CPM-MCHM：一种基于极大团和哈希表的空间并置模式挖掘算法

张绍雪 王丽珍 陈文和
(云南大学信息学院 昆明 650504)

摘 要 空间并置（co-location）模式挖掘是指在大量的空间数据中发现一组空间特征的子集，这些特征的实例在地理空间中频繁并置出现. 传统的空间并置模式挖掘算法通常采用逐阶递增的挖掘框架，从低阶模式开始生成候选模式并计算其参与度（空间并置模式的频繁性度量指标）. 虽然这种挖掘框架可以得到正确和完整的结果，但是带来的时间和空间开销非常大. 此外传统方法对于空间并置模式的最小频繁性阈值较为敏感，当最小频繁性阈值改变时整个挖掘过程需要重新进行. 因此，本文提出一种基于极大团和哈希表的空间并置模式挖掘算法 CPM-MCHM (Co-location Pattern Mining based on Maximal Clique and Hash Map) 来发现完整并且正确的频繁空间并置模式. CPM-MCHM 算法不仅避免逐阶候选-测试框架带来的巨大开销问题，还降低了算法对最小频繁性阈值的敏感. 首先，采用基于位运算的分区 Bron–Kerbosch 算法生成给定空间数据集的所有极大团，并将其存储在哈希表中. 然后，提出一种两阶段挖掘框架计算所有模式的参与度并过滤所有频繁空间并置模式. 最后，在真实和合成数据集上进行了大量的对比实验. 与经典的传统算法和近两年内学者提出的两种算法相比，当实验数据的规模达到 20 万实例数时，本文提出的 CPM-MCHM 算法的挖掘时间和空间耗费分别降低了 90%和 70%以上，当实验数据量进一步加大时 CPM-MCHM 算法的优势更加明显.

关键词 空间数据挖掘；空间并置模式；两阶段挖掘框架；极大团；哈希表
中图法分类号 TP18 **DOI 号** 10.11897/SP.J.1016.2022.0526

CPM-MCHM: A Spatial Co-location Pattern Mining Algorithm Based on Maximal Clique and Hash Map

ZHANG Shao-Xue WANG Li-Zhen TRAN Van-Ha
(*School of Information Science and Engineering, Yunnan University, Kunming* 650504)

Abstract Spatial co-location pattern mining refers to the discovery of a set of spatial features in large spatial data sets, and instances of these features frequently appear together in the geographic space. The traditional co-location pattern mining algorithms usually employ an incremental mining framework, generating high-size candidate patterns from the low-size prevalent patterns and compute their participation index (prevalence metrics of the co-location pattern). The participation indexes of candidate patterns are compared with a minimum prevalence threshold given by users to filter prevalent co-location patterns. This iterative process continues to be executed size-by-size, all prevalent co-location patterns are generated completely. Although this mining framework can yield a correct and complete mining result, the cost of

收稿日期：2020-08-31；在线发布日期：2021-07-26. 本课题得到国家自然科学基金项目（61966036, 61662086）、云南省创新团队项目（2018HC019）、云南大学研究生科研创新基金项目（2020315）资助. **张绍雪**，硕士研究生，主要研究领域为空间数据挖掘. E-mail: zhangshaoxue@mail.ynu.edu.cn. **王丽珍**（**通信作者**），博士，教授，博士生导师，中国计算机学会（CCF）高级会员（21619S），主要研究领域为空间数据挖掘、交互式数据挖掘、大数据分析等. E-mail: lzhwang@ynu.edu.cn. **陈文和**，博士研究生，主要研究领域为空间数据挖掘.

execution time and memory space is very expensive. In addition, this mining framework is very sensitive to the minimum prevalence threshold used to filtering prevalent co-location patterns. If the minimum prevalence threshold is changed, the whole mining process needs to be restarted. Since the row instances supporting the prevalence of a co-location pattern are a clique relation under the neighbor relationship, all row instances of the co-location patterns are included in the maximal cliques of the spatial data set, thus they can be used to generate all row instances of all spatial co-location patterns. Therefore, under the premise of discovery of complete and correct prevalent spatial co-location patterns, a Co-location Pattern Mining algorithm based on Maximal Clique and Hash Map (CPM-MCHM) is proposed to address the above shortcomings. The CPM-MCHM algorithm not only avoids the problem of the expensive running time posed by the size-by-size candidate-test framework, but also reduces the sensitivity of the algorithm to the minimum prevalence threshold. Firstly, a bit operation-based and partition Bron-Kerbosch algorithm is presented to enumerate all the maximal cliques of an input spatial data set. The performance of this enumeration maximal clique algorithm is examined by experiments to prove that it can quickly and accurately enumerate all maximal cliques. Secondly, these maximal cliques are compressed into the hash map, by utilizing the characteristic of the hash map to speed up collecting row instances of all co-location patterns. Finally, using the generated hash map and downward closure property of the participation index, a two-stage mining approach is proposed. The first stage generates clique candidate patterns and collects their participation instances, and then calculates the participation indexes of these candidate patterns. The second stage calculates the participation indexes of all remaining patterns based on the participation instances of features in the clique candidates of the first stage. Through the two stages, the participation indexes of all patterns can be obtained efficiently, thereby all prevalence co-location patterns can be effectively filtered. At the end of the paper, a lot of comparative experiments were conducted on real and synthetic data sets. Compared to a classical algorithm and two algorithms proposed in the past two years, when the scale of the experimental data reaches 200,000 spatial instances, the mining time and space cost of the proposed CPM-MCHM algorithm were reduced by more than 90% and 70% respectively, and the CPM-MCHM algorithm advantage of increasing the experimental data volume was more obvious.

Keywords　spatial data mining; spatial co-location pattern; two-stage mining framework; maximal clique; hash map

1　引　言

空间数据是在现实生活中具有重要意义的一类数据，它有着比一般数据更加丰富和复杂的语义信息，因此对于空间数据的挖掘也比事务数据挖掘更复杂[1]. 空间数据挖掘技术旨在从空间数据库中挖掘出大量潜在且有用的信息，已经在城市规划[2-3]、环境科学[4]、地球科学[5]等领域得到广泛的应用. 空间并置模式是空间特征的子集，它们的实例在空间中频繁关联[6]. 例如，社区附近经常出现超市，植物茂盛的地方往往有蚊虫积聚等.

如图 1 所示，传统的空间并置模式挖掘框架采用逐阶候选-测试框架[7]，在计算空间实例之间的邻近关系的基础上，基于 k–1 阶频繁模式生成 k(k>2) 阶候选模式，然后搜集每个候选模式的表实例，通过给定的最小频繁性阈值来测试候选模式是否为频繁模式，通过逐次迭代最终生成所有频繁空间并置模式.

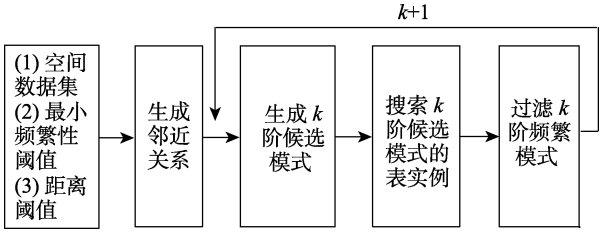


图 1　传统的空间并置模式挖掘框架

尽管通过上述挖掘框架可以正确地生成所有的频繁空间并置模式，但是这种逐阶候选-测试的挖掘

框架存在着几个明显的缺点.

首先，逐阶递增挖掘过程的 k 阶候选模式需要从 k–1 阶频繁模式中产生，所以挖掘过程中需要生成大量的候选模式. 例如，当有 10^3 个空间特征时，基于逐阶候选-测试挖掘框架的算法会生成超过 10^5 个 2 阶候选模式. 另外，在最坏情况下，发现 100 阶频繁模式共需要产生 2^{100}-1 个候选模式.

其次，逐阶候选-测试挖掘框架需要逐阶生成和搜索每个候选模式的表实例，当数据集十分密集或实例数量较大时，逐阶识别候选模式的表实例非常耗费时间和空间. 比如，在基于完全连接的空间并置模式挖掘算法（join-based）[8]中需要将两个 k–1 阶频繁模式的表实例进行全连接来识别出 k 阶候选模式的表实例，同时判断这两个表实例的最后一个实例是否存在空间上的邻近关系，所以还需遍历 2 阶频繁模式的表实例. 又比如基于无连接的空间并置模式挖掘算法（joinless）需要遍历星型邻居和 k-1 阶频繁模式的表实例来识别出 k 阶候选模式的表实例. 逐阶的识别表实例，造成大量的重复计算，并且 2 阶模式、k–1 阶模式和当前正在计算的 k 阶模式的表实例所占用的空间一直无法释放，造成了大量的空间消耗.

最后，逐阶候选-测试挖掘算法对最小频繁性阈值敏感，当用户更改最小频繁性阈值时，算法一般需要重新启动挖掘任务. 原因如下：首先，因为表实例数量巨大，逐阶挖掘算法一般仅留存 2 阶模式表实例和当前正在计算模式的前一阶模式的表实例；其次，k 阶候选模式是基于 k–1 阶频繁模式生成的，最小频繁性阈值的改变可能导致低阶频繁模式集的改变，然后导致对应高阶候选模式的改变；最后，传统逐阶挖掘算法候选表实例的计算大多基于对应的低一阶模式表实例进行. 因此当用户改变最小频繁性阈值时，算法一般需要从最初的步骤开始执行所有挖掘过程，带来很多挖掘步骤的重复计算，比如重新计算候选模式的表实例和参与度.

针于传统的逐阶候选-测试挖掘框架存在的问题，本文提出一种基于极大团和哈希表的空间并置模式挖掘算法. 由于支持空间并置模式频繁性的行实例是空间邻近关系下的团关系[9-10]，模式的所有行实例都包含在空间数据集的极大团中，故可以使用空间数据集产生的极大团来发现所有空间并置模式的表实例. 特别地，本文不将极大团转换为事务类型数据，而是存储到哈希表中，通过搜索哈希表分两阶段来计算出所有模式的参与度，有效提高了算法的挖掘效率.

本文的主要贡献包括四个方面:

（1）利用极大团物化空间实例间的邻近关系，基于极大团计算所有空间并置模式的参与度，解决了传统逐阶候选-测试挖掘框架中，对最小频繁性阈值敏感的问题；

（2）将极大团保存在哈希表中，直接通过关键字找到所需极大团，并成批计算模式的表实例. 同时，基于哈希表存储极大团较传统物化空间邻近关系的方法，有效节省了存储空间. 因此，基于极大团和哈希表的两阶段挖掘框架解决了逐阶候选-测试挖掘框架的重复计算和重复存储问题，提高了算法的挖掘效率；

（3）基于生成的哈希表和空间并置模式的向下闭合性质，设计了一种有效的两阶段挖掘框架来发现所有频繁的空间并置模式.

（4）在合成和真实数据上验证了提出的算法能有效提高空间并置模式挖掘的效率.

本文的剩余部分组织如下：第 2 节概述相关工作；第 3 节介绍空间并置模式挖掘的基本概念；第 4 节详细描述本文提出的基于极大团和哈希表的空间并置模式挖掘算法并进行算法分析；第 5 节给出实验结果和分析；第 6 节总结本文的工作.

2 相关工作

Yoo 在文献[7]中提出了部分连接（partial-join）和无连接（joinless）的空间并置模式挖掘算法. partial-join 是基于团划分模型物化空间邻近关系以减少连接计算，但需要额外存储被团切断的邻近关系. joinless 运用星型划分模型，用实例查找机制代替了实例连接操作，降低了时间消耗，但是在有些情况下过滤步骤可能会很耗时. 文献[11]对无连接的空间并置模式挖掘算法进一步研究，提出了基于前缀树的 iCPI-tree，将空间邻近关系组织成一种树形结构，所有表实例都能通过前缀树快速生成，但存储前缀树的内存开销是制约该类算法的关键. 文献[12]提出了基于 GPU 的空间并置模式挖掘算法，该算法假设 iCPI-tree 在利用 GPU 进行挖掘之前就已经构造完成. 文献[13]消除了文献[12]中算法的限制，提出了 iCPI-tree 构造的并行化方法. 文献[14]利用聚类和重叠图技术来挖掘空间并置模式，算法视每个特征为一个单独层，并将每层中存在的点进行聚类，不同层的重叠实例组成空间并置模式. 文献[15]提出了一种基于密度的空间并置模式挖掘算法，将空间分区，使用密度来测量分区内可能有多少行实例，优先识别密集区域内的空间实例. 文献

[9]提出了基于团的空间并置模式挖掘算法，设计了两种算法来计算空间数据集中的团. 文献[10]提出了基于重叠极大团划分的方法，将空间数据划分为重叠的极大团，然后通过组合操作得到所有的候选模式及其表实例. 文献[9]和文献[10]提出的两种算法虽然避免了逐阶递增生成候选模式的过程，但是在运行过程中需要消耗大量的空间. 文献[16]提出了一种基于网格的空间并置模式挖掘算法，将空间划分为许多小单元，利用并行编程来改进算法效率，每个进程承担着几个小单元的计算任务，但是单元的边长很难设置. 文献[17]提出了一种基于网格和 GPU 的空间并置模式挖掘算法. 文献[18]提出在 MapReduce 框架上并行挖掘空间并置模式的算法，多个机器独立地完成挖掘任务，降低了机器之间的通信成本，能有效处理海量数据，但是这种并行挖掘方法对执行环境有较高的要求.

枚举极大团是图论中的经典问题，在实践中最成功的是由 Bron 和 Kerbosch 在文献[19]中提出的 Bron-Kerbosch 算法，它是一种高效的深度优先搜索算法，枚举了静态图中的所有极大团. 它通过递归地解决由三组顶点集合(包含在团中的顶点集合，从团中排除的顶点集合，需要确定是否包含在团中的顶点集合)指定的子问题来查找空间中的极大团. Eppstein[20]和 Tomita[21]等众多学者提出了基于 Bron-Kerbosch 的改进算法. 文献[22]在两个方向上提出了四种策略来加快枚举极大团的过程. 文献[23]提出了基于分区的极大团枚举算法，解决了用有限内存处理大图的问题. 极大团的概念也被应用于各个领域，文献[24]提出了从时间图中枚举极大团的算法. 文献[25]提出了从空间数据集中枚举极大团的算法. 文献[26]提出了从不确定图中枚举极大团的算法.

本文提出一种新的空间并置模式挖掘算法，其利用一种高效的极大团枚举算法寻找空间数据集中的所有极大团. 基于极大团的哈希存储，分两阶段挖掘空间数据集中所有频繁空间并置模式，第一阶段计算团候选的表实例和参与度，第二阶段计算剩余所有模式的参与度，两阶段的计算框架较好地解决了传统逐阶候选-测试框架存在的问题.

3　相关概念

本节介绍空间并置模式挖掘的相关定义. 空间特征是指在空间中不同的事物类型（例如，超市，饭店等），空间特征集是空间中不同类型事物的集合，记为 $F = \{f_1, f_2, \ldots, f_n\}$. 空间实例是空间特征在空间中具体位置上的出现，记为 $S = s_1 \cup s_2 \cup \cdots \cup s_n$，$s_i$ 是特征为 f_i 的空间实例集合. R 是 S 上的空间邻近关系，如果将欧式距离用于度量空间邻近关系，则 $R = \{(o_i, o_j) \mid \rho(o_i, o_j) \leqslant min\_dist, o_i \in S, o_j \in S\}$，$\rho$ 表示实例间的欧氏距离，min_dist 表示用户给定的距离阈值. $cl = (o_1, o_2, \ldots, o_k)$为空间中的一组实例，当 cl 中任意两个空间实例均满足空间邻近关系 R 时，称 cl 是一个空间团. 当一个空间团不为任意空间团的子集时，称该空间团为空间极大团[27].

例 1.　图 2 中空间实例集{A.2, B.2, C.3, D.1}中任意两个实例都满足空间邻近关系，并且{A.2, B.2, C.3, D.1}不是任何空间团的子集，所以{A.2, B.2, C.3, D.1}是空间极大团. {A.2, B.2, C.3}是{A.2, B.2, C.3, D.1}的子集，所以{A.2, B.2, C.3}是空间团，不是空间极大团.

定义 1.　空间并置模式. 空间并置模式 $c(c \subseteq F)$ 是空间特征集 F 的子集，c 的阶为 c 中特征的数量.

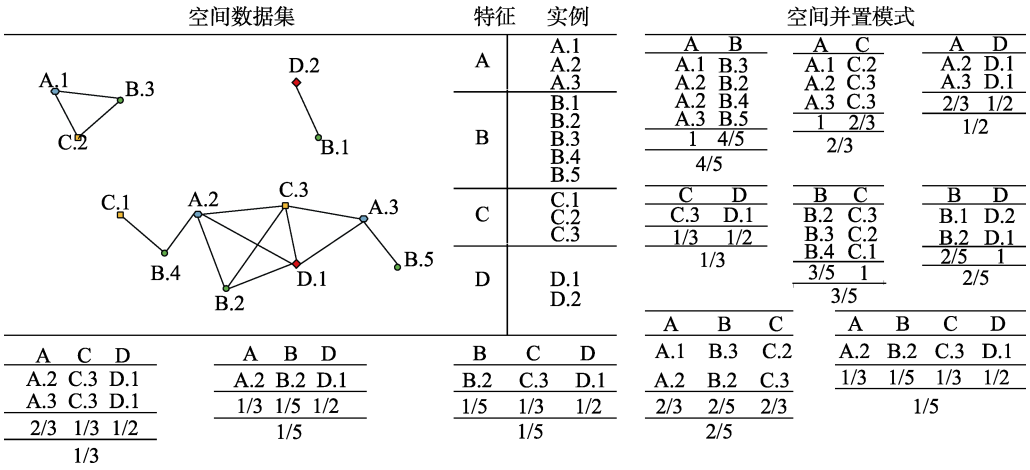


图 2　空间数据集和空间并置模式示例

定义 2. 表实例. 一个空间团 cl 包含了 c 中所有的空间特征并且 cl 的任意子集不包含 c 中所有空间特征，则称 cl 为 c 的一个行实例. c 的所有行实例构成了 $\boldsymbol{c}$ 的表实例 $T(c)$.

定义 3. 参与率. 参与率 $PR(c, f_i)$衡量特征 f_i 在 k 阶空间并置模式 $\boldsymbol{c} = \{f_1, f_2, ..., f_k\}$中的参与情况，$PR(c, f_i)$ = | f_i在 $T(c)$中的不重复实例个数|/|f_i 在空间中的实例个数|.

定义 4. 参与度. 空间并置模式 c 的所有空间特征中参与率的最小值为 c 的参与度 $PI(c)$，即 $PI(c) = \min_{fi \in c}\{PR(c, f_i)\}$.

定义 5. 频繁空间并置模式. 当 $PI(c)$不小于给定的最小频繁性阈值 min_prev 时，称空间并置模式 c 是一个频繁空间并置模式.

图 2 显示了一个空间数据集的分布，该数据集有 A、B、C、D 四个特征，A 有三个实例 A.1、A.2 和 A.3，B 有五个实例 B.1、B.2、B.3、B.4 和 B.5，C 有三个实例 C.1、C.2、C.3，D 有两个实例 D.1 和 D.2. 将具有空间邻近关系的两个实例用线连接，在图 2 中还列举了此空间数据集的并置模式及其表实例. 如模式{A, B, C}的表实例为{{A.1, B.3, C.2},{A.2, B.2, C.3}}，PR{{A, B, C}, A}=|{A.1, A.2}|/|{A.1, A.2, A3}|=2/3，PR{{A, B, C}, B}=|{B.2, B.3}|/|{B.1, B.2, B3, B.4, B.5}|=2/5，PR{{A, B, C}, C}=|{C.2, C.3}|/|{C.1, C.2, C.3}|=2/3，PI({A, B, C})=min{PR{{A, B, C}, A}, PR{{A, B, C}, B}, PR{{A, B, C}, C}}=0.4. 若 min_prev 小于等于 0.4，则{A, B, C}是一个频繁空间并置模式.

引理 1. 空间并置模式的反单调性原理. 空间并置模式的阶数与参与率(PR)和参与度(PI)呈负线性相关.

证明. 假设空间并置模式 $c = \{f_1, f_2, \cdots, f_k\}$，$c' = \{f_1, f_2, \cdots, f_{k'}\}$，其中 $k' < k$，$c' \subseteq c$. 假设某个空间特征 $f_i (i < k')$的实例包含在 $\boldsymbol{c}$ 的行实例中，那么这个实例也一定包含在 c'的行实例中，反之则不成立. 所以 $PR(c, f_i)\} \leqslant PR(c', f_i)$，$PR$ 与 c 的阶呈负线性相关. 并且 $PI(c) = \min_{fi \in c}\{PR(c, f_i)\} \leqslant \min_{fi \in c'}\{PR(c', f_i)\} = PI(c')$，所以 PI 也与 c 的阶呈负线性相关.

定理 1. 空间并置模式的向下闭合性原理. 如果一个空间并置模式不是频繁的，则它的任意超集也一定不是频繁的. 如果一个空间并置模式是频繁的，则它的任意子集一定是频繁的.

证明. 假设空间模式 $c' = \{f_1, f_2, \cdots, f_{k'}\}$，$c = \{f_1, f_2, \cdots, f_k\}$，$s = \{f_1, f_2, \cdots, f_{k''}\}$，其中 $k' < k < k''$，$c' \subseteq c \subseteq s$. 判断模式是否频繁，根据模式的参与度 PI 是否大于等于最小频繁性阈值 min_prev. 由引理 1 可知 $PI(s) \leqslant PI(c) \leqslant PI(c')$，故当 $PI(c)$小于 min_prev 时，$PI(s)$也一定小于 min_prev. 所以当 c 不是频繁空间并置模式时，c 的任意超集 s 也一定不是频繁空间并置模式. 同理可得当 c 是频繁空间并置模式时，c 的任意子集 c'也一定是频繁空间并置模式.

例 2. 设最小频繁性阈值为 2/3，由图 2 可知模式{A，D}的 PI 值为 1/2，它的超集{A，B，D}、{A，C，D}和{A，B，C，D}的 PI 值都不大于 1/2，所以当模式{A，D}不是频繁空间并置模式时，它的所有超集也一定不是频繁空间并置模式.

4 CPM-MCHM 算法

本节将从极大团挖掘、极大团存储以及两阶段挖掘过程三个方面详细介绍本文提出的基于极大团和哈希表的空间并置模式挖掘算法 CPM-MCHM（Co-location Pattern Mining based on Maximal Clique and Hash Map）. 在给定条件下，算法采用如图 3 所示的两阶段挖掘框架，包括以下五个步骤：（1）根据距离阈值和空间数据集产生空间实例之间的邻近关系.（2）根据空间实例间的邻近关系挖掘所有极大团并构建哈希表来存储得到的极大团.（3）根据哈希表生成团候选模式，计算团候选模式的表实例及参与度.（4）应用团候选模式的表实例计算剩余所有模式的参与度.（5）基于最小频繁性阈值过滤频繁空间并置模式.

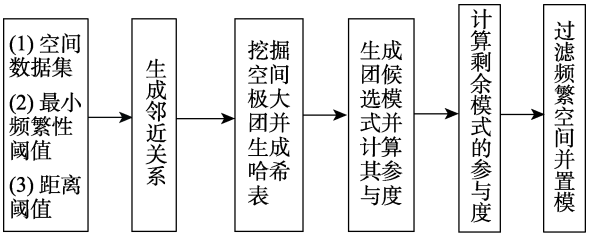


图 3 基于极大团和哈希表的两阶段挖掘框架

将图 1 和图 3 进行对比可以看出，由于逐阶候选-测试挖掘算法是逐阶迭代过程，每一次迭代都需要使用最小频繁性阈值来过滤出频繁空间并置模式且高阶模式表实例的计算是基于其相应低阶模式的表实例进行. CPM-MCHM 算法采用两阶段挖掘框架，只在最后一个步骤中使用到最小频繁性阈值来过滤频繁空间并置模式. 当最小频繁性阈值改变时，逐阶候选-测试挖掘算法大部分工作都要重新进行，而 CPM-MCHM 算法可以基于已经得到的模式

参与度，与新的最小频繁性阈值进行比较就可以快速获得新的频繁模式.

4.1　极大团挖掘算法

行实例实际上是基于实例之间邻近关系的团，如果能采取一个紧凑的数据结构来储存所有的团关系，避免逐阶挖掘，就可以节约生成候选和识别表实例的成本.

引理 2.　若空间极大团 cl 是空间并置模式 c 的一条行实例，则从 cl 中能得到模式 c 及其所有子集的一条行实例.

证明.　假设空间并置模式 $c = \{f_1, f_2, ..., f_k\}$, $c' = \{f_1, f_2, ..., f_{k'}\}$, 其中 $k' < k$, $c' \subseteq c$. 如果极大团 $cl = (o_1, o_2, \ldots, o_k)$是空间并置模式 c 的一条行实例，其中 o_i 是 f_i 的实例. 因为 c'是 c 的子集，所以从极大团 cl 中得到的团 $cl' = (o_1, o_2, \ldots, o_{k'})$也是 c'的一条行实例. 故从 cl 中能得到模式 c 及其所有子集的一条行实例.

引理 3.　任何候选模式的行实例一定包含在空间极大团中.

证明.　因为行实例实际上就是满足空间邻近关系下的团关系，而数据集的任何团关系一定包含在该数据集的极大团中，所以任何候选模式的行实例一定可以从空间极大团中得到.

例 3.　表 1 中得到的空间极大团{A.2, B.2, C.3, D.1}就是空间并置模式{A, B, C, D}的一条行实例，并且极大团{A.2, B.2, C.3, D.1}包含的团关系{A.2, B.2, C.3}、{A.2, B.2, D.1}、{A.2, C.3, D.1}、{B.2, C.3, D.1}又是模式{A, B, C}、{A, B, D}、{A, C, D}、{B, C, D}的行实例.

基于引理 2 和引理 3，可以采用极大团来物化空间实例的邻近关系，然后基于极大团挖掘频繁空间并置模式. 因为模式表实例的计算不依赖于其低阶模式的表实例，所以彻底摆脱了逐阶候选-测试框架. 另外，空间极大团存储压缩了邻近关系，算法的空间耗费也得到有效减少.

通常从已知数据集中生成极大团是 NP 难问题，因此需要一种有效的算法能够快速从空间数据集中生成所有极大团. Bron–Kerbosch 算法是一种被广泛应用于枚举极大团的算法，由于它易于理解和编码，在实际运用中表现良好[28]. 本文采用了 Bron–Kerbosch 算法枚举极大团的思想并将其改良后运行于枚举空间极大团. 表 1 列举了图 2 中空间数据集的所有极大团.

表 1　图 2 空间数据集中的极大团

空间极大团	
{A.1, B.3, C.2}	{A.3, B.5}
{A.2, B.2, C.3, D.1}	{B.4, C.1}
{A.2, B.4}	{B.1, D.2}
{A.3, C.3, D.1}	

4.1.1　带轴 Bron–Kerbosch 算法

如算法 1 所示，带轴 Bron–Kerbosch 算法使用四个顶点集，Q、P、X 和 $N(v)$，其中 Q 中的顶点形成一个团，P 是尚未被考虑的顶点，X 是已被考虑的顶点，$N(v)$是顶点 v 的所有直接邻居顶点. 为了提高算法的效率，减少递归的数量，选择邻近顶点最多的顶点 u 作为枢轴元素，对于任何极大团，要么 u 本身，要么 u 的非邻近顶点必然包含其中，所以只需要在 u 及其非邻近顶点上进行迭代.

算法 1.　带轴 Bron-Kerbosch 算法

输入：三个不相交的顶点集合 Q、P、X
　　$N(u)$:u 的所有邻近节点集合

输出：所有极大团

```
1.function Bron–Kerbosch-piovt(Q, P, X)
2.    IF P∪X = ϕ.
3.    把 Q 加入极大团集合
4.    选择点 u∈P∪X 使|P∩N(u)| = max|P∩N(u)|
5.    FOR v∈P-N(u) DO
6.        Bron-Kerbosch-piovt (Q∪{v}, P∩N(v), X∩
          N(v))
7.        P = P-{v}
8.        X = X∪{v}
9.END function
```

算法过程如下：选择邻近顶点最多的顶点 u 作为枢轴元素（步骤 4），对 P 中每个 u 的非邻居顶点 v 执行如下操作，将 v 放入集合 Q 中，并将不与 v 相邻的顶点从 P 和 X 中移出，之后递归集合 Q、P、X（步骤 5~6）. 回溯到上一步后，将本次选择的顶点从 P 移到 X（步骤 7~8）.

4.1.2　基于位运算的分区 Bron–Kerbosch 算法

从算法 1 中可以看出，带轴的 Bron–Kerbosch 算法经常进行求两个集合的交集操作. 因为对两个实例集合求交集时，首先需要检查特征是否一致，再检查 ID 是否一致，如果输入数据集很大求交集操作就会变得非常繁琐. 因此本文提出了利用位运算来求解两个集合的交集以提高算法运行的效率. 首先，对所有实例按其特征进行字典序排序，如果两个实例具有相同的特征类型，则按照实例 ID 排序，生成实例表 B. 然后将求交集的实例集合分别按照实例表 B 映射为二进制形式的集合. 在求两个集合的交集时，只需要对两个二进制形式的集合进行按位与(AND)操作，再把结果映射到集合 B 中获得真

正的公共实例集，最后将所有二进制集合占用的空间释放.

当空间数据集很庞大但是非常稀疏时，仍然将两个集合映射为与实例个数相同大小的二进制实例集会造成大量的空间浪费. 因此本文采用了分区的方法来进一步提高枚举极大团的效率，如算法 2 所示. 首先从空间中随机获得 1 个实例，将该实例及其所有邻近顶点作为种子集，将种子集和与种子集相邻的所有实例作为一个分区. 接下来采用带轴 Bron-Kerbosch 算法挖掘出这个分区中的所有极大团. 最后将种子集中的实例从空间数据集中删除，进行其余部分的挖掘. 通过这种分区挖掘的方法不仅能够有效降低实例表 *B* 的长度，减少位运算的空间耗费，还能在内存有限时处理更大的数据集，提高算法的运行效率. 文献[23]证明了这种分区挖掘极大团的方法可以挖掘出正确且完整的极大团.

算法 2. 基于位运算的分区 Bron-Kerbosch 算法

输入：空间数据集 *S*

1.function *Partition-Based*(*S*)

2. WHILE *S* 中还有未被访问的顶点

3. 从 *S* 中选取 1 个未被访问的顶点，将该顶点及其所有邻近顶点放入集合 *N*

4. 将集合 *N* 以及与 *N* 中实例具有邻近关系的所有实例放入集合 N^+

5. *Bron–Kerbosch-piovt* (ϕ, N^+, ϕ)

6. *S*=*S*–*N*

7.END function

例 4. 从图 2 的空间数据集中选取实例 A.2，A.2 与它的邻近顶点{B.2, B.4, C.3, D.1}组成种子集{A.2, B.2, B.4, C.3, D.1}. 种子集以及与之相连接的元素组成分区{A.2, A.3, B.2, B.4, C.1, C.3, D.1}，可以看出从这个分区中挖掘出的极大团{A.2, B.2, C.3, D.1}、{A.2, B.4}、{A.3, C.3, D.1}、{B.4, C.1}也是整个空间数据集中的极大团.

图 4 显示了例 4 中 A.2 为枢轴元素时挖掘极大团的按位与操作过程，*B* 表示排序后的空间实例，将 *P*、*X* 和 *N* 分别映射为二进制形式的集合 *BP*、*BX* 和 *BN*，其中 1 表示集合中有该实例，0 表示集合中没有该实例. 通过 AND 操作得到了两个新的二进制位有序集 *BP*&*BN* 和 *BX*&*BN*，位为 1 表示这两个集合均有该实例，位为 0 表示这两个集合不全有该实例.

4.2 极大团存储

本文通过查找极大团来计算模式的表实例，因此需要一种数据结构来存储极大团并且在计算模式

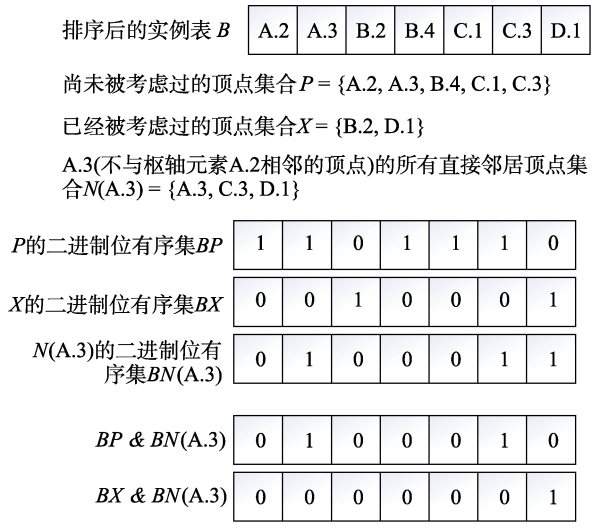


图 4 按位与操作示例

表实例时能够快速找到极大团，又因为储存的极大团是无序的，所以本文采用了一种二维的哈希结构来存储极大团，命名为 CL-hash. 所有的 CL-hash 构成哈希表，命名为 *CL-hashmap*. 哈希表能够通过给定的关键字直接访问对应的值，所以利用哈希表的特点可以提高查找极大团的速度. CL-hash 的结构为<*key*, *value*>，其中 *key* 是空间并置模式，*value* 是特征组为 *key* 的所有极大团. 在查找时，可以直接找到所有特征组为 *key* 的极大团.

根据图 2 空间数据集中的极大团构建的 *CL-hashmap* 如图 5 所示. 当得到一个极大团 *cl* 时，首先检查 *CL-hashmap* 中是否存在某个 CL-hash 的 *key* 与 *cl* 的特征组相同. 若存在则将 *cl* 中的空间实例加入该 CL-hash 的 *value* 中. 若不存在，则新建一个 CL-hash 节点，该节点的 *key* 等于 *cl* 的特征组，*value* 等于 *cl* 中的空间实例.

例 5. 空间极大团{A.2, B.2, C.3, D.1}的关键字 *key* 为{A, B, C, D}，值 *value* 为{<A, {A.2}>, <B, {B.2}>, <C, {C.3}>, <D, {D.1}>}. 空间极大团{A.1, B.3, C.2}的关键字 *key* 为{A, B, C}，价值 *value* 为

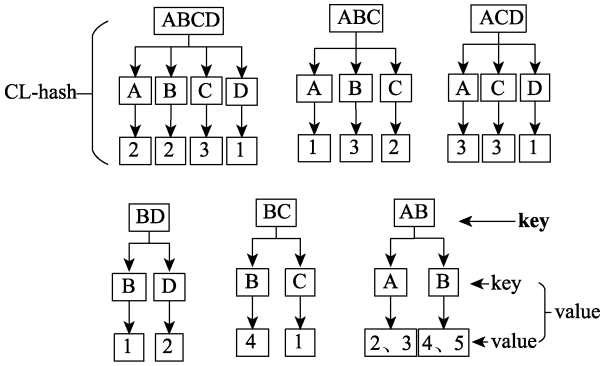


图 5 根据图 2 构建的 CL-hashmap

{<A, {A.1}>, <B, {B.3}>, <C, {C.2}>}.

4.3　两阶段挖掘过程

这一节将详细介绍本文提出的两阶段挖掘过程. 与传统的逐阶候选-测试挖掘框架不同，本文不采用逐阶过滤频繁模式的方法，而是采用了基于极大团和哈希表的两阶段挖掘框架得到所有的频繁空间并置模式. 首先由CL-hash结构的关键字*key*形成第一阶段的候选模式（也称团候选），基于CL-hash结构的值 *value* 得到团候选模式中每个特征的参与实例并计算团候选模式的参与度，所谓参与实例是指出现在模式表实例中的实例，是一种对表实例的列存储方法. 第二阶段基于团候选中特征的参与实例，计算剩余所有模式的参与度. 最后基于最小频繁性阈值过滤频繁空间并置模式.

引理 4.　任何不在团候选中的候选模式，其参与实例可以从团候选模式的参与实例中求出.

证明.　若候选模式 ***c*** 没有对应的极大团，则由引理 3 可知，*c* 的所有行实例一定包含在极大团中，而所有极大团已包含在团候选模式的表实例中，所以不在团候选中的候选模式 *c* 的参与实例可以从团候选模式的参与实例中求出.

例 6.　图 5 中模式{A, C}不是团候选模式，但是模式{A, C}的参与实例可以通过其在团候选模式中的超集{A, B, C, D}、{A, B, C}和{A, C, D}对特征 A 和特征 C 的参与实例进行求并集得到.

定理 2.　基于团候选模式的参与实例可以计算剩余所有模式的参与度.

证明.　基于引理 4，可以计算剩余所有模式的参与实例，于是可以计算出所有模式的参与率，进而计算参与度.

基于定理 1 和定理 2，本文设计了一种两阶段挖掘算法，如算法 3 所示. 在第一阶段，首先将*CL-hashmap*中所有CL-hash结构的关键字提取并存入命名为 *keyset* 的集合（团候选集合）. 从 *CL-hashmap* 中得到 *keyset* 中模式的参与实例，并计算相应模式的参与度. 在第二阶段，基于所有团候选模式的参与实例来得到剩余模式的参与实例，从而计算剩余模式的参与度，最后过滤频繁模式. 定理 1 和定理 2 保证了两阶段计算结果的正确性. 需要说明的是：第 1，第一阶段计算结束后，存储极大团的空间可以释放，第二阶段计算的剩余模式只保留参与度值. 第 2，剩余模式的产生和计算方法采用在最长极大候选模式（在团候选中并且不存在超集的模式）中进行分解和哈希查找的方法. 例如，对于图 2 所示数据集，剩余模式可以从最长团候选{*A*, *B*, *C*, *D*}中产生并通过哈希查找计算参与实例. 第 3，因为第二阶段计算的时间和空间耗费远低于计算极大团的耗费，尽管第一阶段计算结果为频繁的模式的所有子集一定也是频繁的，但考虑到改变最小频繁性阈值后算法能快速响应，两阶段算法计算了所有剩余模式的参与度.

算法 3.　两阶段挖掘过程

输入：*CL-hashmap*

输出：所有频繁并置模式

1.function *FindPrevalent*()
2.　从 *CL-hashmap* 取出所有团候选模式存入 *keyset*
3.　基于 *CL-hashmap* 得到 *keyset* 中所有模式的参与实例
4.　计算 *keyset* 中所有模式的参与度
5.　WHILE *keyset* 中含有未访问的最长极大候选模式
6.　　取出一个最长极大候选模式 *c*
7.　　将 *c* 分解为其不存在 *keyset* 中的子集，并将这些子集存入 *H*(*c*)
8.　　基于 *keyset* 中模式的参与实例计算 *H*(*c*)中所有模式的参与度
9.　将参与度大于等于 *min_prev* 的模式加入结果集
10.END function

例 7.　从图 5 获得 *keyset* = {ABCD, ABC, ACD, BC, BD, AB}（即基于极大团获得的候选模式），计算 *keyset* 中所有模式的参与实例及参与度. 比如从 *CL-hashmap* 中得到模式{A, B, C, D}的参与实例分别为(A: {A.2}), (B: {B.2}), (C: {C.3}), (D: {D.1}). 从 *CL-hashmap* 中得到模式{A, B, C}的参与实例分别为(A: {A.1, A.2}), (B: {B.2, B.3}), (C: {C.2, C.3}). 第二阶段，将最长极大候选{A, B, C, D}分解得到不存在 *keyset* 中的子集{BCD, ABD, AC, AD, CD}，从 *keyset* 中得到这些子集的参与实例并计算其参与度. 如模式{A, C}的参与实例分别为(A: {A.1, A.2, A.3}), (C: {C.2, C.3})，参与度为 2/3.

4.4　时空复杂度分析

4.4.1　时间复杂度

CPM-MCHM 算法的时间复杂度主要分为以下三个部分：

（1）极大团物化 $T_{\text{极大团物化}}(S)$（*S* 表示空间数据集的实例集合）：Tomita 等人在文献[21]中证明，有 *n* 个顶点的无向图 *G* 中，带轴 Bron–Kerbosch 算法的最坏运行时间为 $O(3^{n/3})$. 假设在每个分区中顶点个数的平均值为 n_{avg}，则挖掘极大团的时间复

杂度为 $O(|M|3^{navg/3})$，其中 M 为分区个数. 这是最坏情况下的分析，再加上基于位运算的改进，极大团物化的执行效率更高，这将在后续的实验中得到验证.

（2）团候选过滤 $T_{团候选过滤}(MC)$（MC 为空间数据集 S 的极大团集）：由于为极大团构造了哈希表 *CL-hashmap*，这部分的时间复杂度为 $O(|MC|^2)$.

（3）剩余模式过滤 $T_{剩余模式过滤}(C)$（C 是团候选之外的剩余模式集）：假设空间数据集中特征数是 m，在 2^m 的特征组合中，由于剩余模式参与实例的计算是基于哈希查找结果的合并运算，如果一次求并记为一个单位时间，则最坏的情况下，剩余模式过滤的时间复杂度为 $O(s\times m_{avg}\times 2^m)$，其中 s 是模式参与实例的平均长度，m_{avg} 是剩余模式的平均长度.

综上，CPM-MCHM 算法的时间复杂度为

$$T_{极大团化物}(S)+T_{团候选过滤}(MC)+T_{剩余模式过滤}(C)$$
$$=O(|M|3^{navg/3}+|MC|^2+s\times m_{avg}\times 2^m).$$

4.4.2 空间复杂度

本文提出的 CPM-MCHM 算法的主要空间耗费包括 *CL-hashmap* 的耗费和团候选参与实例的耗费. 设 w_{avg} 是极大团的平均长度，极大团个数为 $|MC|$，则存储哈希结构的空间复杂度约为 $O(w_{avg}\times|MC|)$. 而存储所有团候选参与实例的空间复杂度约为 $O(v_{avg}\times w_{avg}\times|MC|)$，其中 v_{avg} 是团候选中参与实例的平均长度. 所以，CPM-MCHM 算法的空间耗费约为 $O(v_{avg}\times \mathrm{w}_{avg}\times|MC|+U)$，其中 U 是存储剩余模式及参与度值的空间耗费.

4.4.3 复杂度比较

传统的逐阶候选-测试挖掘框架中速度最快的 joinless 算法的总时间耗费 T_{jl} 是：

$$T_{jl}=T_{星型邻居}+T_{jl}(2)+\sum_{k>2}T_{jl}(k),$$

其中，$T_{星型邻居}$ 是计算空间数据集的星型邻居的时间，$T_{jl}(k)$是计算 k 阶模式需要的时间.

从两方面比较 CPM-MCHM 算法和传统的 joinless 算法. 一方面是 CPM-MCHM 算法中极大团物化和求解团候选模式的时间耗费，与 joinless 算法中星型邻居物化和生成 2 阶频繁模式的时间耗费的比较. 另一方面是计算剩余模式的时间耗费与逐阶计算 $k(k>2)$阶频繁模式的时间耗费的比较.

（1）物化邻近关系与初始计算的比较

在最坏情况下，有下面的不等式成立：

$$T_{极大团物化}(S)+T_{团候选过滤}(MC)>T_{星型邻居物化}(S)+T_{jl}(2).$$

但是，若分区效果良好、各分区中实例较稠密（即极大团的长度较长）等情况下，基于位运算的分区 Bron–Kerbosch 算法表现更好.

（2）计算剩余部分的比较

首先，joinless 算法计算 k 阶($k>2$)模式的耗费主要包括 k 阶候选 C_k 的星型实例过滤和团实例过滤，即

$$T_{jl}(k)\approx T_{星型实例过滤}(C_k)+T_{团实例过滤}(C'_k),$$

其中，C_k 是 k 阶候选模式集，C'_k 是通过 joinless 算法粗糙剪枝过滤后得到的 k 阶候选模式集.

相比于 CPM-MCHM 算法中计算剩余模式的时间耗费 $T_{剩余模式过滤}(C)$，由于剩余模式集中模式参与度的计算是基于相应团候选模式的参与实例的合并计算完成的，相较于 joinless 算法的表实例查找操作，有较高的效率，另外，逐阶计算过程需要基于 k–1 阶模式的表实例计算 k 阶模式的表实例，会出现大量重复计算过程，因此，在最坏情况下，有下面的不等式成立：

$$\sum_{k>2}T_{jl}(k)>T_{剩余模式过滤}(C).$$

事实上，除了最长频繁模式的阶数很小和星型实例过滤效果良好的情况除外，CPM-MCHM 基于团候选实例计算剩余模式参与度的时空耗费都少于 joinless 逐阶计算 k 阶模式参与度的时间耗费.

Joinless 的空间耗费为 $(n_{\mathrm{star}}+m_{table}+U_{\mathrm{all}})$，其中 n_{star} 为存储星型实例的空间耗费，m_{table} 为计算过程中必须存储的模式表实例的空间耗费，U_{all} 为存储所有频繁模式的空间耗费. 对比两个算法的主要空间耗费 $n_{\mathrm{star}}+m_{tabl}$ 与 $v_{avg}\times w_{avg}\times|MC|$，由于空间极大团将空间实例间的邻近关系压缩存储，所以 joinless 具有比 CPM-MCHM 更高的空间复杂度.

4.5 算法的有效性分析

4.5.1 完整性分析

完整性意味着本文提出的算法能挖掘到所有频繁空间并置模式. 由于空间实例间的邻近关系已被完整地包含在空间极大团中，本文使用哈希表来存储所有空间极大团不会丢失空间邻近关系. 算法 3 分两阶段计算了所有模式的参与度，没有遗漏任何模式的计算，而引理 2、引理 3 和定理 2 保证了算法 3 中计算的模式参与度的可靠性. 因此，CPM-MCHM 算法可以找到所有频繁空间并置模式.

4.5.2 正确性分析

正确性意味着挖掘出的所有模式的参与度都大

于等于最小频繁性阈值. 从引理 3 和定理 2 可知，从 *CL-hashmap* 和团候选表实例可以正确地计算所有的空间并置模式的表实例及参与度，从算法 3 中可以看出只有参与度大于等于最小频繁性阈值的模式，才会被加入结果集. 故 CPM-MCHM 算法最终生成的模式均为符合要求的频繁空间并置模式.

5　实验结果与分析

5.1　对比算法及实验环境

实验的比较算法选取了 Yoo 提出的无连接的空间并置模式挖掘算法(joinless)[7]，因为 joinless 采用了星型划分模型物化了空间邻近关系，解决了连接表实例所产生的开销问题，被证明是逐阶候选-测试挖掘框架中的最优算法，将 joinless 与 CPM-MCHM 算法进行比较，证明 CPM-MCHM 算法的优越性.

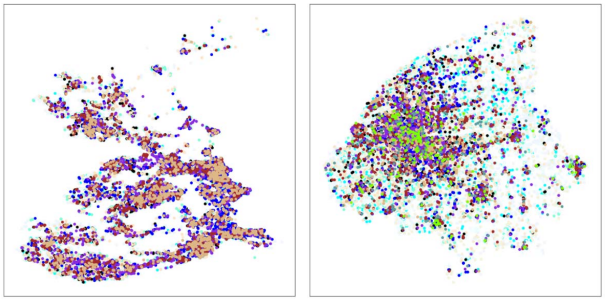
在对算法的可伸缩性进行比较时，还与文献[9]提出的基于团的算法(clique-based(NDS))和文献[10]提出的基于重叠极大团分割的算法(OMCP)进行了比较. Clique-based(NDS)的思想是在空间数据集中找出所有的空间团之后，从这些空间团中得到所有模式及其表实例，但是需要大量的空间来储存这些空间团. OMCP 的思想是利用网格划分从空间数据集中找到所有极大团之后，对极大团进行组合操作来得到所有模式及其表实例，但是组合操作可能会很耗时，储存这些从组合操作中得到的候选模式及其表实例会占用很多空间. 通过与这两个近两年内提出的且具有良好运行效率的空间并置模式挖掘算法进行对比，进一步证明了本文提出算法的优越性.

本文提出的算法 CPM-MCHM 和对比算法均使用 C++语言编写，电脑配置为 Win10 系统、16GB 内存、Intel i7-3770 3.4GHz 处理器.

5.2　实验数据集

本文采用合成数据集生成器生成合成数据集. 真实数据集采用从谷歌地图中爬取的两种分布不同的数据集：深圳市兴趣点（POI）数据集和上海市兴趣点（POI）数据集. 数据集 1 是深圳市 POI，包括城市的设施点如餐厅，超市，银行等，数据的分布如 6(a)所示，图中用不同的颜色表示不同的空间特征，可见这个数据集呈现簇状分布. 数据集 2 是上海市 POI，分布如图 6(b)所示，这个数据集呈现聚集分布，高密度分布在城市中心. 上述两个数据集的具体参数如表 2 所示.

本文的所有实验数据集和代码均可从 https://figshare.com/articles/dataset/CPM-MCHM/13301477 下载.



（a）深圳市兴趣点(POI)数据集（b）上海市兴趣点(POI)数据集

图 6　真实数据集分布图

表 2　真实数据集的参数

编号	分布空间(m^2)	实例数量	特征种类	分布特点
1	90,000 × 30,000	71,608	13	簇状
2	120,000 × 70,000	120,355	15	集中

5.3　实验评估

本文从以下 3 个方面评估提出的算法：

（1）枚举极大团：测试基于分区的位运算加入带轴 Bron-Kerbosch 算法来挖掘极大团的效果.

（2）挖掘性能：测试算法每个步骤的性能.

（3）可伸缩性：测试算法在不同数据集和参数设置上的适应性.

5.4　枚举极大团

图 7 和图 8 显示了在不同空间数据集下三种极

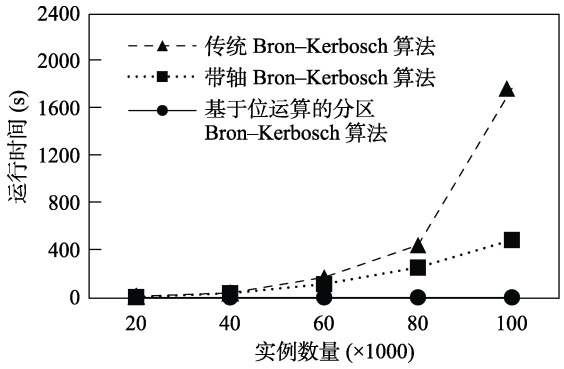


图 7　在稠密数据集中的运行时间

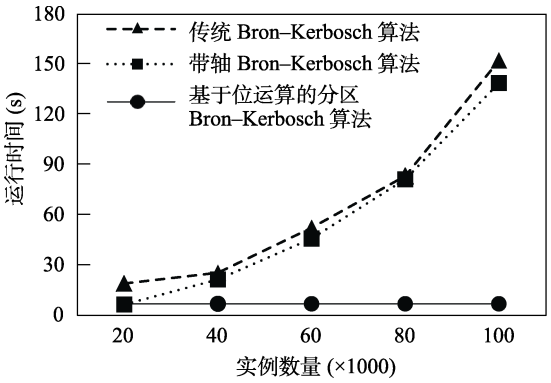


图 8　在稀疏实例集中的运行时间

大团枚举算法的运行时间. 从图中可以看出，随着实例数量的增多，三种算法的运行时间都在增加，但是基于位运算的分区 Bron-Kerbosch 算法运行时间的增长速度远小于另外两种算法. 实验证明了在不同空间大小、不同的实例数量下，基于位运算的分区 Bron-Kerbosch 算法都拥有小于带轴 Bron-Kerbosch 算法和经典的 Bron-Kerbosch 算法的运行时间，可以看出本文采用的分区和位运算方法能有效提高 Bron-Kerbosch 算法挖掘极大团的效率.

5.5 挖掘性能对比

本实验测试了 joinless 算法和本文提出的 CPM-MCHM 算法各个执行步骤的执行时间. 实验合成数据及参数设置：实例数量为 6 万（60k）个实例，特征数目为 15，距离阈值为 10，最小频繁性阈值为 0.2，*clumpiness* 为 1（参数 *clumpiness* 控制同一区域内的同一个模式的行实例数量）. 表 3 中列出的各个步骤的含义如下：

（1）*T_gen_neighbor* 是物化邻近关系所用的时间（joinless 为星型邻居物化，CPM-MCHM 为极大团物化）.

（2）*T_gen_2* 是生成二阶模式所用的时间.

（3）*T_gen_k* 是 joinless 生成 *k*(*k*>2)阶模式所用的时间.

（4）*T_cal_clique_candidate* 是 CPM-MCHM 计算团候选模式所用的时间.

（5）*T_cal_surplus* 是 CPM-MCHM 计算剩余模式所用的时间.

（6）*T_filter_pattern* 是两个算法过滤频繁模式所用的时间.

（7）*T_total*(s)是算法的总运行时间.

（8）*S_consp* (MB)是算法所需的空间耗费.

两种算法都挖掘出了相同且正确的频繁模式，但是从表 3 的结果可以看出，相同环境下无论空间数据集是相对稀疏或稠密（以相同数量空间实例的不同分布范围控制其相对稀疏或稠密），CPM-MCHM 算法的运行时间和空间耗费都要小于 joinless 算法. joinless 是一种无连接的算法，使用实例查找机制避免了实例的连接，所以 joinless 的运行时间主要是用来生成和查找 *k* 阶模式及其表实例. CPM-MCHM 算法不产生候选模式和查找表实例，运行时间主要分为四部分，物化邻近关系（枚举极大团）、计算团候选模式、计算剩余候选模式和过滤频繁模式. 其中枚举极大团是 CPM-MCHM 算法最耗时的步骤. 但是本文采用了分区和位运算的方法来降低极大团枚举的时间，大量的实验结果表明，枚举极大团的时间耗费均小于joinless算法中星型邻居物化的时间. 在相对稠密的数据集中，满足邻近关系且构成团的实例数量更多，所以 joinless 和 CPM-MCHM 在相对稠密的数据集中时空耗费更大. 从表 3 中可以看出在相对稠密的数据集中 joinless 算法的运行时间超过 CPM-MCHM 算法运行时间的 248 倍，空间耗费超过 CPM-MCHM 算法的 42 倍. 在相对稀疏的数据集中 CPM-MCHM 算法的优势仍然非常明显.

表 3 joinless 算法和 CPM-MCHM 算法各个步骤的执行时间对比

算法	joinless		CPM-MCHM	
执行时间 / 各个步骤	稠密 (1000×1000)	稀疏 (10000×10000)	稠密 (1000×1000)	稀疏 (10000×10000)
T_gen_neighbor(s)	254.314	84.674	13.069	1.871
T_gen_2(s)	62.453	14.557	–	–
T_gen_k(s)	6446	180.586	–	–
T_cal_clique_candidate(s)	–	–	11.033	0.829
T_cal_surplus(s)	–	–	3.091	0.287
T_filter_pattern(s)	4.103	0.966	0.001	0.001
T_tota(s)	6766.87	280.783	27.194	2.988
S_consp (MB)	4724.234	465.14	111.668	60.45

5.6 可伸缩性比较

这一部分测试了实例数量、特征数量、*clumpiness*、距离阈值和最小频繁性阈值对 joinless、clique-based(NDS)、OMCP 和 CPM-MCHM 的影响来检验算法的可伸缩性. 前三个实验中，测试了实例数量、特征数量和 *clumpiness* 对于算法的影响. 后两个实验中，使用两组真实数据集来测试距离阈值、最小频繁性阈值对于算法的影响. 对于实例数量、距离阈值、最小频繁性阈值这三个重要参数，还测试了它们对算法空间耗费的影响. 通过以上实验证明了 CPM-MCHM 在不同的数据集和不同的参数下都有着更少的时空耗费.

5.6.1 实例数量的影响

本实验测试了不同实例数量对算法的影响. 实验数据设置：范围为 2k×2k，特征数目为 15，距离阈值为 10，最小频繁性阈值为 0.2，*clumpiness* 为 1. 如图 9 所示，当实例数量从 50k 到 250k 时，四种算法的运行时间都随着实例数量的增加而增加. 图 10 显示了四种算法在不同数据集中的空间耗费. 从图中可以看出随着空间实例数量的增加，joinless、clique-based(NDS)和 OMCP 的空间耗费远大于 CPM-MCHM 的空间耗费. 因为实例数量达到 250k

时，joinless 的时空耗费远远大于其他三种算法，为了更好的效果呈现，图中省去了 joinless 在实例数量为 250k 时的运行时间和空间耗费.

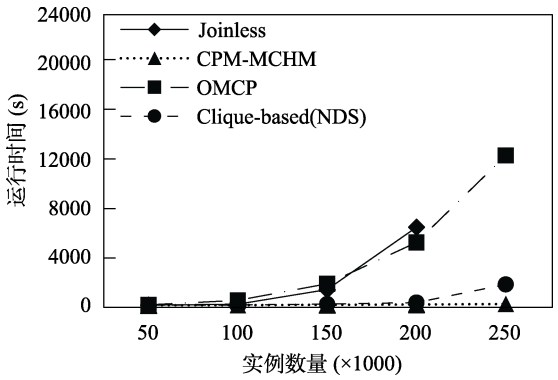


图 9　不同实例数量下各算法的运行时间

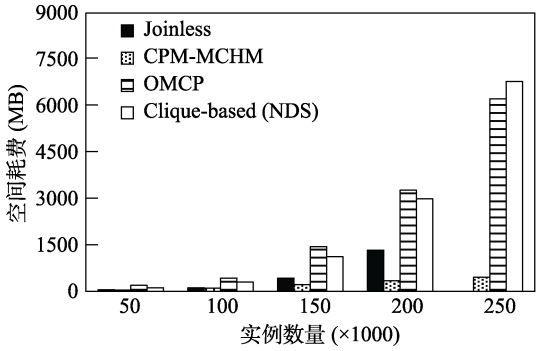


图 10　不同实例数量下各算法的空间消耗

实验证明了 CPM-MCHM 在不同实例数量下都拥有比 joinless、clique-based(NDS)和 OMCP 更低的时空耗费，并且实例数量越大优势越明显. 出现上述结果的原因是实例数量增加会导致空间中团和极大团的数量增加，模式的行实例增多，joinless 的查找时间以及储存表实例的空间也随之增加. 空间中团和极大团数量增加，clique-based(NDS)查找和存储团的时空耗费增加，OMCP 对极大团进行组合操作的时空耗费也在增加. 因为空间实例集很大时，空间极大团个数远小于空间团的个数并且 CPM-MCHM 不用逐阶生成模式的表实例也不用对极大团进行组合操作，所以 CPM-MCHM 有着比其他三种算法更小的时空耗费.

5.6.2　特征数量的影响

本实验测试了不同特征数量对算法的影响. 实验数据设置：范围为 2k×2k，实例数目为 60k，距离阈值为 10，最小频繁性阈值为 0.2，*clumpiness* 为 1. 如图 11 所示，当特征数目从 10 到 30 时，CPM-MCHM 和 clique-based(NDS)的运行时间随特征数目的增加而增长，joinless 的运行时间随特征数目的增加而减少，OMCP 的运行时间变化不明显.

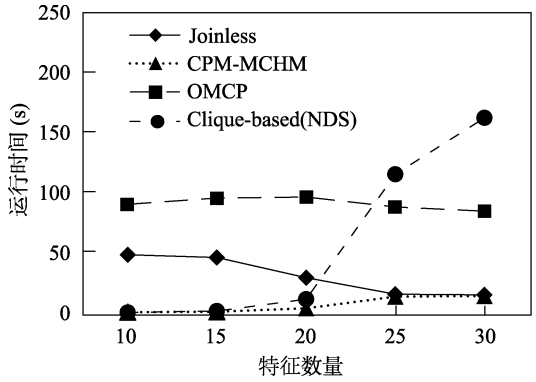


图 11　不同特征数量下各算法的运行时间

joinless 查找高阶模式表实例所用的时间大于查找低阶模式表实例的时间. 实例数量不变，特征数量增加时，低阶模式增加高阶模式减少，所以 joinless 的运行时间随特征数量的增加而减少. 特征数量增加时，空间团和空间极大团个数也增加，所以 clique-based(NDS)和 CPM-MCHM 的运行时间有所增加. 但是空间团的个数大于空间极大团的个数，所以随着特征数量的增加，clique-based(NDS)运行时间的增长幅度要高于 CPM-MCHM. 由于 OMCP 在查找空间极大团时采用了网格划分的方法，查找极大团的时间只与实例分布有关，所以 OMCP 的运行时间变化不明显.

5.6.3　*clumpiness* 的影响

本实验测试了不同 *clumpiness* 对算法的影响. 实验合成数据设置：范围为 10k×10k，实例数量为 25k，特征数量为 17，距离阈值为 10，最小频繁性阈值为 0.2. 将空间区域划分为相同的网格，*clumpiness* 就表示网格内同一模式行实例的数量，*clumpiness* 值越大表示网格内的同一模式行实例数目越多. 如图 12 所示，四种算法的运行时间随着

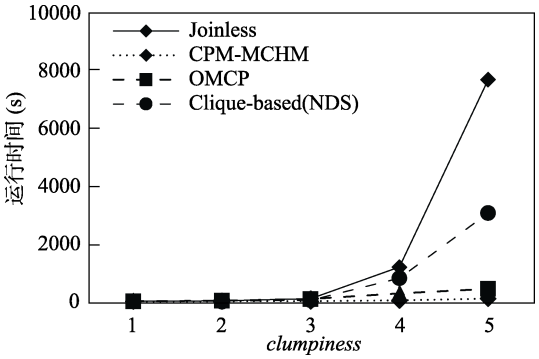


图 12　不同 *clumpiness* 下各算法的运行时间

clumpiness 值的增加而增加. 而 joinless 和 clique-based(NDS)的增长幅度远大于 CPM-MCHM 和 OMCP. 因为 *clumpiness* 值越大，空间极大团和空间团的个数越多，行实例也越多，所以 joinless 和 clique-based(NDS)的运行时间越长. 又因为空间团的个数增长幅度大于空间极大团，所以 clique-based(NDS)运行时间的增长幅度要高于 CPM-MCHM 和 OMCP.

5.6.4 距离阈值的影响

本实验测试了不同距离阈值对算法的影响. 使用两个真实数据集进行实验. 图 13 和图 14 显示了四种算法在真实数据集 1 和真实数据集 2 下的运行时间. 两个图都显示四种算法的运行时间随着距离阈值的增加而增长，joinless 的增长速度最快，运行时间最长，CPM-MCHM 的增长速度最慢，运行时间最短. 图 15 和图 16 显示了四种算法在真实数据集 1 和真实数据集 2 下的空间耗费. 两个图都显示四种算法的空间耗费随着距离阈值的增加而增加，clique-based(NDS)的增加速度最快，空间耗费最多，CPM-MCHM 的增加速度最慢，空间耗费最小. 当距离阈值为 160 时，在上海市 POI 中 joinless 的时空耗费远大于其他三种算法，为了更好的效果呈现，所以省略了该点.

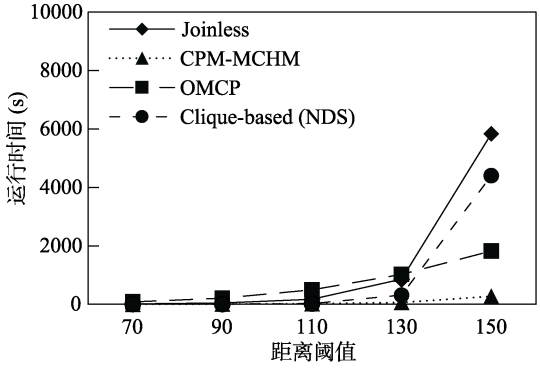


图 13 不同距离阈值下各算法的运行时间

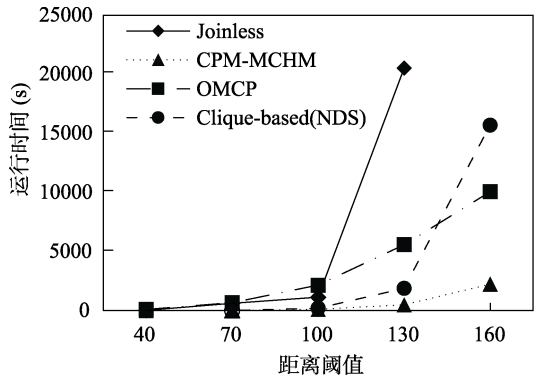


图 14 不同距离阈值下各算法的运行时间

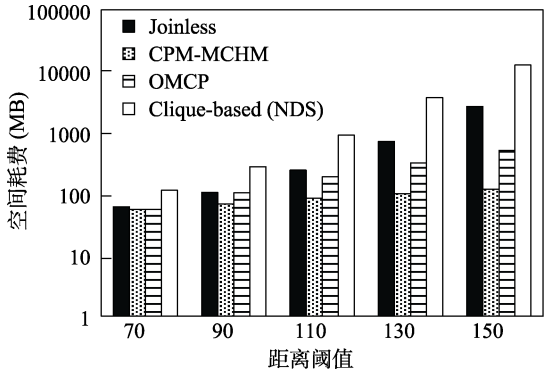


图 15 不同距离阈值下各算法的空间耗费

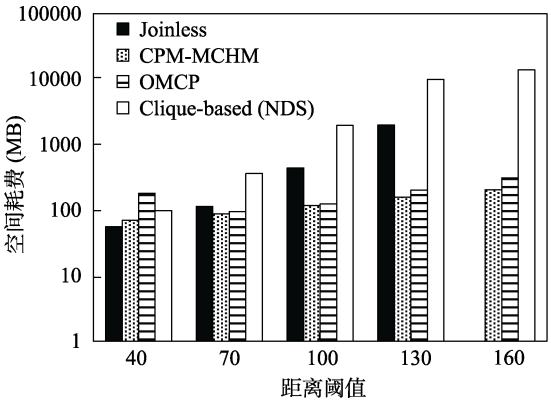


图 16 不同距离阈值下各算法的空间耗费

从实验可以看出，在不同的距离阈值下 CPM-MCHM 有比 joinless、clique-based(NDS)和 OMCP 更低的时空耗费，随着距离阈值的增大这种优势更加明显. 距离阈值决定邻近实例的数量，距离阈值增大则邻近实例的数量增多，而邻近实例的数量增多，相应的表实例也会增大，导致 joinless 的时空耗费增大. 邻近实例的数量增多还会导致空间团和空间极大团数量的增多，所以 CPM-MCHM、clique-based(NDS)和 OMCP 的时空耗费也会增大. 由于空间团的增长速度远大于空间极大团的增长速度，所以 clique-based(NDS)时空耗费的增长速度要大于 CPM-MCHM 和 OMCP. 深圳市 POI 呈现簇状分布，当距离阈值到达 130 时，四种算法的时空耗费快速增大. 上海市 POI 呈现聚集分布，当距离阈值达到 100 时，四种算法的时空耗费快速增大，并且增长幅度大于在深圳市 POI 中. 由于聚集分布类型的数据比簇状分布的更加密集，当距离阈值增加时，满足邻近关系且构成团的实例数量增长更快，所以四种算法在上海市 POI 中时空耗费增加的更快.

5.6.5 最小频繁性阈值的影响

本实验测试了不同的最小频繁性阈值对算法的影响. 使用两个真实数据集进行实验，图 17 和图 18 显示了四种算法在真实数据集 1 和真实数据集 2 下的

运行时间. 图 19 和图 20 显示了四种算法在真实数据集 1 和真实数据集 2 下的空间耗费. 四个图显示了 joinless 的时空耗费随着最小频繁性阈值的增加而减少，其他三种算法的时空耗费均没有明显变化.

最小频繁性阈值决定了频繁模式的数量，当阈值较小时，满足条件的模式更多. 从图中可以看出，在深圳市 POI 和上海市 POI 中，最小频繁性阈值分别为 0.2 和 0.3 时,joinless 的运行时间急剧下降. 由于上海市 POI 呈聚集分布且实例数量比深圳市 POI 多，参与度较大的模式更多，满足较大阈值的模式也更多，当最小频繁性阈值增加时，频繁模式的数量减少的速度也更快. 由于 clique-based(NDS)、OMCP 和 CPM-MCHM 均使用了团来物化邻近关系，对最小频繁性阈值不敏感，所以不论数据呈现什么分布，时空耗费变化都不明显.

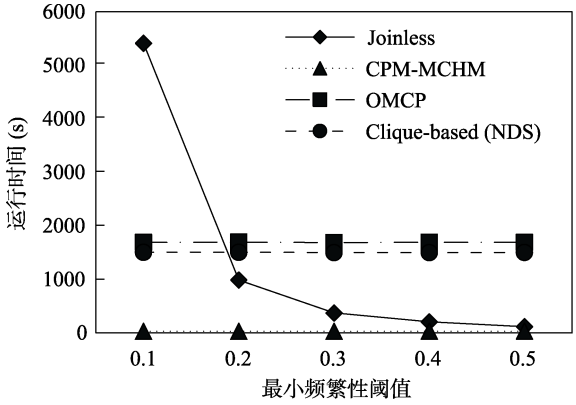


图 17　不同最小频繁性阈值下各算法的运行时间

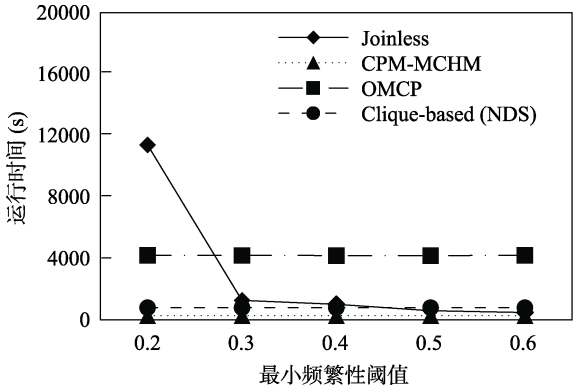


图 18　不同最小频繁性阈值下各算法的运行时间

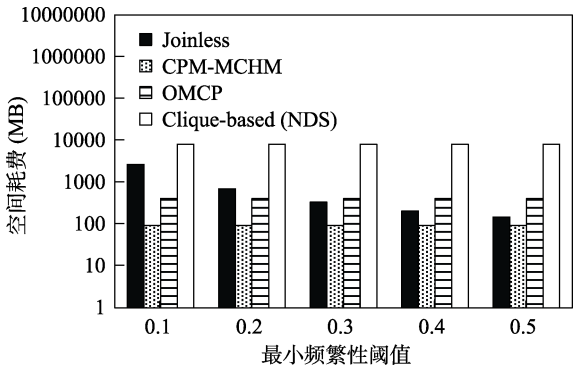


图 19　不同最小频繁性阈值下各算法的空间耗费

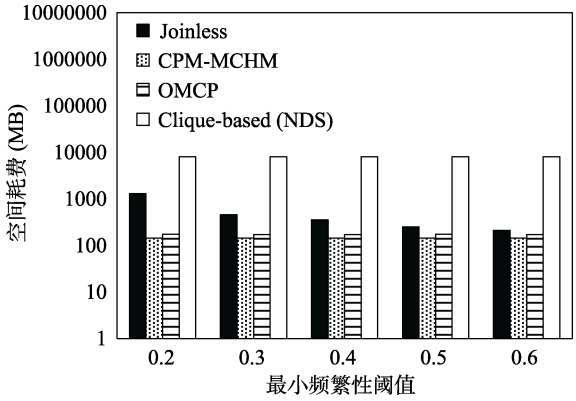


图 20　不同最小频繁性阈值下各算法的空间耗费

6　结束语

传统的空间并置模式挖掘算法，通常需要逐阶生成每个候选模式及其表实例，所以时间和空间开销较大. 本文提出了一种基于极大团和哈希表的空间并置模式挖掘算法，避免了逐阶生成模式和表实例所产生的巨大开销. 同时传统的空间并置模式挖掘算法对最小频繁性阈值敏感，当用户指定的最小频繁性阈值改变时，整个算法都要重头再来，而本文提出的算法只需重新进行少部分步骤即可得到新的频繁空间并置模式. 最后本文在合成数据集和真实数据集上进行了大量实验，与经典的传统算法和近两年内学者们提出的两种效果较好的算法进行比较，证明了本文提出方法的有效性和准确性.

参　考　文　献

[1] Wang L, Chen H. Spatial Pattern Mining Theory and Method. Beijing: Science Press, 2014: 15-135(in Chinese)
(王丽珍, 陈红梅. 空间模式挖掘理论与方法. 北京: 科学出版社, 2014)

[2] Yao X, Jiang X, Wang D, et al. Efficiently mining maximal co-locations in a spatial continuous field under directed road networks. Information Sciences, 2021, 542(4): 357-379

[3] Masrur A., Thakur G, Sparks K, et al. Co-location pattern mining of geosocial data to characterize urban functional spaces// Proceedings of the 2019 IEEE International Conference on Big Data. Los Angeles, USA, 2019: 4099-4102

[4] Li J, Adilmagambetov A, Zaiane, et al. On discovering co-location patterns in datasets: a case study of pollutants and child cancers. GeoInformatica, 2016, 20: 651-692

[5] Flouvat F, Selmaoui-Folcher N, Gay D, et al. Constrained colocation mining: application to soil erosion characterization// Proceedings of the ACM Symposium on Applied Computing. Sierre, Switzerland, 2010: 1054-1059

[6] Deng M., He Z, Liu Q, et al. Multi-scale approach to mining significant spatial co-location patterns. Transactions in GIS, 2017, 21(5): 1023-1039

[7] Yoo J S, Shekhar S. A joinless approach for mining spatial co-location patterns. IEEE Transactions on Knowledge and Data Engineering, 2006, 18(10): 1323-1337

[8] Huang Y, Shekhar S, Xiong H. Discovering colocation patterns from spatial data sets: A general approach. IEEE Transactions on Knowledge and Data Engineering. 2004, 16(12): 1472-1485

[9] Bao X, Wang L. A clique-based approach for co-location pattern mining. Information Sciences, 2019, 490: 244-264

[10] Tran V, Wang L, Zhou L. Mining spatial co-location patterns based on overlap maximal clique partitioning//Proceedings of the 20th IEEE International Conference on Mobile Data Management. HongKong, China, 2019: 467-472

[11] Wang L, Bao Y, Lu Z. Efficient discovery of spatial co-location patterns using the iCPI-tree. The Open Information Systems Journal, 2009, 3(1): 69-80

[12] Andrzejewski W, Boinski P. GPU-accelerated collocation pattern discovery. Advances in Databases and Information Systems, 2013, 8133: 302-315

[13] Andrzejewski W, Boinski P. Parallel GPU-based PlaneSweep algorithm for construction of iCPI-trees. Journal of Database Management, 2015, 26(3): 1-20

[14] Kumar N, Sathya S. COPMOC: Co-location pattern mining using map overlay and clustering techniques//Proceedings of the 16th International Conference on Informatics and Analytics. Pondicherry, India, 2016: 1-8

[15] Xiao X, Xie X, Luo Q, et al. Density based co-location pattern discovery//Proceedings of the 16th ACMSIGSPATIAL International Conference on Advances in Geographic Information Systems. New York, USA, 2008: 1-10

[16] He F, Deng X, Fang J. A fast approach for spatial co-location pattern mining//Proeedings of 2013 IEEE International Geoscience and Remote Sensing Symposium. Melbourne, Australia, 2013: 3654-3657

[17] Sainju A, Aghajarian D, Jiang Z. Parallel grid-based colocation mining algorithms on GPUs for big spatial event data. IEEE Transactions on Big Data, 2018, 6(1): 107-118

[18] Yang P, Wang L, Wang X. A MapReduce approach for spatial co-location pattern mining via ordered-clique-growth. Distributed and Parallel Databases, 2019, 38: 531–560

[19] Bron C, Kerbosch J. Algorithm 457: finding all cliques of an undirected graph. Communications of the ACM, 1973, 16(9): 575- 576

[20] Eppstein D, Strash D. Listing all maximal cliques in large sparse real-world graphs//Proceedings of the 10th International Symposium on Experimental Algorithms. Crete, Greece, 2011: 364-375

[21] Tomita E, Tanaka A, Takahashi H. The worst-case time complexity for generating all maximal cliques and computational experiments. Theoretical Computer Science, 2006, 363(1): 28-42

[22] Li X, Zhou R, Chen L, et al. Finding a summary for all maximal cliques//Proceedings of the 37th IEEE International Conference on Data Engineering. Chania, Greece, 2021: 1344-1355

[23] Cheng J, Zhu L, Ke Y, Chu S. Fast algorithms for maximal clique enumeration with limited memory//Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Beijing, China, 2012: 1240-1248

[24] Viard T, Latapy M, Magnien C. Computing maximal cliques in link streams. Theoretical Computer Science, 2016, 609(1): 245-252

[25] Al-Naymat G. Enumeration of maximal clique for mining spatial co-location patterns//Proceedings of the IEEE/ACS International Conference on Computer Systems & Applications. Doha, Qatar, 2008: 126-133

[26] Mukherjee A., Xu, P, Tirthapura S. Enumeration of maximal cliques from an uncertain graph. IEEE Transactions on Knowledge and Data Engineering, 2017, 29(3), 543-555

[27] Moon J, Moser L. On cliques in graphs. Israel Journal of Mathematics, 1965, 3(1): 23-28

[28] Himmel A, Molter H, Niedermeier R, et al. Adapting the bron–kerbosch algorithm for enumerating maximal cliques in temporal graphs. Social Network Analysis & Mining, 2017, 7(1): 1-16

ZHANG Shao-Xue, master student. Her current research interest is spatial data mining.

WANG Li-Zhen, Ph. D., professor, Ph.D. supervisor. Her research interests include spatial data mining, interactive data mining, big data analytics.

TRAN Van-Ha, Ph. D. candidate. His current research interests include spatial data mining.

Background

With the rapid development of information, data has penetrated into every industry today and has become an important production factor. There is a stronger demand for the mining and applying of the massive data. Spatial data, which carries geospatial information, is a common type of the data. It has richer and more complex semantic information than general data. Therefore, the mining of spatial data is more complicated than the transactional data. Spatial data mining aims to dig out a large amount of potential and useful patterns from spatial databases, and has been widely used in many fields. Spatial co-location pattern is a set of spatial features, and their instances are frequently located together in space. For example, supermarkets often appear near communities, and mosquitoes tend to accumulate in places where plants are lush.

Traditional co-location pattern mining algorithms such as join-based, partial-join, and joinless all use a size-by-size candidate-test mining framework, which requires candidate patterns to be generated and their table instances to be collected, but when the amount of spatial data is very large and/or dense, the number of table instances is huge, Then collecting table instance will cause a lot of execution time and memory space. Therefore, this paper proposes a novel two-stage mining framework named Co-location Pattern Mining algorithm based on Maximal Clique and Hash Map (CPM-MCHM). Instead of relying on table instances, this paper uses maximal cliques to compress neighbor relationships of instances of patterns, thereby saving space overhead. And then, the maximal cliques are compressed in a hash table structure instead of converting it into transaction data.

At the same time, the traditional spatial co-location pattern mining algorithms are extremely sensitive to the minimum prevalence threshold. When the user changes the minimum prevalence threshold which is used to measure the prevalence of patterns, the algorithm must re-perform the mining process, which leads to the repetition of many steps. CPM-MCHM avoids the problem efficiently. When the user-specified prevalence threshold is changed, only a small number of steps need to be redone to obtain new prevalent collocation patterns. Finally, it is proved by theory and experiments that the algorithm proposed in this paper can find all prevalent patterns completely and correctly. In addition, we verified that CPM- MCHM can effectively improve the performance of co-location pattern mining on both synthetic and real data sets.

This work is supported by the National Natural Science Foundation of China (61966036, 61662086), and the Project of Innovative Research Team of Yunnan Province (2018HC019).

In recent years, the authors have been committed to the researches of spatial co-location pattern mining. In this work, we break the traditional spatial co- location pattern mining framework based on the table instance, and propose a novel two-stage mining framework based on maximal cliques and hash tables.