京和伦敦的 POI 数据集来自 SLIPO 项目②,由于数据集中 POI 的类型描述粒度适当,本文直接采用 POI 对象的类型说明作为其关键字标签. 表 3 显示了各个 POI 数据集的统计信息. 北京和上海数据集较大,拥有超过 50 万个 POI 对象;东京和伦敦数据集相对较小,各自拥有约 12 万个和约 9 万个的 POI 对象.

表 3 空间文本数据集统计信息

数据集	POI 对象数量	POI 类型数量
北京	517585	125
上海	577245	126
东京	125052	203
伦敦	94356	204

实验环境. 本节所有算法均使用 Python3 编程实现,所有实验均运行在配备有 Intel Core i5-6500 3.20 GHz CPU 和 16 GB 内存的 Windows 10 机器上. 表 2 显示了本节实验的主要参数及默认设置. 参数 $d_{\max}$ 表示查询样例 S 中边长上限的最大约束;L 表示分区大小,其中 $r=\frac{d_{\max}}{2}$ 是对象匹配范围圆的半径;k 表示查询样例 S 中涉及的关键字对象数目;α 是 SSM 查询对于边的方向误差容忍度.

6.2 实验结果与分析

本节实验在不同参数设置下对比分析不同算法的查询准确性和查询效率. 以下实验中,非研究参数将设置为表 2 中的默认值.

6.2.1 分区大小 L 对本文算法的影响

在第 5.1.2 节中,本文提出分区策略来提高扫描线算法在大的查询区域的查询效率. 图 6 比较了不同分区大小 L 对本文算法在不同数据集上的平均查询时间和存储开销两方面的性能表现,其中 L 的大小设置为对象匹配范围圆半径 r 的倍数. 在四个数据集上,本文算法的平均查询时间随着 L 的增大而先变小后变大(见图 6(a)). 当 L 过小时(如 $<10\times r$),小分区使得处在分区边界的对象圆变多,需要重复地在两个分区对同一个对象的相交情况进行计算,越多的冗余计算使得查询时间变长. 而当 $L>10\times r$ 时,更大的分区会带来更多误判,进而带来额外的边匹配核查计算开销. 图 6(b)显示的结果也证实小分区因为冗余的数据存储而导致更大的存储开销. 当 $L\geq 10\times r$ 时,更大的分区可减少冗余数据存储,总体的存储开销趋于稳定. 图 6 显示本文算法在分区大小为 $L=10\times r$ 时能够获得较好的查询

① http://geodata.pku.edu.cn
② http://download.slipo.eu/results/osm-to-csv/

效率及存储开销，具有更好的综合查询性能. 因此，以下对比实验本文算法默认采用 $L=10\times r$ 的设置.

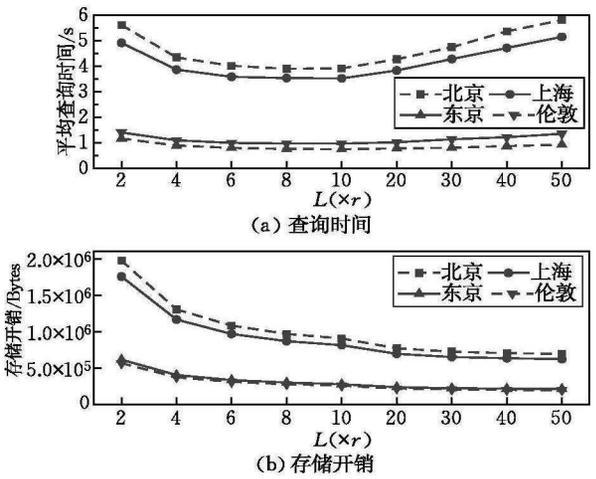


图 6　分区大小 L 对本文算法查询性能的影响

6.2.2　查询算法有效性对比分析

表 4 比较了 SSM 与 SPM[17] 查询技术针对空间结构查询样例的平均查询结果数和查询准确率. SSM 查询从关键字对象，距离和方向等空间约束来搜索匹配用户需求的对象组合，而 SPM 查询[18-19]忽视了对象之间的方向关系，将返回大量无效结果. 表 4 显示 SSM 的结果集远小于 SPM 的结果集，实验结果表明 SSM 相比 SPM 能返回更为精简有效的结果. 由于本文的 SSM 查询方法采取了保守但有效的剪枝策略，能够快速过滤无效对象但又不会遗漏任何有效的边匹配或聚合匹配组合，因此可保证返回的结果与基准方法完全一致，达到 100% 的准确率. 虽然 SPM 查询的结果集同样包含所有的有效结果，但是无效结果占绝大多数，使得其查询准确率≤1%.

表 4　不同查询技术的平均查询结果数和准确率

	查询结果数量		查询准确率/%	
	SSM	SPM	SSM	SPM
北京	14	1558	100	0.9
上海	21	2642	100	0.8
东京	19	2001	100	1.0
伦敦	13	1903	100	0.7

图 7 可视化对比了 SSM 和 SPM 的部分查询结果. 给定图 5 中模式 4，图 7(a)显示了一个实际查询样例在上海数据集上 SSM 和 SPM 查询结果的可视化对比：SSM 查询结果很好地匹配了查询需求，满足对象之间距离和方向的约束；而 SPM 查询结果仅仅考虑了对象之间的距离约束，整体的空间关系并没有满足用户的要求. 图 7(b)显示了给定图 5 中模式 6 的一个查询样例在伦敦数据集上各个查询技术的结果可视化效果. 同样地，SSM 查询结果相比 SPM 查询结果能更好地匹配用户的空间结构查询约束. 表 4 的统计信息和图 7 的可视化案例反映出 SSM 查询相比 SPM 查询能更好地表示用户的空间查询需求，获得更精简且准确的空间查询结果.

图 7　查询结果可视化对比

6.2.3　查询对象数 k 的影响

为了评估 k 的影响，首先将所有查询样例根据查询对象数目分为五组，即包含 3，4，5，6 或 7 个关键字对象，然后计算各组 SSM 查询样例的平均查询时间，实验结果如图 8 所示(本文算法标记为“Ours”，下同). 总体上，随着查询对象数 k 的增加，各个算法

的查询时间随之增长.这是因为查询对象数与边的数目是正相关的,越多的边意味着越多的边匹配计算和越复杂的聚合匹配计算,使得 SSM 查询的处理时间随之增长.在所有查询算法中,Baseline 算法的查询效率最低,在北京和上海数据集的查询时间分别在 35～312 s 之间和 19～198 s 之间,远高于其它算法的平均查询时间.因此 Baseline 算法的查询时间在图 8(a)和图 8(b)中未予显示.即使在小规模数据集上,图 8(c)和图 8(d)显示 Baseline 算法的平均查询时间也远高于其他对比算法.

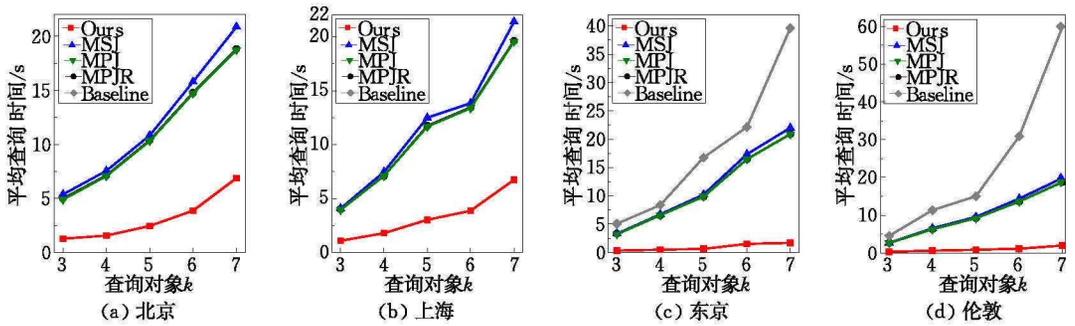


图 8 查询对象数 k 对各个算法查询性能的影响

MPJ 算法通过优化聚合边的匹配顺序[18],其查询性能在大部分情况略优于 MPJR 算法,平均查询时间减少约 3%.相比于 MPJ 和 MPJR 算法,MSJ 算法将首先对查询模式 S 进行优化.主要体现在对有环图的模式 S 进行边距离范围的优化,使 S 中各边的距离区间缩小从而使满足边匹配条件的对象对相应减少,最终降低聚合匹配的计算开销[19].本文统计了各个算法在图 5 中包含环的查询模式 2,5 和 9 的所有查询样例上的查询时间,统计结果如图 9 所示.基准方法的平均查询时间依然远超其他对比算法,故未予显示.由于 SSM 查询已经同时限定对象之间的距离和方向,边的空间结构约束已经较为紧凑,因此 MSJ 算法的优化提升效果不太明显.MSJ 算法在有环查询样例上的查询性能略优于 MPJ 和 MPJR 算法,在所有数据集上的平均查询时间减少约 2%.图 9 显示本文算法仍是平均查询时间最少的方法,相比于 MSJ 算法在查询效率上提升约 4.2～14.7 倍.

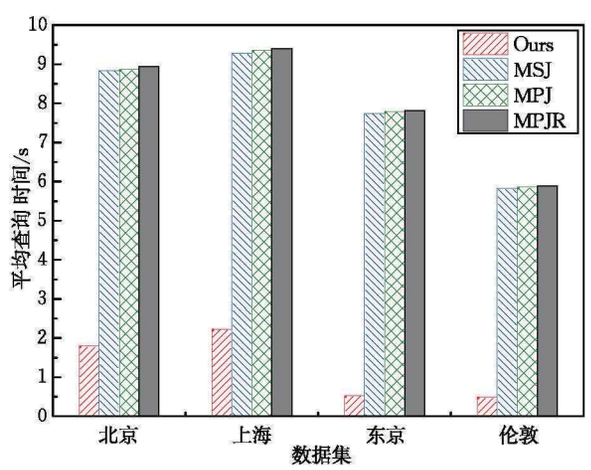


图 9 各个算法在有环查询样例上的性能比较

图 8 和图 9 显示本文算法的查询效率在所有算法中最高,其平均查询时间远低于对比算法.例如,当 $k=5$ 时,本文算法、MSJ、MPJ、MPJR 和 Baseline 在北京数据集的平均查询时间分别为 2.4 s、10.8 s、10.3 s、10.4 s 和 101.9 s.根据图 8 的实验结果,相比于当前最好的算法 MPJ[18] 和 MSJ[19],本文算法在大数据集(即北京和上海数据集)上的平均查询时间快 3.5～4.1 倍,在小数据集(即东京和伦敦数据集)上快 10.3～13.2 倍.

6.2.4 查询模式最大距离约束 d_{max} 的影响

图 10 显示了不同数据集下各个算法在不同最大距离约束 d_{max} 下的查询性能.实验结果表明随着 d_{max} 的增大,不同算法的查询时间随之增长.Baseline 算法在所有数据集上的平均查询时间仍然是最长的,远大于其它算法.例如,Baseline 算法在北京和上海数据集上的平均查询时间分别为 20～220 s 和 22～124 s,远超其他算法的查询时间,故未在图 10(a)和图 10(b)中显示.由于查询 S 的边最大距离约束 d_{max} 的增大,边距离区间也随之变大,满足边匹配要求的对象对也会变多,进而带来更多的计算开销.Baseline 算法的平均查询时间随着 d_{max} 的增大而急速增加,这是因为距离约束的变大使得更多的对象对满足边的距离约束,进而产生更多的边匹配和聚合匹配的遍历计算开销.其它算法的平均查询时间随着 d_{max} 的增大而缓慢增长.

如图 10 所示,本文算法的查询性能在不同 d_{max} 的设置下均显著优于对比算法.在 d_{max} 较小时,本文算法查询效率的优势更为明显.例如,当 $d_{max}=100$ m 时,本文算法在北京和上海数据集的平均查询时间比

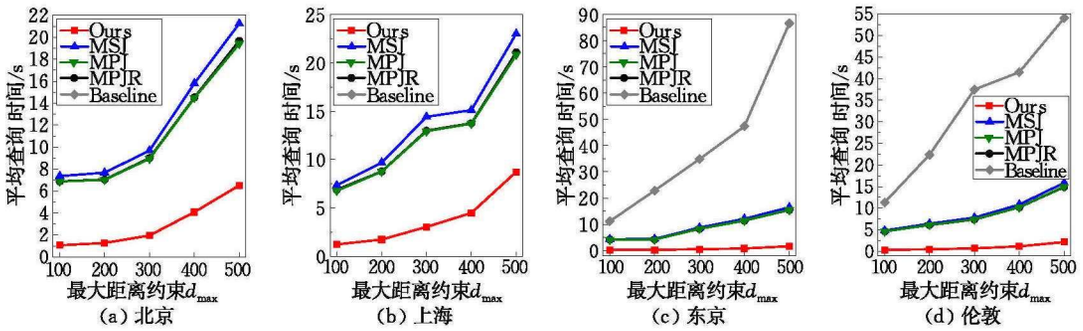


图 10　最大距离约束 d_{max} 对各个算法查询性能的影响

MSJ 和 MPJ 算法快 5.5～6.9 倍，在东京和伦敦数据集的平均查询时间比 MSJ 和 MPJ 算法快 13.7～16.5 倍. 当 $d_{max}=500$ m 时，本文算法的平均查询时间相比 MSJ 和 MPJ 算法，在北京和上海数据集上快 2.3～3.2 倍，而在东京和伦敦数据集上快 7.0～9.9 倍. 上述对比数据表明本文算法比对比算法具有更高效的查询效率，而且非常适合在空间文本数据库中查找小范围的空间结构.

6.2.5　查询模式方向误差容忍度 α 的影响

本对比实验通过 α 改变查询模式 S 中每条边的方向约束区间，从而支持用户在对象之间方向结构的模糊查询. 图 11 显示了不同算法在不同数据集上和不同 α 设置下的查询性能对比. 从图 11 所示的实验结果分析，方向误差容忍度 α 对 Baseline 算法的影响较大，但是对其他算法的查询性能影响较小. 这是因为随着 α 的增大，SSM 查询对边的方向的约束变得更为稀松，Baseline 算法将保留更多的对象对组成的边，进而产生更多的边匹配和聚合匹配的计算开销. 在不同的 α 设置下，Baseline 算法在北京和上海数据集上的平均查询时间分别为 147～185 s 和 128～156 s，远超其他算法，故在图 11(a)和图 11(b)中未予显示. 参数 α 对于其他算法的查询时间性能影响不大，查询性能表现较为平稳. 这可能是因为这些算法利用空间剪枝技术已经过滤掉足够多的对象组合边，使得方向约束的宽松并未对总体计算带来太多影响.

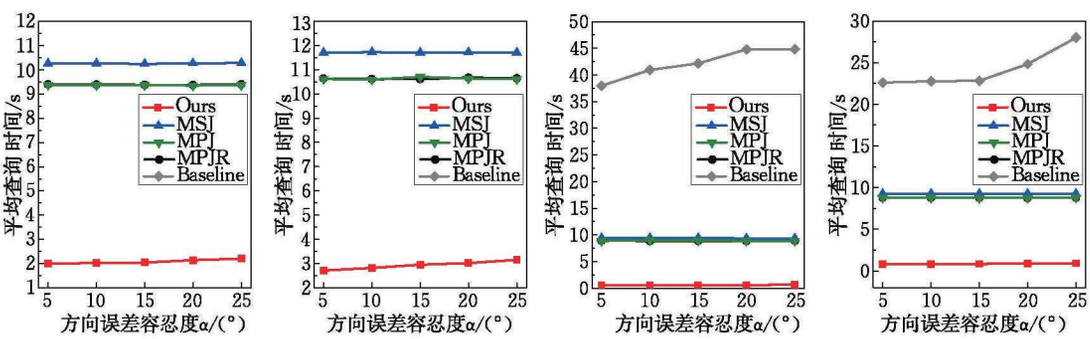


图 11　方向误差容忍度 α 对各个算法查询性能的影响

7　总结与展望

为了支持用户在大规模空间文本数据库中更为精准的查询需求，本文提出并定义了一种新的空间结构匹配(SSM)查询问题. SSM 查询允许用户查找任意关键字对象集合并定义对象之间的距离和方向约束. 为了解决 SSM 问题，本文提出了基于多路连接的基准方法. 为了提高查询效率，本文进一步提出了扫描线算法和分区策略来快速判断对象之间的相交关系，加速边匹配计算. 此外，本文利用对象相交关系构建对象连接图，并提出了基于子图同构匹配的聚合匹配计算，提升了 SSM 查询的效率. 本文算法无需提前建立索引结构，因而适用于在动态数据库中的快速查询. 实验结果表明，本文算法在真实大规模空间文本数据集中的查询效率显著优于对比算法.

后续，将重点考虑对象在方向约束层面的剪枝策略来进一步提高 SSM 查询的性能. 此外，考虑设计一种评分机制来评估 SSM 查询结果与用户的查询偏好属性的契合度并以此评分来向用户推荐匹配结果，从而为用户提供高效的个性化查询服务.

参 考 文 献

[1] Junglas I A, Watson R T. Location-based services. Communications of the ACM, 2008, 51(3): 65-69

[2] Feng K, Cong G, Bhowmick S S, et al. Towards best region search for data exploration//Proceedings of the ACM SIGMOD International Conference on Management of Data. San Francisco, USA, 2016: 1055-1070

[3] Zhao P, Zhu H, Liu Y, et al. Where to go next: A spatio-temporal gated network for next POI recommendation//Proceedings of the AAAI Conference on Artificial Intelligence. Hawaii, USA, 2019: 5877-5884

[4] Cenamor I, de la Rosa T, Núez S, Borrajo D. Planning for tourism routes using social networks. Expert Systems with Applications, 2017, 69(1): 1-9

[5] Liu Y, Liu C, Lu X, et al. Point-of-interest demand modeling with human mobility patterns//Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Halifax, Canada, 2017: 947-955

[6] Chen Z, Chen L, Cong G, Jensen C S. Location-and keyword-based querying of geo-textual data: A survey. The VLDB Journal, 2021, 30: 1-38

[7] De Felipe I, Hristidis V, Rishe N. Keyword search on spatial databases//Proceedings of the IEEE 24th International Conference on Data Engineering. Cancun, Mexico, 2008: 656-665

[8] Liu Xi-Ping, Wan Chang-Xuan, Liu De-Xi, Liao Guo-Qiong. Survey on Spatial Keyword Search. Journal of Software, 2016, 27(2): 329-347(in Chinese)
(刘喜平, 万常选, 刘德喜, 廖国琼. 空间关键词搜索研究综述. 软件学报, 2016, 27(2): 329-347)

[9] Cao X, Chen L, Cong G, et al. Spatial keyword querying//Proceedings of the International Conference on Conceptual Modeling. Berlin, Germany, 2012: 16-29

[10] Cong G, Jensen C S, Wu D M. Efficient retrieval of the top-k most relevant spatial Web objects. Proceedings of the VLDB Endowment, 2009, 2(1): 337-348

[11] Wu D, Yiu M L, Cong G, Jensen C S. Joint top-k spatial keyword query processing. IEEE Transactions on Knowledge and Data Engineering, 2011, 24(10): 1889-1903

[12] Zhang D, Chan C Y, Tan K L. Processing spatial keyword query as a top-k aggregation query//Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval. Queensland, Australia, 2014: 355-364

[13] Guo T, Cao X, Cong G. Efficient algorithms for answering the m-closest keywords query//Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data. Melbourne, Australia, 2015: 405-418

[14] Chan H K H, Long C, Wong R C W. On generalizing collective spatial keyword queries. IEEE Transactions on Knowledge and Data Engineering, 2018, 30(9): 1712-1726

[15] Cao X, Cong G, Jensen C S, Ooi B C. Collective spatial keyword querying//Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data. Athens, Greece, 2011: 373-384

[16] Long C, Wong R C W, Wang K, Fu A W C. Collective spatial keyword queries: A distance owner-driven approach//Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data. New York, USA, 2013: 689-700

[17] Li Y, Fang Y, Cheng R, Zhang W. Spatial pattern matching: A new direction for finding spatial objects. SIGSPATIAL Special, 2019, 11(1): 3-12

[18] Fang Y, Cheng R, Cong G, et al. On spatial pattern matching//Proceedings of the 34th IEEE International Conference on Data Engineering. Paris, France, 2018: 293-304

[19] Fang Y, Li Y, Cheng R, et al. Evaluating pattern matching queries for spatial databases. The VLDB Journal, 2019, 28(5): 649-673

[20] Zhang D, Chee Y M, Mondal A, et al. Keyword search in spatial databases: Towards searching by document//Proceedings of the IEEE 25th International Conference on Data Engineering. Shanghai, China, 2009: 688-699

[21] Liu J, Deng K, Sun H, et al. Clue-based spatio-textual query. Proceedings of the VLDB Endowment, 2017, 10(5): 529-540

[22] Feng K, Cong G, Jensen C S, Guo T. Finding attribute-aware similar regions for data analysis. Proceedings of the VLDB Endowment, 2019, 12(11): 1414-1426

[23] Fan J, Li G, Zhou L, et al. Seal: Spatio-textual similarity search. Proceedings of the VLDB Endowment, 2012, 5(9): 824-835

[24] Schnaiberg J, Riera J, Turner M G, Voss P R. Explaining human settlement patterns in a recreational lake district: Vilas County, Wisconsin, USA. Environmental Management, 2002, 30(1): 24-34

[25] Carletti V, Foggia P, Saggese A, Vento M. Challenging the time complexity of exact subgraph isomorphism for huge and dense graphs with VF3. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2018, 40(4): 804-818

[26] Chen L, Cong G, Jensen C S, Wu D. Spatial keyword query processing: An experimental evaluation. Proceedings of the VLDB Endowment, 2013, 6(3): 217-228

[27] Wu D M, Yiu M L, Jensen C S, Cong G. Efficient continuously moving top-k spatial keyword query processing//Proceedings of the 2011 IEEE 27th International Conference on Data Engineering. Hannover, Germany, 2011: 541-552

[28] Huang W H, Li G L, Tan K L, Feng J H. Efficient safe-region construction for moving top-k spatial keyword queries//Proceedings of the 2012 ACM International Conference on Information & Knowledge Management. Maui, USA, 2012: 932-941

[29] Guo L, Shao J, Aung H H, Tan K L. Efficient continuous top-k spatial keyword queries on road networks. GeoInformatica, 2015, 19(1): 29-60

[30] Zheng BL, Zheng K, Xiao X H, et al. Keyword-aware continuous kNN query on road networks//Proceedings of the 2016 IEEE 32nd International Conference on Data Engineering. Helsinki, Finland, 2016: 871-882

[31] Chen L, Shang S, Yang C, Li J. Spatial keyword search: A survey. GeoInformatica, 2020, 24(1): 85-106

[32] Liu X P, Chen L, Wan C X. LINQ: A framework for location-aware indexing and query processing. IEEE Transactions on Knowledge and Data Engineering, 2015, 27(5): 1288-1300

[33] Mamoulis N, Papadias D. Multiway spatial joins. ACM Transactions on Database Systems, 2001, 26(4): 424-475

[34] Ullmann J R. An algorithm for subgraph isomorphism. Journal of the ACM, 1976, 23(1): 31-42

[35] Carletti V, Foggia P, Greco A, et al. Comparing performance of graph matching algorithms on huge graphs. Pattern Recognition Letters, 2020, 134: 58-67

[36] Sun S, Luo Q. In-memory subgraph matching: An in-depth study//Proceedings of the ACM SIGMOD International Conference on Management of Data. Portland, USA, 2020: 1083-1098

LIU Zhi-Dan, Ph. D., assistant professor, M. S. supervisor. His research interests include urban computing and Internet of Things.

LIN Wei-Xin, M. S. candidate. His main research interests include spatial data management and analysis.

WU Kai-Shun, Ph. D., professor, Ph. D. supervisor. His research interests include Internet of Things and mobile computing.

Background

The increasing popularity of location-based services (LBS) has enabled the accumulation of a huge volume of spatial-textual data. These data are usually associated with geographic locations and text descriptions, e. g., point of interest (POI) categories or users' reviews. The availability of such large-scale spatial-textual data can facilitate our daily life by inspiring a variety of intelligent applications, such as POI recommendations, tourism planning, and human mobility analysis.

To support these LBS applications, various techniques have been proposed to query and search on spatial-textual databases. Previously, much attention was paid on the research of spatial keyword query (SKQ), which returns a set of objects such that they locate around the query location and meanwhile cover all the query keywords with a certain cost. Recently, spatial pattern matching (SPM) query has been proposed to allow users to query a set of objects, which form a specific spatial pattern, from the dataset. The spatial pattern defines the distance constraints between any two keyword objects.

Although SKQ and SPM could find the objects a user may concern, they are insufficient on accurately capturing the user's intention and usually return a huge number of irrelevant results, resulting in the poor query efficiency. Therefore, in this paper we study a novel query problem—Spatial Structure Matching (SSM) query, which allows a user to provide a set of keyword objects and meanwhile define the constraints on both distance and direction for any two concerned keyword objects. To answer the SSM query, this paper firstly presents a multi-way join based baseline method. Due to its high computation overhead, we further devise a series of optimizations to improve the baseline. Specifically, we propose a sweep-line algorithm to efficiently filter out objects that cannot meet the distance constraints. We then exploit these retained objects to construct a object-connection graph, and transform the SSM query into a subgraph isomorphism problem by treating the object-connection graph as the target graph and the spatial structure of the SSM query as the pattern graph. We adopt a fast subgraph matching solver, i. e., VF3, to derive the final query results. Extensive experiments based on four large real-world spatial-textual datasets demonstrate that our proposed approach can significantly outperform the compared approaches, by reducing the average query processing time by at least 3×.

This work is supported in part by the National Natural Science Foundation of China (Nos. 62172284, 61872248, and U2001207) and the Natural Science Foundation of Guangdong Province (Nos. 2020A1515011502 and 2017A030312008).