中国科学: 信息科学 2022 年 第 52 卷 第 6 期: 1053–1068
SCIENTIA SINICA Informationis
论文

一种基于列计算的空间并置模式挖掘方法

杨培忠, 王丽珍*, 王晓璇, 周丽华

云南大学信息学院, 昆明 650504
* 通信作者. E-mail: lzhwang@ynu.edu.cn

收稿日期: 2020–12–13; 修回日期: 2021–04–01; 接受日期: 2021–06-08; 网络出版日期: 2022–06–10

国家自然科学基金 (批准号: 62062066, 61966036, 61662086, 61762090) 和云南省创新团队 (批准号: 2018HC019) 项目资助

摘要 空间并置 (co-location) 模式挖掘旨在发现空间特征间的关联关系. 一个并置模式是空间特征集合的子集, 它们的实例在空间中频繁并置出现. 传统的并置模式挖掘方法大多基于表实例计算模式的并置程度, 但表实例的生成和存储将导致巨大的时间、空间消耗. 针对这一问题, 本文提出了一种基于列计算的空间并置模式挖掘方法, 不再生成表实例, 只需要搜索模式的参与实例. 为了加速参与实例搜索, 设计了实例搜索空间剪枝、候选参与实例验证、频繁性提前感知等优化策略. 在此基础上, 提出了 CPM-Col 算法, 讨论了算法的复杂度、正确性和完备性. 在真实和模拟数据集上进行了大量实验, 实验结果表明, 本文提出的算法比其他 7 个 baseline 算法具有更好的性能和可扩展性, 特别地, CPM-Col 算法的效率提升达到数倍至数个量级. 此外, 实验验证了本文提出的优化策略的有效性.

关键词 空间数据挖掘, 并置模式, 列计算, 搜索算法, 剪枝技术

1 引言

随着空间数据库技术的发展, 空间数据呈爆炸式增长. 空间数据挖掘技术旨在从海量空间数据中发现潜在的、具有指导性和预测性的知识. 空间并置 (co-location) 模式挖掘是空间数据挖掘研究的重要分支, 旨在发现空间特征 (事物) 之间的关联关系. 并置模式是空间特征集的子集, 其实例在地理空间中频繁地并置出现 [1], 例如, {医院, 花店, 药店}是一个空间并置模式, 因为在城市设施中它们经常一起出现. 并置模式挖掘在许多应用领域中具有重要作用, 例如, 在生态保护方面, 挖掘并置模式可以发现物种间的共生关系 [1], 对物种多样性保护具有积极意义; 在公共卫生方面, 利用并置模式挖掘研究癌症病例与污染物排放间的关联关系 [2], 可以对居民区和工厂等的合理选址布局提供指导; 其他应用领域还包括地球科学、交通运输、城市建设等 [1~3].

空间并置模式的概念首次由 Shekhar 等学者 [1] 提出, 他们使用基于距离约束的参与度作为兴趣度量 (2.1 小节介绍相关概念). 如果一个模式的参与度大于频繁性阈值则被称为频繁并置模式, 并置

引用格式: 杨培忠, 王丽珍, 王晓璇, 等. 一种基于列计算的空间并置模式挖掘方法. 中国科学: 信息科学, 2022, 52: 1053–1068, doi: 10.1360/SSI-2020-0384
Yang P Z, Wang L Z, Wang X X, et al. A spatial co-location pattern mining approach based on column calculation (in Chinese). Sci Sin Inform, 2022, 52: 1053–1068, doi: 10.1360/SSI-2020-0384

模式挖掘旨在发现所有的频繁并置模式. 参与度因具有空间统计意义而被广泛采用, 随后, 许多学者针对不同的应用和挖掘任务对并置模式挖掘研究进行扩展. 例如, 文献 [4] 将研究对象进行扩展, 研究从空间扩展对象 (如线、面等) 数据中挖掘并置模式, 突破了传统方法只能处理点对象的限制; 文献 [5] 研究从变化的空间数据中挖掘动态并置模式, 从而反映特征间的动态关系; 文献 [6] 研究多层并置模式挖掘, 通过考虑并置模式的形成机制和局部分布特点, 发现区域和全局的频繁并置模式; 文献 [7] 考虑了实例对并置模式频繁度的分数贡献, 提出新的兴趣度量方法, 解决参与度度量中重叠出现的实例的贡献被过高地计算的问题; 文献 [8] 提出基于本体知识的交互式挖掘框架, 对频繁并置模式集进行二次挖掘, 发现用户偏好的并置模式; 文献 [9] 研究挖掘掩藏在并置模式中的竞争对, 从而反映更丰富的特征实例的空间信息和位置分布, 可以从空间分布层面对 POI 竞争关系预测 [10] 提供辅助. 不同于空间影响力最大化研究 [11] 从用户移动轨迹数据中发现一组空间位置使其在有限的预算内达到影响力最大化目标, 并置模式挖掘旨在利用实例间的位置分布信息挖掘特征间的关联关系, 两者的研究对象和挖掘目标存在差异. 从空间数据中挖掘并置模式是一项具有挑战性的工作, 在现有的挖掘方法中, 并置模式的兴趣度计算需要高昂的计算成本. 因此, 本文工作致力于解决并置模式挖掘性能问题.

在并置模式挖掘中, 表实例生成是计算模式兴趣度的关键, 也是最耗时的步骤 [12]. 一个并置模式的表实例是它的所有行实例的集合, 行实例是一组空间实例的集合, 这一集合涵盖该模式的所有特征且不存在两个实例的特征相同, 并且集合中的实例形成团关系. 在提升并置模式挖掘效率的研究工作中, 主要集中在如何快速生成模式的表实例, 并提出了一些优化算法. 文献 [1] 提出基于连接的表实例生成方法, 称为 Joinbase 算法, 通过连接两个 $k-1$ 阶模式的表实例得到 k 阶模式的表实例, 但需要大量的连接操作. 文献 [12] 提出无连接的 Joinless 算法, 使用星型邻居物化空间邻近关系, 通过星型邻居快速产生模式的星型实例, 但星型实例不一定满足团关系, 需要进行团关系检查. Wang 等 [13] 提出 CPI-tree 结构物化空间邻近关系, 通过深度优先遍历 CPI-tree 生成所有模式的表实例, 不需要团关系检查. 随后, 文献 [14] 改进了 CPI-tree 算法, 提出 iCPI-tree 物化模型, 以宽度优先遍历的方式逐阶生成模式的表实例. Lin 等 [15] 提出基于行实例扩展的 NCA 算法, 借助每条行实例的邻居簇结构, 可以快速将 k 阶模式的行实例扩展成 $k+1$ 阶模式的行实例, 但该方法需要额外耗费大量的存储空间. 文献 [16] 提出基于团的并置模式挖掘方法, 该方法没有明确生成表实例, 而是先搜索空间数据集中的所有完备团, 然后将完备团以 C-hash 表压缩存储, 通过 C-hash 表保存的信息计算参与度, 但完备团的计算是耗时的. 此外, 还提出了基于 MapReduce [17] 和 GPU [18] 的并行挖掘算法, 通过并行技术解决表实例生成问题.

尽管上述工作可以提升表实例生成的效率, 但表实例的生成本质上依然是耗时的, 特别是在处理海量空间数据的时候. 因此, 本文提出一种基于列计算的空间并置模式挖掘方法, 不需要生成表实例计算并置模式的兴趣度, 只需要搜索模式的参与实例 (被行实例包含的实例). 在此基础上, 提出了 CPM-Col 算法 (co-location pattern mining algorithm based on column calculation). CPM-Col 算法主要从 3 方面提升性能. (1) 将并置模式的参与度计算由基于表实例的方式转变为基于列计算的方式, 避免了表实例的生成在时间、空间上的巨大消耗; (2) 为了加速参与实例搜索过程, 提出了一系列搜索优化策略及剪枝技术, 目的是尽可能地缩小搜索空间, 高效地搜索参与实例; (3) 在参与实例搜索过程中, 边搜索边判断模式的频繁性, 及时停止对不频繁模式的参与实例搜索. 本文从理论上分析了 CPM-Col 算法的时间、空间复杂度, 并讨论了算法的正确性和完备性. 一般情况下, CPM-Col 算法只需要搜索模式表实例中的部分行实例即可找到所有参与实例, 在性能上优于基于表实例计算的方法. 在真实和模拟数据集上进行了大量实验验证 CPM-Col 算法的性能, 与其他 7 个 baseline 算法相比, CPM-Col 算法的效率提升达到数倍至数个量级, 并且 CPM-Col 算法的空间消耗也较小, 具有更好的可扩展性.

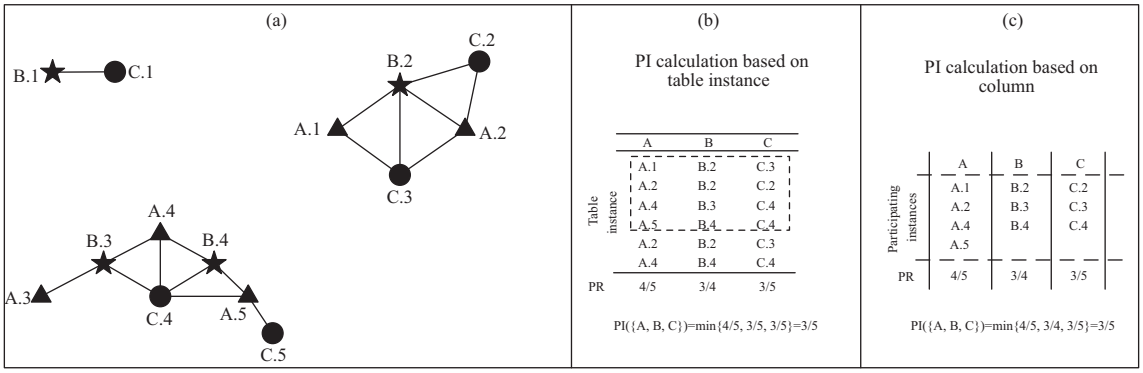

图 1 空间数据集示例及两种计算方法

Figure 1 Examples of spatial dataset and two calculation methods. (a) Spatial dataset; (b) the calculation method based on table instance; (c) the calculation method based on column

此外, 通过实验对 CPM-Col 算法进行剖析, 分析所使用的各种搜索优化策略及剪枝策略的有效性, 揭示性能提升的原因.

2 问题转化

2.1 空间并置模式挖掘

在空间数据库中, 不同类型的事物由不同的特征表示, $F = \{f_1, f_2, \ldots, f_n\}$ 表示特征集合, 每个特征有若干实例, $O = \{o_1, o_2, \ldots, o_m\}$ 表示所有实例的集合, 每个实例包含 ⟨编号, 特征类型, 位置⟩ 三元组信息. 给定距离阈值 d, 如果两个实例间的距离不大于 d, 则称它们之间具有邻近关系, 表示为 $R(o, o') \Leftrightarrow \text{distance}(o, o') \leqslant d$, 其中 $\text{distance}(o, o')$ 表示两个实例间的距离. 对于实例 o, 所有与 o 有邻近关系的实例集合称为 o 的邻居集, 表示为 $\text{Neigh}(o) = \{o' | o' \in O, R(o, o')\}$. 在 o 的邻居集中, 具有相同特征 f 的实例的集合称为 o 在 f 下的分组邻居集, 记为 $\text{groupN}(o, f) = \{o' | o' \in \text{Neigh}(o), o'.t = f\}$, 其中 $o'.t$ 表示 o' 的特征类型. 图 1(a) 展示了具有 3 个特征 (A, B, C) 的空间数据集, 同一特征下的实例由其编号区分. 实线连接的两个实例具有邻近关系. 对实例 A.4, 有 Neigh(A.4)={B.3, B.4, C.4}, groupN(A.4, B)={B.3, B.4}.

空间并置模式是一组特征集合的子集, $C = \{f_1, f_2, \ldots, f_k\}$, $k \geqslant 2$, $C \subseteq F$, 表示一个并置模式. C 中所包含的特征数目称为模式的阶数. 实例集合 $\text{RI} = \{o_1, o_2, \ldots, o_k\}$ 被称为 C 的行实例, 如果 (1) RI 中的实例数量与 C 的阶数相等, 并且 RI 中的实例涵盖 C 中的所有特征, (2) RI 中的任意两个实例具有邻近关系, 即对任意 $o_i \in \text{RI}$, $o_j \in \text{RI}$, $1 \leqslant i \leqslant k$, $1 \leqslant j \leqslant k$, $R(o_i, o_j)$ 恒成立, 也就是 RI 中的实例形成团关系. 模式 C 的行实例集合称为表实例, 表示为 $\text{table\_instance}(C)$. 为了度量并置模式的频繁性, Shekhar 等 [1] 定义了参与率和参与度. 对于特征 $f \in C$, f 在 C 中的参与率 (participation ratio, PR) 表示为 $\text{PR}(C, f)$,

$$\text{PR}(C, f) = \frac{|\Pi_f(\text{table\_instance}(C))|}{|N(f)|}, \tag{1}$$

其中, $N(f)$ 表示 f 的实例集合, Π 是带冗余消除的投影操作, 用于找到表实例中 f 的不重复实例集合, $|\cdot|$ 表示集合的元素个数. 模式 C 的参与度 (participation index, PI) 是 C 中特征的参与率最小值, 即

$$\text{PI}(C) = \min_{i=1}^{k} \{\text{PR}(C, f_i)\}. \tag{2}$$

图 1(a) 中, {A, B, C}是一个三阶并置模式, 实例集合 {A.1, B.2, C.3}是 {A, B, C}的一条行实例, 模式 {A, B, C}的表实例在图 1(b) 中完整列出, 并计算了每个特征的参与率及模式的参与度.

给定频繁性阈值 min_prev, 如果模式 C 满足 $\mathrm{PI}(C) \geqslant \mathrm{min\_prev}$, 则 C 是频繁并置模式, 并置模式挖掘旨在发现空间数据集中的所有频繁并置模式. 参与度满足向下闭合性质 [1], 即对模式 C 及其超模式 C', $C \subset C'$, 不等式 $\mathrm{PI}(C) \geqslant \mathrm{PI}(C')$ 恒成立. 借助这一性质, 大部分并置模式挖掘方法采用类 Apriori 的候选 – 测试机制逐阶进行挖掘. 在候选生成阶段通过连接 $k-1$ 阶频繁并置模式生成 k 阶候选模式, 并使用 Apriori 性质对候选模式进行剪枝; 对于未被剪枝的候选模式, 在测试阶段生成它的表实例并计算参与度, 挑选出频繁并置模式. 逐层迭代, 直到找到所有的频繁并置模式.

2.2 基于列计算的方法

在空间并置模式挖掘中, 候选模式的表实例可能是巨大的, 找到完整的表实例显然是耗时的. 尽管参与度的定义基于表实例, 但参与度的正确计算并不需要模式的完整表实例.

表实例生成是计算并置模式参与度的充分非必要条件. (充分性) 根据参与率、参与度的定义, 充分性显然成立. (非必要性) 参与度的计算并不一定需要完整的表实例, 以图 1(b) 中的模式 {A, B, C}为例, 计算其参与度只需要虚线框内的 4 条行实例即可, 因为它们包含了该模式每个特征在表实例中出现的所有实例, 这足以计算模式的参与度, 因此, 生成表实例对计算模式的参与度是非必要的.

基于列计算的方法. 基于上述考虑, 可以通过直接搜索模式中每个特征的参与实例计算其参与率, 最终得到模式的参与度.

定义1 给定并置模式 C, 特征 $f \in C$, 如果 f 的实例 o ($o.t = f$) 被包含在 C 的任一行实例中, 则 o 是 f 在 C 中的参与实例. f 在 C 中的所有参与实例集合称为 f 在 C 中的参与实例集, 表示为 $\mathrm{Obj}(C, f)$.

图 1(a) 中, A.1 是特征 A 在模式 {A, B, C}中的参与实例, 因为有行实例 {A.1, B.2, C.3}包含 A.1. 图 1(c) 列出了模式 {A, B, C}每个特征的参与实例集.

事实上, 式 (1) 对表实例按照特征 f 进行投影操作得到的就是 f 在 C 中的参与实例集 $\mathrm{Obj}(C, f)$, 因此, f 在 C 中的参与率计算等效于式 (3), 而参与度的计算不变.

$$\mathrm{PR}(C, f) = \frac{|\mathrm{Obj}(C, f)|}{|N(f)|}. \tag{3}$$

根据式 (3), 计算 $\mathrm{Obj}(C, f)$ 后就可以正确计算参与率 $\mathrm{PR}(C, f)$, 依次计算 C 中每个特征的参与率即可得到参与度. 因此, 基于列计算的方法可以避免生成模式的表实例. 然而, 在海量空间数据中, 每个特征的实例数目是巨大的, 依次搜索模式中每个特征的所有实例显然是不现实的, 因此, 如何快速搜索参与实例是基于列计算的方法急需解决的问题. 接下来, 将探索参与实例搜索策略, 提升搜索效率.

3 参与实例搜索策略

3.1 实例搜索空间剪枝

实例搜索空间剪枝策略主要依赖于参与实例集的向下包含性质.

引理1 对于 k ($k \geqslant 2$) 阶并置模式 C' 及其 k+1 阶超模式 C, $C' \subset C$, 如果特征 $f \in C'$ 且 $f \in C$, 则 $\mathrm{Obj}(C', f) \supseteq \mathrm{Obj}(C, f)$.

证明 因篇幅限制, 证明过程见补充材料, 下同.

根据引理 1, 在搜索特征 f 在模式 C 中的参与实例时, 可能的参与实例一定被包含在子模式 C' 的参与实例集 $\mathrm{Obj}(C', f)$ 中.

定义2 特征 f 在 k ($k > 2$) 阶并置模式 C 中的候选参与实例集是 f 在模式 C 的所有包含 f 的 $k-1$ 阶子模式中的参与实例集的交集, 表示为

$$\mathrm{CObj}(C, f) = \bigcap_{C' \in C_{k-1}^{f}} \mathrm{Obj}(C', f), \tag{4}$$

其中, C_{k-1}^{f} 是模式 C 的所有包含特征 f 的 $k-1$ 阶子模式的集合.

引理2 在 $\mathrm{CObj}(C, f)$ 中搜索 f 在 C 中的参与实例集是完备的, 即 $\mathrm{Obj}(C, f) \subseteq \mathrm{CObj}(C, f)$.

借助引理 2, 搜索候选参与实例集 $\mathrm{CObj}(C, f)$ 中的实例即可得到 f 在 C 中的完整参与实例集, 而不再需要搜索 f 的所有实例, 可以有效地减少实例搜索空间.

3.2 候选参与实例验证

在计算模式 C 每个特征的候选参与实例集后, 需要依次验证每个候选参与实例. 验证候选参与实例 o ($o.t \in C$) 的关键是能否找到 C 的一条行实例包含 o. 能与 o 形成 C 的行实例的实例一定来自于 o 的邻居集, 因此, 只需要将 o 在特征为 $C - \{o.t\}$ 上的分组邻居集求笛卡尔 (Decare) 积, 并找到一个满足团关系的实例组合 S, $\mathrm{RI} = S \cup \{o\}$ 一定是 C 的行实例. 但这一搜索范围可能较大, 需要进一步缩小搜索空间.

定义3 对于并置模式 C 和实例 o, $o.t \in C$, C 中其他特征 f ($f \in C$, $f \neq o.t$) 的实例可能与 o 形成 C 的行实例的实例集合称为特征 f 上的实例搜索空间, 表示为 $\mathrm{Oss}(o, f, C)$, 其中

$$\mathrm{Oss}(o, f, C) = \begin{cases} \mathrm{groupN}(o, f) \cap \mathrm{Obj}(C, f), & \mathrm{Obj}(C, f)\ \text{known}, \\ \mathrm{groupN}(o, f) \cap \mathrm{CObj}(C, f), & \mathrm{Obj}(C, f)\ \text{unknown}. \end{cases} \tag{5}$$

引理3 特征 f 上的实例搜索空间 $\mathrm{Oss}(o, f, C)$ 对于回答实例 o 是否是模式 C 的参与实例是完备的.

借助引理 3, 在计算笛卡尔积时, 特征 f 下的实例取值空间从 o 在 f 下的分组邻居集 $\mathrm{groupN}(o, f)$ 缩小到 $\mathrm{Oss}(o, f, C)$, 相应地, 需要搜索的实例组合数量也被缩减. 为了快速从所有组合中找到一个满足团关系的实例组合 S, 使用回溯算法进行搜索, 目的是及时剪掉不可能的实例组合. 此外, 如果在验证实例 o 时搜索到的行实例 RI 包含尽可能多的未被验证的其他特征的实例, 则 RI 除了可以验证 o 是 C 的参与实例, 还可以验证 RI 中其他特征的实例, 即搜索一条行实例同时验证多个候选参与实例.

实例搜索优化策略. 为了在验证实例 o 时搜索到的行实例 RI 尽可能多地包含未被确认的候选参与实例, 在回溯搜索过程中, 应优先搜索特征 f 在 $\mathrm{Oss}(o, f, C)$ 中未被确认的实例, 即将 $\mathrm{Oss}(o, f, C)$ 中还没有被包含在 $\mathrm{Obj}(C, f)$ 中的实例放在前面, 使它们被优先搜索.

3.3 模式频繁性提前感知

在搜索参与实例过程中, 可以逐步感知模式的频繁性, 及时停止对不频繁模式的参与实例搜索, 从而避免一些无效的搜索.

引理4 对于并置模式 C 及其特征 $f \in C$, $\mathrm{UPR}(C, f) = \frac{|\mathrm{CObj}(C, f)|}{|N(f)|}$ 是 f 在 C 中的参与率上界.

基于参与率上界的剪枝策略. 根据引理 4, 如果 C 中存在任一特征 f 的参与率上界小于频繁性阈值 min_prev, 则 C 一定不是频繁并置模式, 因为 $\mathrm{PI}(C) \leqslant \mathrm{PR}(C,f) \leqslant \mathrm{UPR}(C,f) < \text{min\_prev}$ 成立. 因此, 在计算模式每个特征的候选参与实例集之后, 可以先计算每个特征的参与率上界, 然后借助参与率上界条件提前将不频繁的模式剪掉. 被剪掉的模式不再需要搜索其特征的参与实例, 对于不能被剪枝的模式, 再依次搜索每个特征的参与实例.

基于参与率的剪枝策略. 在搜索模式每个特征的参与实例过程中, 当一个特征的所有参与实例都被找到后, 利用式 (3) 可以计算该特征的参与率. 根据模式频繁性的判定条件, 如果存在任一特征的参与率小于 min_prev, 该模式一定不频繁, 因此不必继续搜索其他特征的参与实例.

特征搜索优化策略. 为了充分发挥基于参与率剪枝策略的优势, 应优先搜索模式中参与率可能小于 min_prev 的特征. 在模式 C 中, 特征 f 的参与率上界越小, 则它的参与率越可能小于 min_prev. 基于这一考虑, 将 C 中的特征按其参与率上界升序进行排序, 然后再按这一排序逐个搜索特征的参与实例.

4 算法及分析

4.1 算法描述

基于第 3 节的参与实例搜索策略, 本小节提出 CPM-Col 算法 (算法 1). CPM-Col 算法采用候选 – 测试机制逐阶进行挖掘. 步骤 1 计算空间邻近关系, 并以每个实例在不同特征下的分组邻居进行物化. 步骤 2 初始化每个特征为一阶频繁并置模式作为迭代的开始. 步骤 3~12 从二阶开始, 逐阶生成候选模式, 并判断每个候选模式的频繁性, 直到没有任何频繁并置模式被找到为止, 算法最终返回所有的频繁并置模式. 步骤 6 调用 isPrevalent(C) 过程是 CPM-Col 算法的核心.

Algorithm 1 CPM-Col algorithm

Input: (a) instance set O; (b) feature set F; (c) distance threshold d; (d) prevalence threshold min_prev.

BEGIN:

1: Calculate neighbor relationships between instances, and store as grouped neighbor sets;
2: Initialize $P_1 = F$, $k = 2$;
3: **while** $P_{k-1} \neq \emptyset$ **do**
4: $P_k = \emptyset$, $C_k = \text{apriori\_gen}(P_{k-1})$;
5: **for all** $C \in C_k$ **do**
6: flag = isPrevalent(C);
7: **if** flag **then**
8: $P_k = P_k \cup \{C\}$;
9: **end if**
10: **end for**
11: $k = k + 1$;
12: **end while**
13: **return** $(P_2, \ldots, P_{k-1})$.

在 isPrevalent(C) 过程 (算法 2) 中, 步骤 1~6 计算候选模式 C 中每个特征的候选参与实例集以及参与率上界, 并根据参与率上界剪枝策略, 及时发现不频繁的模式, 停止后续搜索. 如果 C 不能被剪枝, 则步骤 7 和 8 根据特征搜索优化策略对 C 中的特征进行重新排序并初始化每个特征的参与实例集为空. 步骤 9~24 按照排序后的次序依次遍历 C 中每个特征的候选参与实例集 $\mathrm{CObj}(C,f)$. 对于

CObj(C, f) 中的实例 o, 如果 o 已经被包含在参与实例集 Obj(C, f) 中, 则不需要验证; 否则, 步骤 12 调用 SearchRI(o, C) 搜索一条包含 o 的 C 的行实例, 如果找到一条这样的行实例, 即返回的集合 RI 不为空, 则步骤 14∼16 将 RI 中的每个实例按照其特征类型保存到各自的参与实例集中, 需要注意的是, Obj(C, f) 中的实例是不重复的. 在搜索完一个特征的参与实例之后, 步骤 20 计算该特征的参与率, 如果参与率小于 min_prev, 则判定模式 C 不频繁, 并停止对其他特征的搜索. 如果搜索完 C 中的所有特征, 没有任何特征的参与率小于 min_prev, 则 C 是频繁并置模式, isPrevalent(C) 返回 True.

Algorithm 2 isPrevalent(C)

```
1: for all f ∈ C do
2:    Calculate CObj(C, f) and UPR(C, f);
3:    if UPR(C, f) < min_prev then
4:       return False;
5:    end if
6: end for
7: Sort features of C based on the feature search optimization strategy;
8: Initialize Obj(C, f) = ∅ for each feature f in C;
9: for all f ∈ C do
10:   for all o ∈ CObj(C, f) do
11:      if o ∉ Obj(C, f) then
12:         RI = SearchRI(o, C);
13:         if RI ≠ ∅ then
14:            for all o′ ∈ RI do
15:               Obj(C, o′.t) = Obj(C, o′.t) ∪ {o′};
16:            end for
17:         end if
18:      end if
19:   end for
20:   Calculate PR(C, f) by Eq. (3);
21:   if PR(C, f) < min_prev then
22:      return False;
23:   end if
24: end for
25: return True.
```

SearchRI(o, C) 过程 (算法 3) 使用带实例搜索优化策略的回溯算法寻找一条包含实例 o 的模式 C 的行实例. 其中, 步骤 1∼3 计算 C 中除 $o.t$ 之外的特征的实例搜索空间 Oss(o, f, C), $f \neq o.t$, 并根据实例搜索优化策略对其进行排序. 步骤 4 进行初始化, S 用于保存当前解空间中的实例, $\boldsymbol{X}$ 是长度为 $|C|-1$ 的向量, $\boldsymbol{X}[i]$ 记录当前解空间 S 搜索到 $C-\{o.t\}$ 中第 i 个特征 f_i 的实例在 Oss(o, f_i, C) 中的位置, flag 用于标记是否找到满足条件的解. 步骤 5 执行回溯搜索步骤 (详细代码见补充材料), 首先搜索 $C-\{o.t\}$ 中的第 1 个特征 f_1 的 Oss(o, f_1, C) 中的第 1 个实例, 由于当前 S 为空, 该实例可以加入 S; 接下来, 搜索下一个特征 f_2 的 Oss(o, f_2, C), 如果有实例满足与当前 S 中的所有实例都有邻近关系的条件, 则将该实例加入 S, 然后进入下一个特征的搜索; 否则, 进行回溯, 回溯到上一个特征的实例搜索空间中的下一个实例继续进行搜索. 以此类推, 直到找到一个实例集合 S 涵盖 $C-\{o.t\}$ 中的所有特征并且 S 中的实例具有团关系, 或者搜索完所有的可能路径. 如果找到一个满足条件的 S, 则 $S = S \cup \{o\}$ 是包含 o 的 C 的行实例被返回; 否则, S 为空集被返回.

Algorithm 3 SearchRI(o, C)

1: **for all** $f \in C - \{o.t\}$ **do**
2: Calculate Oss(o, f, C) and sort it by the instance search optimization strategy;
3: **end for**
4: Initialize $S = \emptyset$, $\boldsymbol{X} = \mathbf{0}$, $i = 1$, flag = False;
5: S = BacktrackingSearch($S, \boldsymbol{X}, i$, flag);
6: **if** $S \neq \emptyset$ **then**
7: $S = S \cup \{o\}$;
8: **end if**
9: **return** S.

4.2 算法分析

时间复杂度. 在算法 1 中, 步骤 1 计算空间邻近关系最坏情况下的复杂度为 $O(|O|^2)$, 使用平面扫描等技术, 其复杂度远小于 $O(|O|^2)$ [16]. 步骤 3~12 从二阶开始逐阶进行挖掘, 在第 $k-1$ ($k \geqslant 2$) 次迭代中, 搜索的候选模式数量为 $|C_k|$, 其复杂度为 $O(|C_k|)$. 对于 C_k 中的每个候选模式 C, 需要调用 isPrevalent(C) 过程判断其频繁性, 在 isPrevalent(C) 过程中, 步骤 9~24 是最耗时的, 最坏情况下, 需要遍历 C 中每个特征的候选参与实例, 复杂度为 $O(k \times n_1)$, n_1 为候选参与实例集 CObj(C, f) 的最大长度. 对每个候选参与实例, 步骤 12 调用 SearchRI(o, C) 搜索包含实例 o 的模式 C 的一条行实例是最耗时的, 最坏情况下, 需要搜索来自不同特征上的实例搜索空间 Oss(o, f, C) 的实例的所有组合, 复杂度为 $O(n_2^{k-1})$, n_2 为 Oss(o, f, C) 的最大长度, 并且 n_2 小于 groupN(o, f) 的长度. 借助回溯算法的剪枝作用, 实际搜索的组合数将远小于 $O(n_2^{k-1})$. 综上所述, 判断候选模式 C 的频繁性的时间复杂度为 $O(k \times n_1 \times n_2^{k-1})$, 由定义 3 易知, n_2 远小于 n_1. 需要注意的是, 随着阶数 k 增大, n_1 和 n_2 都将变小, 这主要得益于引理 2 和 3 的剪枝作用. 在每一轮迭代中, k 为常数; 此外, 因邻近关系和频繁性阈值的约束, 需要搜索的模式的阶数 k 不会无限制增长.

空间复杂度. CPM-Col 算法的空间耗费主要来自两个方面, (1) 保存每个实例的分组邻居集 groupN(o, f), 这部分的空间耗费为 $O(|O| \times |F| \times m_1)$, 其中 m_1 为 groupN(o, f) 的最大长度, 而保存邻近关系的空间消耗在大多数并置模式挖掘算法中都存在; (2) 在挖掘 $k+1$ 阶模式时, 需要额外保存 k 阶频繁并置模式的参与实例集用于计算候选参与实例, 空间消耗为 $O(|P_k| \times k \times m_2)$, 其中 P_k 是 k 阶频繁并置模式的数量, m_2 是参与实例集 Obj(C, f) 的最大长度, 在 $k+1$ 阶候选模式搜索完成后, 这部分空间占用将被释放.

引理5 调用 isPrevalent(C) 过程判断候选模式 C 的频繁性是正确的.

完备性和正确性. CPM-Col 算法从二阶开始逐阶测试每个候选模式的频繁性, 生成候选模式的完备性由参与度的向下闭合性质保证, 因此, CPM-Col 算法一定能搜索到所有的频繁并置模式, 算法是完备的. CPM-Col 算法的正确性由两方面保证, (1) 调用 isPrevalent(C) 过程判断候选模式 C 的频繁性是正确的, 已在引理 5 证明; (2) 算法 1 中步骤 7~9 只保留满足频繁性阈值约束的候选模式.

4.3 CPM-Col 算法 VS 表实例生成

本小节讨论 CPM-Col 算法在验证候选参与实例的过程中搜索的行实例与表实例之间的关系.

定义4 在模式 C 的表实例 table_instance(C) 中, 包含模式 C 中的特征的所有参与实例的行实例集合称为 C 的表实例完备子集.

在图 1(b) 所示的模式 {A, B, C}的表实例中, 虚线框内的 4 条行实例是该模式的表实例完备子

集. CPM-Col 算法在判定模式 C 是否频繁的过程中搜索的行实例可分为两种情况进行讨论. (1) 如果 C 是频繁的, 则 CPM-Col 算法搜索到的行实例是 C 的表实例完备子集, 最坏情况下, 如果完备子集就是整个表实例, 则 CPM-Col 算法也需要搜索 C 的完整表实例. (2) 如果 C 是不频繁的, 则可能在调用 isPrevalent(C) 过程中的两个位置被判定, 如果在步骤 3∼5 被判定为不频繁, 则不需要搜索 C 的任何行实例; 如果在步骤 21∼23 被判定为不频繁, 则搜索的行实例是 C 的表实例完备子集的子集, 最坏情况下, 直到搜索完最后一个特征才判定 C 不频繁, 则搜索的行实例为 C 的表实例完备子集. 通过上述讨论, CPM-Col 算法判定候选模式 C 的频繁性时, 大部分情况下只需要搜索 C 的一部分行实例而不是完整的表实例.

5 实验评估

5.1 实验设置

真实数据集. 在实验中, 使用两个真实数据集[1)], 分别是高黎贡山植被数据集, 包含 10 个特征, 11062 个实例; 北京兴趣点 (points of interest, POI) 数据集, 包含 63 个特征, 303895 个实例.

模拟数据集. 模拟数据集使用类似文献 [12] 的模拟数据生成器生成. 首先, 设置实例的空间分布区域为 $D \times D$ 大小的矩形, 并将整个空间划分为边长 gridL 的正方形网格, 特征数为 F, 实例总数为 N, 将 N 个实例平均分配给 F 个特征, 并将每个特征下的实例进行编号. 然后, 生成平均阶数为 S 的 P 个核模式, 每个核模式的特征从包含 F 个特征的集合中随机挑选. 对每个核模式, 为了保证其频繁度, 平均生成该模式的 I 条行实例, 每条行实例包含的实例分别从该核模式包含的特征的实例集中不重叠地随机选取, 随后将每一条行实例随机放置到一个网格中, 并将该行实例中的实例随机分布到该网格内. 参数 clumpy 控制网格内的数据密度, 在一个网格被选定放置核模式 C 的行实例后, 在该网格内放置 clumpy 个 C 的行实例. 参数 θ 控制同一特征的实例聚集度, 其中, 所有特征的实例聚集度服从均值为 θ 的泊松 (Poisson) 分布, 对于特定的特征 f, 假定其实例聚集度为 $\varepsilon(f)$, 即特征 f 的实例平均与 $\varepsilon(f)$ 个 f 的其他实例邻近, 在空间中放置 f 的实例 o 时, 在 o 的 gridL 邻域内随机放置一定数目的 f 的其他实例, 该数目服从均值为 $\varepsilon(f)$ 的泊松分布. 对未被选中形成行实例的剩余实例, 将其随机分布到整个空间区域中, 并服从每个特征的实例聚集度约束. 所有实例都被放置之后可得到其位置坐标, 即每个实例都包含特征类型、编号、位置三元组信息. 在没有特别说明的情况下, 合成数据参数的默认取值为: $D = 5000$, gridL $= 10$, $F = 50$, $N = 500K$ ($K = 1000$), $S = 5$, $P = 20$, $I = 2000$, clumpy $= 1$, $\theta = 0$.

对比算法. 在实验中, 对比算法包括基于连接的 Joinbase 算法[1]、基于无连接的 Joinless 算法[12]、基于行实例扩展的 NCA 算法[15]、基于树的 CPI-tree 算法[13] 和 iCPI-tree 算法[14]、基于团的 N-clique 算法[16], 以及改编自文献 [7] 的过滤 – 验证机制搜索模式的参与实例的 FV 算法.

实验环境. 所有算法均用 JAVA 语言实现, 所有实验在 Windows 10 操作系统上完成, 硬件环境为 Intel Core i7-7820X CPU @3.60 GHz, 64 GB 内存.

5.2 阈值影响

首先, 在高黎贡山植被数据集和模拟数据集上测试阈值对算法性能的影响, 其中模拟数据集的特定参数 $F = 20$, $N = 200K$, $I = 1000$, $\theta = 3$. 北京 POI 数据集上的实验评估见补充材料.

1) https://github.com/PeizhongYang/SSI-DataSet.

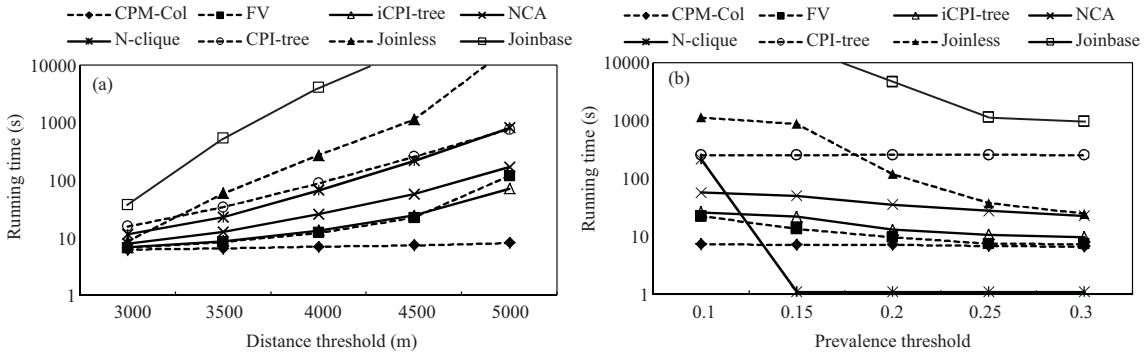

图 2　高黎贡山植被数据集上的运行时间

Figure 2　Running time over the plant dataset of Gaoligong Mountain. (a) By distance thresholds; (b) by prevalence thresholds

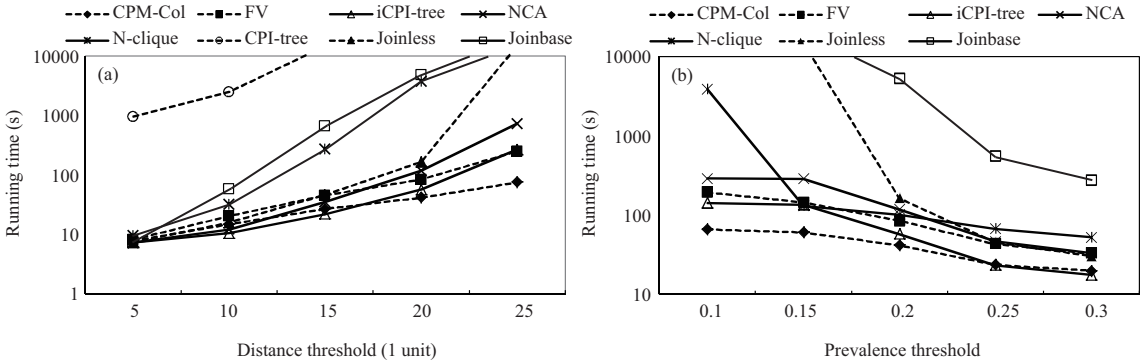

图 3　模拟数据集上的运行时间

Figure 3　Running time over the synthetic dataset. (a) By distance thresholds; (b) by prevalence thresholds

距离阈值影响. 距离阈值主要影响空间邻近关系, 距离阈值越大, 空间邻近关系越多, 越多的实例形成团关系. 图 2(a) 展示了高黎贡山植被数据集上的实验结果, 固定频繁性阈值为 0.1. 随着距离阈值增大, 所有算法的执行时间都在增加. 大多数基于表实例计算的方法由于需要生成候选模式完整的表实例, 所以性能较差, 特别是 Joinbase 和 Joinless 算法. 基于列计算的 CPM-Col 算法具有最好的性能. 图 3(a) 展示了在模拟数据集上的实验结果, 固定频繁性阈值为 0.2. 相似的结论可以被发现, 值得注意的是, CPI-tree 和 N-clique 算法的性能较差, 因为模拟数据集的实例数较大, 所包含的行实例 (或完备团) 数量也较大, 而这两个算法需要识别所有的行实例 (或完备团), 所以性能变差. CPM-Col, NCA, iCPI-tree 和 FV 算法在两个数据集上表现较好且性能稳定.

频繁性阈值影响. 频繁性阈值主要影响候选模式的数量, 随着频繁性阈值增大, 大量的候选模式不频繁. 图 2(b) 展示了高黎贡山植被数据集上的实验结果, 固定距离阈值为 4500 m. 基于候选 – 测试框架的算法 (包括 Joinbase, Joinless, NCA, iCPI-tree, FV 和 CPM-Col 算法) 随着频繁性阈值增大, 运行时间都在减小. CPI-tree 算法对频繁性阈值变化不敏感, 由于该算法不管频繁性阈值如何变化, 都需要生成所有模式的表实例. N-clique 算法在首次运行时较为耗时, 因为需要找到所有完备团并生成 C-hash 表, 随后改变频繁性阈值可以直接通过 C-hash 表快速计算参与度, 所以 N-clique 算法对频繁性阈值变化较友好. 图 3(b) 展示了在模拟数据集上的实验结果, 固定距离阈值为 20 (单位为 1), 其

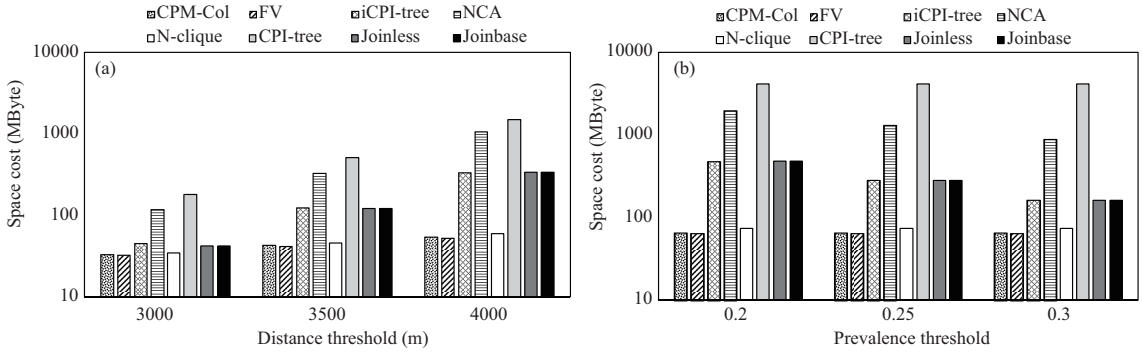

图 4 高黎贡山植被数据集上的空间消耗

Figure 4 Space cost over the plant dataset of Gaoligong Mountain. (a) By distance thresholds; (b) by prevalence thresholds

中, CPI-tree 算法的运行时间均超过 10000 s. 值得注意的是, N-clique 算法在首次运行构造 C-hash 表之后, 随着频繁性阈值改变, N-clique 算法的运行时间仍然多于 CPM-Col 算法, 因为模拟数据集上的特征数和实例数都较大, 生成的完备团的数量也较多, 从而导致 N-clique 算法在计算参与度时需要搜索的 C-hash 表变得很庞大, 因此较为耗时; 而 CPM-Col 算法具有更好的性能是因为其不需要生成表实例并使用了大量剪枝和搜索优化技术. 从该实验结果可以得出相似的结论, 随着频繁性阈值变化, CPM-Col, NCA, iCPI-tree 和 FV 算法都表现较好且稳定, 并且 CPM-Col 算法具有最好的性能.

随着阈值变化, CPM-Col 算法均表现出较好的性能, 特别是在距离阈值较大和频繁性阈值较小的情况下, CPM-Col 算法相对于其他对比算法的效率提升达到数倍至数个量级.

5.3 空间耗费

其次, 在高黎贡山植被数据集上评估所有算法的空间消耗, 实验结果如图 4 所示.

图 4(a) 展示了空间消耗随距离阈值变化的情况, 固定频繁性阈值为 0.1. 随着距离阈值增大, 所有算法的空间消耗均在增加, 因为距离阈值增大导致邻近关系增多, 模式的行实例也在增加. 基于表实例计算的方法由于需要保存模式的表实例, 其空间消耗较大. 其中, CPI-tree 算法由于需要生成所有模式的表实例, 空间消耗最大. NCA 算法需要额外保存行实例的簇结构, 空间耗费也较大. Joinbase, Joinless 和 iCPI-tree 算法的空间消耗基本相当. N-clique 算法将生成的完备团以 C-hash 表进行压缩存储, 空间耗费较小. CPM-Col 和 FV 算法的空间消耗最小, 因为它们不需要生成和保存表实例, 避免了大量的空间消耗.

图 4(b) 展示了空间消耗随频繁性阈值变化的情况, 固定距离阈值为 4500m. CPI-tree 算法依然是空间消耗最大的, 且对频繁性阈值不敏感. Joinbase, Joinless, iCPI-tree 和 NCA 算法的空间消耗随频繁性阈值增大而变小, 因为频繁性阈值增大导致频繁并置模式的数量减少, 候选模式的数量也减少, 需要保存的模式表实例也减少; NCA 算法依然是这 4 个算法中空间耗费最大的. N-clique, FV 和 CPM-Col 算法的空间消耗受频繁性阈值影响较小, 且空间消耗较低, 因为它们不需要保存模式的表实例.

CPM-Col 算法由于不需要生成和保存模式的表实例, 可以避免大量的空间开销. 因此, 在处理数据量较大的空间数据时, CPM-Col 算法比基于表实例计算的算法更有优势.

5.4 可扩展性

本小节测试算法的可扩展性, 为了更好地反映 CPM-Col 算法在数据量较大情况下的性能, 对比算法只选取了效率较高、性能稳定且空间消耗较小的 iCPI-tree 和 FV 算法. 所有实验均在模拟数据集上进行, 其中, 距离阈值与网格边长一致, 频繁性阈值为 0.1.

实例总数影响. 在相同的空间内, 增加实例总数 (N) 将使整个空间的数据密度增大. 如图 5(a) 所示, 随着实例总数增加, 3 个算法的运行时间增加. 对于 FV 和 CPM-Col 算法, 实例总数增加意味着参与实例的搜索空间变大; 而对于 iCPI-tree 算法, 数据密度增大将导致更多的空间邻近关系, 表实例的生成会更耗时. 总的来看, FV 算法效率最差, 因为该算法需要逐个验证候选模式中每个特征的所有实例, 而 CPM-Col 算法由于采用了一些剪枝及优化搜索策略, 效率高于 FV 算法. CPM-Col 算法的性能与 iCPI-tree 算法相当, 由于在生成数据时, 设置参数 clumpy 为 1, θ 为 0, 这导致几乎不存在一个实例重叠在多条行实例中的情况, 即 CPM-Col 算法可能也需要搜索模式的所有行实例才能验证所有的参与实例, 因此相比 iCPI-tree 算法没有优势.

邻域内数据密度影响. 参数 clumpy 控制同一邻域 (网格) 内的行实例数量. clumpy 越大, 网格内的行实例数量呈爆炸式增长, 例如, 当 clumpy 为 5 时, 选定一个网格放置阶数为 6 的模式的行实例, 则该网格内存在的该模式的行实例数量可能达到 $5^6 = 15625$ 个. 图 5(b) 展示了 3 个算法的运行时间随 clumpy 变化的情况. 随着 clumpy 增大, iCPI-tree 算法的运行时间急剧增加; 特别地, 当 clumpy 大于 3 时, iCPI-tree 算法由于保存模式的表实例而耗费巨大的空间, 导致内存溢出. 尽管模式的行实例数量随 clumpy 增大爆炸式增长, 但其参与实例没有改变, 因此 clumpy 变化对于 FV 和 CPM-Col 算法的影响较小. 特别地, CPM-Col 算法由于采用了一些优化策略, 可以在搜索尽可能少的行实例的情况下验证所有的参与实例, 在上述例子中, CPM-Col 算法只需要搜索 5 条行实例就可以验证这一网格内的所有参与实例, 效率提升是显然的. FV 算法不具备这样的优势, 它可能需要搜索更多的行实例来验证网格内的参与实例, 所以效率低于 CPM-Col 算法.

邻域内特征的实例聚集度影响. 参数 θ 控制同一特征的实例在邻域内聚集出现的数量. θ 越大, 同一特征在邻近区域内出现的实例数量越多, 导致这一特征的实例在行实例中重叠出现的概率变大, 相应地, 搜索模式的行实例或参与实例的难度也变大. 图 5(c) 展示了 3 个算法的运行时间受 θ 的影响. 随着 θ 增大, iCPI-tree 和 FV 算法的运行时间明显增加, 而 CPM-Col 算法几乎没受影响. 具体地, 当 θ 小于 3 时, iCPI-tree 算法具有一定的优势, 这是因为 iCPI-tree 算法在搜索行实例时不需要进行团关系检查; 当 θ 大于 3 时, iCPI-tree 算法的运行时间急剧增长, 因为需要搜索的行实例数量增大, 在这一情况下, FV 和 CPM-Col 算法优势明显. 特别地, CPM-Col 算法的性能优于 FV 算法.

CPM-Col 算法在不同情况下其性能保持高效和稳定, 具有较好的可扩展性. 特别是当邻域内数据分布较为稠密以及实例聚集度较高的情况下, CPM-Col 算法的优势更明显.

5.5 算法剖析

接下来, 在北京 POI 数据集上分析 CPM-Col 算法所使用的各种优化策略的有效性, 表 1 展示了 CPM-Col 算法关键步骤的分析指标, 设置距离阈值为 300 m, 频繁性阈值为 0.2.

实例搜索剪枝. 在实例搜索空间剪枝阶段, 核心是缩小候选模式中每个特征的实例搜索范围. 从表 1 的结果看, 基于参与实例集的向下包含性质实现的实例搜索空间剪枝技术可以有效地缩小实例搜索范围, 在当前阈值下, 剪枝率达到 75% 以上, 并且该剪枝技术在每一阶挖掘过程中都发挥作用. 实例搜索空间剪枝是 CPM-Col 算法相对于 FV 算法的主要优势之一, 借助这一技术, CPM-Col 算法不

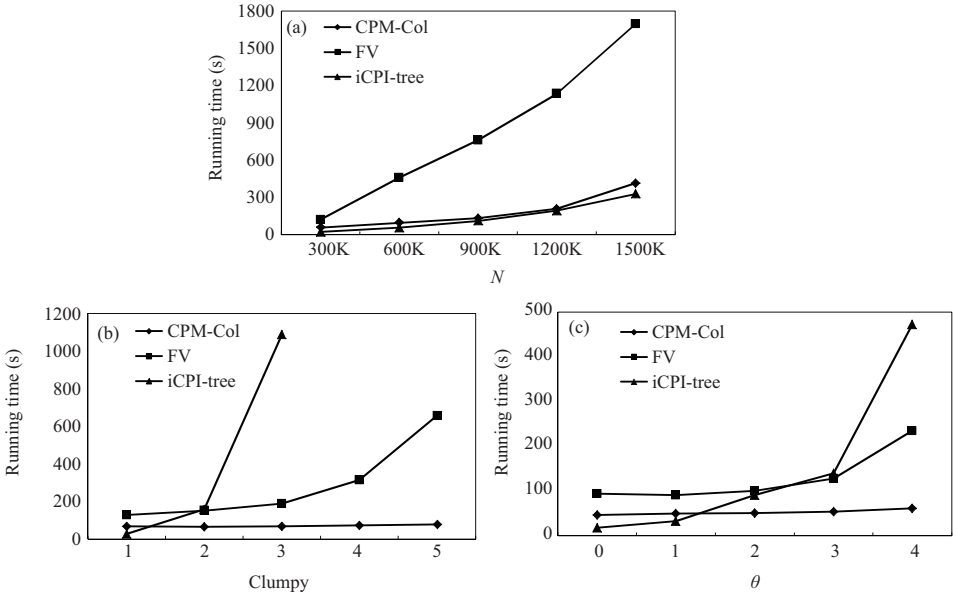

图 5 模拟数据集上的可扩展性实验

Figure 5 Scalability test on synthetic datasets. (a) By the total number of instances; (b) by the data density in the neighborhood; (c) by the degree of instances congregation with the same feature

表 1 CPM-Col 算法剖析

Table 1 Analysis of CPM-Col algorithm

Behavior	Size of candidate patterns				
	Size-3	Size-4	Size-5	Size-6	Size-7
Number of candidate patterns	11982	4521	971	89	2
Pruning ratio of instances (%)	76.55	75.41	75.08	75.50	76.53
Pruning ratio of row instances (%)	86.98	96.70	99.18	99.75	99.93
Average # of SearchRI call per candidate instance	0.290	0.241	0.212	0.198	0.169
Average # of backtracking per SearchRI call	0.09	0.26	0.83	2.40	6.20
Pruning ratio of patterns (%)	82.56	67.37	56.74	52.80	50.00

需要像 FV 算法一样依次判断每个特征下的所有实例.

行实例搜索剪枝. 在候选参与实例验证阶段, 核心的任务是在搜索尽可能少的行实例的情况下完成所有候选参与实例的验证. 从表 1 的结果看, CPM-Col 算法搜索的行实例数量只是模式表实例的很小的子集, 其剪枝率达到 85% 以上. 这主要得益于实例搜索优化策略, 该策略使搜索的每条行实例验证尽可能多的候选参与实例. 由于 CPM-Col 算法只需要搜索模式很少一部分行实例, 其性能相对于大多数基于表实例计算的方法都具有优势.

SearchRI 调用情况. 在候选参与实例验证阶段, 调用 SearchRI 过程搜索行实例验证候选参与实例是最耗时的操作. 从表 1 的结果看, 平均每个候选参与实例调用 SearchRI 的次数远小于 1, 也就是说, 不必每个候选参与实例都调用 SearchRI 过程进行验证. 这主要得益于两方面: (1) 实例搜索优化策略使搜索的每条行实例验证尽可能多的候选参与实例; (2) CPM-Col 算法采用边搜索边剪枝技术, 及时终止对不频繁模式的搜索.

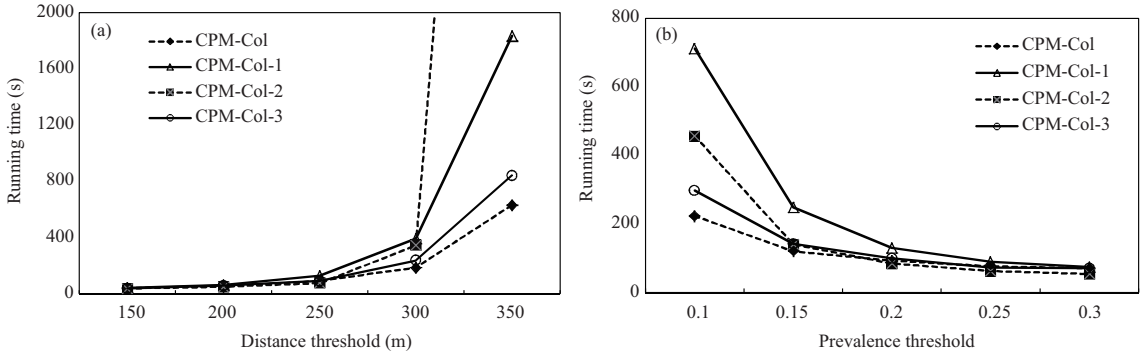

图 6　北京 POI 数据集上的消融测试

Figure 6　Ablation test on the POI dataset of Beijing. (a) By distance thresholds; (b) by prevalence thresholds

调用 SearchRI 发生回溯的情况. 在调用 SearchRI 过程中, 发生回溯的次数是对性能影响较大的因素. 从表 1 的结果看, 平均每次调用 SearchRI 发生回溯的次数较少, 特别是当模式阶数较低的时候. 随着阶数升高, 使用回溯算法搜索行实例的搜索路径加深, 发生回溯的概率增大. 发生回溯次数较少的原因是定义 3 和引理 3 对搜索空间的剪枝作用, 其尽可能使搜索的每个实例组合满足团关系, 因此很多情况下只需要搜索一条路径即可找到满足条件的解.

频繁性提前感知剪枝模式. 频繁性提前感知策略的主要任务是及时发现不频繁的候选模式, 并停止对不频繁模式的参与实例搜索. 从表 1 的结果看, 频繁性提前感知技术在搜索过程中剪掉 50% 以上的不频繁模式, 可以有效地减少不必要的参与实例搜索.

消融测试. 图 6 展示了在北京 POI 数据集上进行消融测试的结果, 图 6(a) 固定频繁性阈值为 0.2, 图 6(b) 固定距离阈值为 250 m. 实验中依次去掉 CPM-Col 算法 3 个核心部件中的一个, 构造 CPM-Col 算法的变体, 其中, CPM-Col-1 不采用实例搜索空间剪枝技术, 需要验证模式每个特征的所有实例; CPM-Col-2 不采用候选参与实例验证技术, 使用 FV 算法的过滤 – 验证机制验证候选参与实例; CPM-Col-3 不使用频繁性提前感知技术. CPM-Col 算法的效率高于其他 3 个变体, 说明 3 个核心部件对效率的提升都是有效的. 特别地, 缺少实例搜索空间剪枝技术的 CPM-Col-1 算法和缺少候选参与实例验证技术的 CPM-Col-2 算法性能较差, 说明这两个技术对 CPM-Col 算法的效率提升贡献较大.

6　总结

空间并置模式挖掘由于可以发现空间特征间的关联关系而具有广泛的应用. 本文分析了现有并置模式挖掘算法所面临的问题, 提出一种新的基于列计算的挖掘方法, 解决了传统的基于表实例计算的方法生成和保存表实例在时间、空间上耗费较大的问题. 为了加速基于列计算的方法对参与实例的搜索过程, 本文研究了一系列搜索优化策略和剪枝技术, 并提出了 CPM-Col 算法. 实验结果证明, CPM-Col 算法相对于其他并置模式挖掘算法具有更好的性能, 其效率提升达到数倍至数个量级, 并有更好的可扩展性. 在未来工作中, 将 CPM-Col 算法扩展成分布式并行算法用于处理海量空间数据是下一步的研究方向.

补充材料　附录 A. 本文的补充材料见网络版 infocn.scichina.com. 补充材料为作者提供的原始数

据, 作者对其学术质量和内容负责.

参考文献

1 Huang Y, Shekhar S, Xiong H. Discovering colocation patterns from spatial data sets: a general approach. IEEE Trans Knowl Data Eng, 2004, 16: 1472–1485

2 Li J D, Adilmagambetov A, Jabbar M S M, et al. On discovering co-location patterns in datasets: a case study of pollutants and child cancers. Geoinformatica, 2016, 20: 651–692

3 Yao X J, Jiang X F, Wang D C, et al. Efficiently mining maximal co-locations in a spatial continuous field under directed road networks. Inf Sci, 2021, 542: 357–379

4 Ge Y, Yao Z J, Li H Y. Computing co-location patterns in spatial data with extended objects: a scalable buffer-based approach. IEEE Trans Knowl Data Eng, 2021, 33: 401–414

5 Hu X, Wang G Y, Duan J L. Mining maximal dynamic spatial colocation patterns. IEEE Trans Neural Netw Learn Syst, 2021, 32: 1026–1036

6 Liu Q L, Liu W K, Deng M, et al. An adaptive detection of multilevel co-location patterns based on natural neighborhoods. Int J Geogr Inf Sci, 2021, 35: 556–581

7 Chan H K, Long C, Yan D, et al. Fraction-score: a new support measure for co-location pattern mining. In: Proceedings of IEEE International Conference on Data Engineering, Macau SAR, 2019. 1514–1525

8 Bao X G, Gu T L, Chang L, et al. Knowledge-based interactive postmining of user-preferred co-location patterns using ontologies. IEEE Trans Cybern, 2021. doi: 10.1109/TCYB.2021.3054923

9 Lu J L, Wang L Z, Fang Y, et al. Mining competitive pairs hidden in co-location patterns from dynamic spatial databases. In: Proceedings of Pacific-Asia Conference on Knowledge Discovery and Data Mining, Jeju, 2017. 467–480

10 Li S L, Zhou J B, Xu T, et al. Competitive analysis for points of interest. In: Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, 2020. 1265–1274

11 Zhang K C, Zhou J B, Tao D L, et al. Geodemographic influence maximization. In: Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, 2020. 2764–2774

12 Yoo J S, Shekhar S. A joinless approach for mining spatial colocation patterns. IEEE Trans Knowl Data Eng, 2006, 18: 1323–1337

13 Wang L Z, Bao Y Z, Lu J, et al. A new join-less approach for co-location pattern mining. In: Proceedings of IEEE International Conference on Computer and Information Technology, Sydney, 2008. 197–202

14 Wang L Z, Bao Y Z, Lu Z Y. Efficient discovery of spatial co-location patterns using the iCPI-tree. Open Inform Syst J, 2009, 3: 69–80

15 Lin Z S, Lim S J. Fast spatial co-location mining without cliqueness checking. In: Proceedings of ACM Conference on Information and Knowledge Management, 2008. 1461–1462

16 Bao X G, Wang L Z. A clique-based approach for co-location pattern mining. Inf Sci, 2019, 490: 244–264

17 Yoo J S, Boulware D, Kimmey D. Parallel co-location mining with MapReduce and NoSQL systems. Knowl Inf Syst, 2020, 62: 1433–1463

18 Andrzejewski W, Boinski P. Efficient spatial co-location pattern mining on multiple GPUs. Expert Syst Appl, 2018, 93: 465–483

A spatial co-location pattern mining approach based on column calculation

Peizhong YANG, Lizhen WANG*, Xiaoxuan WANG & Lihua ZHOU

School of Information Science and Engineering, Yunnan University, Kunming 650504*, China*
* Corresponding author. E-mail: lzhwang@ynu.edu.cn

Abstract Spatial co-location pattern mining aims to discover correlations between spatial features. A co-location pattern corresponds to a subset of spatial features whose instances are frequently located in spatial neighborhoods. Most co-location pattern mining approaches calculate the prevalence based on table instances, but the time and space costs of generating table instances are enormous. To address this problem, this paper presents a novel co-location pattern mining approach based on column calculation, which only searches for participating instances of features in a pattern without having to generate the table instance. Furthermore, this paper designs the instance search space pruning, the candidate participating instance verification, and the prevalence beforehand awareness technologies to speed up the search of participating instances. The CPM-Col method is proposed on this basis, and its complexity, accuracy, and completeness are addressed. Extensive experiments are conducted on real and synthetic datasets, and experimental results show that the CPM-Col algorithm has better performance and scalability than seven other baseline algorithms, especially with a performance gain of several times or even orders of magnitude. Moreover, the effectiveness of the proposed optimization strategies is verified experimentally.

Keywords spatial data mining, co-location pattern, column calculation, search algorithm, pruning technique