软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2024,35(6):2999−3012 [doi: 10.13328/j.cnki.jos.006926]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

面向超图数据的最大独立集算法*

徐兰天[1], 李荣华[1], 戴永恒[2], 王国仁[1]

[1](北京理工大学 计算机学院, 北京 100081)

[2](电科云北京科技有限公司, 北京 100043)

通信作者: 李荣华, E-mail: lironghuabit@126.com

摘　要: 超图是普通图的泛化表示, 在许多应用领域都很常见, 包括互联网、生物信息学和社交网络等. 独立集问题是图分析领域的一个基础性研究问题, 传统的独立集算法大多都是针对普通图数据, 如何在超图数据上实现高效的最大独立集挖掘是一个亟待解决的问题. 针对这一问题, 提出一种超图独立集的定义. 首先分析超图独立集搜索的两个特性, 然后提出一种基于贪心策略的基础算法. 接着提出一种超图近似最大独立集搜索的剪枝框架即精确剪枝与近似剪枝相结合, 以精确剪枝策略缩小图的规模, 以近似剪枝策略加快搜索速度. 此外, 还提出 4 种高效的剪枝策略, 并对每种剪枝策略进行理论证明. 最后, 通过在 10 个真实超图数据集上进行实验, 结果表明剪枝算法可以高效地搜索到更接近于真实结果的超图最大独立集.

关键词: 超图; 最大独立集; 启发式算法

中图法分类号: TP311

中文引用格式: 徐兰天, 李荣华, 戴永恒, 王国仁. 面向超图数据的最大独立集算法. 软件学报, 2024, 35(6): 2999–3012. http://www.jos.org.cn/1000-9825/6926.htm

英文引用格式: Xu LT, Li RH, Dai YH, Wang GR. Maximum Independent Set Algorithm for Hypergraph Data. Ruan Jian Xue Bao/Journal of Software, 2024, 35(6): 2999–3012 (in Chinese). http://www.jos.org.cn/1000-9825/6926.htm

Maximum Independent Set Algorithm for Hypergraph Data

XU Lan-Tian[1], LI Rong-Hua[1], DAI Yong-Heng[2], WANG Guo-Ren[1]

[1](School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China)

[2](Diankeyun Technologies Co. Ltd., Beijing 100043, China)

Abstract: Hypergraphs are generalized representations of ordinary graphs, which are common in many application areas, including the Internet, bioinformatics, and social networks. The independent set problem is a fundamental research problem in the field of graph analysis. Most of the traditional independent set algorithms are targeted for ordinary graph data, and how to achieve efficient maximum independent set mining on hypergraph data is an urgent problem to be solved. To address this problem, this study proposes a definition of hypergraph independent sets. Firstly, two properties of hypergraph independent set search are analyzed, and then a basic algorithm based on the greedy strategy is proposed. Then a pruning framework for hypergraph approximate maximum independent set search is proposed, i.e., a combination of exact pruning and approximate pruning, which reduces the size of the graph by the exact pruning strategy and speeds up the search by the approximate pruning strategy. In addition, four efficient pruning strategies are proposed in this study, and a theoretical proof of each pruning strategy is presented. Finally, experiments are conducted on 10 real hypergraph data sets, and the results show that the pruning algorithm can efficiently search for hypergraph maximum independent sets that are closer to the real results.

Key words: hypergraph; maximum independent set; heuristic algorithm

现实世界中的实体关系大多不能用简单地二元关系来表示, 而超图能更好地表示实体间的多元关系. 超图作为一种比普通图承载更多信息, 也有更强大的表现能力的图表示形式, 有着重要的研究价值. 超图和普通图有明显

* 基金项目: 国家重点研发计划 (2021YFB3301301); 国家自然科学基金 (U2241211, 62072034)
收稿时间: 2022-06-21; 修改时间: 2022-08-28, 2022-11-10; 采用时间: 2023-03-05; jos 在线出版时间: 2023-08-23
CNKI 网络首发时间: 2023-08-28

的区别. 其不同在于每条边不再仅关联两个点, 而可能关联更多的点. 因此原有的普通图上的一些基础定义和独立集定义等需要在超图上重新诠释. 在现有的相关研究中, 更多的图挖掘算法是针对普通图上的, 由于超图与普通图的结构差异, 很多普通图算法无法直接应用于超图.

而最大独立集问题是图算法中最重要的问题之一, 与一些基本的图问题[1−4]密切相关, 如最大诱导子图、最小顶点覆盖、图着色和最大公共边子图等. 例如, 它已被用于计算社交网络覆盖率和到达率[5], 研究无线网络中的最大加权独立集问题, 方便以分层方式组织网络的节点[6]等. 由于计算精确的最大独立集的复杂度非常高, 现有的算法多采用启发式技术来近似计算最大独立集, 在实践中性能良好.

因此本文希望能结合超图的特性和普通图上最大独立集算法的搜索思想, 提出一种新的高效的启发式超图最大独立集算法.

本文的主要贡献在于提出一种新的超图上的独立集和最大独立集的定义, 并基于对超图性质和独立集定义的分析, 提出两种启发式的近似超图最大独立集算法.

本文第 1 节介绍超图和独立集的相关方法和研究现状. 第 2 节介绍本文所需的基础知识, 包括超图独立集和超点度的定义. 第 3 节介绍超图邻接关系的存储格式. 第 4 节介绍超图独立集的性质分析. 第 5 节介绍一种超图最大独立集的基础算法. 第 6 节介绍一种超图最大独立集的改进算法. 第 7 节分析算法的复杂度. 第 8 节通过实验验证所提算法的有效性. 最后总结全文.

1 相关工作

对于超图的特征, 许多学者进行了深入的研究. Lee 等人[7]定义了 26 个超图基序 (h-motif), 以描述 3 个连通超边 (而不是节点) 的连通性模式. 每个连接模式都是由一个独特的 h-motif 描述的, 独立于超边的大小. 文献统计了来自 5 个不同域的 11 个真实世界超图中每个 h-motif 的实例数. 然后, 通过比较 h-motif 在每个超图中的实例数与适当随机的超图中的实例数来衡量每个 h-motif 在每个超图中的显著性. 最后, 计算每个超图的特征轮廓 (定义为每个 h-motif 的归一化显著性向量). 通过比较不同超图的计数和特征轮廓, 可以得出真实世界超图的结构设计原则可由不同 h-motif 的频率捕获, 与随机超图的结构设计原则明显不同. 来自相同域的超图具有相似的特征轮廓, 而来自不同域的超图具有不同的特征轮廓.

文献 [8−10] 不是直接处理超图的复杂性, 而是将超图简化为普通图的集合, 这些普通图包含超图的部分或全部信息, 然后使用定义良好的图度量来识别这些图的性质. 最基本的方法是将一个超图转换为具有相同节点集和相同边集的图, 将每个超边替换为一个团, 称为 2-投影图. 在 2-投影图上, 科研人员研究了单纯闭包, 将三元闭包推广到超图[11,12], 定义了子超图中心性, 并将聚类系数的概念推广到超图[13,14]. 这个简单的抽象被扩展到 n 投影图和 n 级分解图, 它们旨在以不同的方式捕获 n 个节点子集之间的交互. Do 等人[15]提出, 由实超图得到的 n 级分解图保留了 5 种结构模式: 巨连通分量、重尾度分布、小直径、高聚类系数和偏奇值分布. 除此以外, 人们还研究了超边缘的子集相关性、近因偏差和重复模式的动态模式[16]和更多关于扩散和同步的模式[17]. Kook 等人[9]研究了超图的 7 个结构和动力学性质. 并定义了新的度量标准, 将普通图属性的扩展到了超图.

最大独立集问题是图论中经典的组合优化问题, 在生物信息学, 网络通信, 传输调度以及社交网络分析等领域有着非常广泛的实际应用. Carraghan 等人[18], Bable 等人[19], Östergård 等人[20], Li 等人[21]提出了几种精确算法, 其核心都是基于分支界定策略. 这种方法通过探索所有可能的情况来获得精确解. 每个顶点 u 可以包含在独立集中, 也可以不包含在独立集中. 则对于边 (u, v) 有 3 种情况, 选择 u, v 或者都不选[22]. 枚举所有的可能, 最终获得精确解. 此外还可利用动态规划、染色、覆盖等方法加速剪枝的过程, 可以提高算法的速度[23]. 尽管如此, 在图上精确求解最大独立集始终是个 NP-Hard 问题[24], 并且图的规模越大, 最大独立集算法的时间复杂度越高, 这使得精确求解是不实际的.

因此许多学者开发了启发式的近似算法, 这些算法虽不能保证得到精确的最大独立集, 但其搜索速度快, 在解决规模较大的图上的最大独立集搜索问题时具有极高的性价比. 目前应用比较广泛的启发式算法有贪婪算法[25],

局部搜索[26]和禁忌搜索[27−29]等. 贪婪算法按一定的顺序逐次扩充解的集合, 直至顺序搜索结束, 最终获得一个可行解. Liu 等人[30]提出了逐步地添加最小度数的顶点到最大独立集中的解决方案. Lamm 等人[31]提出了优先选择更可能加入最大独立集的点的进化算法. 局部搜索对可行解的局部邻域进行搜索, 来提高已知解的质量. Andrade 等人[32]提出的局部搜索算法, 用已获得的独立集中的点与非独立集点交换来扩展最大独立集大小. Dahlum 等人[33]提出了局部搜索与进化规则相结合的算法. 禁忌搜索是一种改进的局部搜索算法, 通过设立“禁忌列表”来避免循环搜索, 可以快速跳出循环并搜索新的区域. Gao 等人[34]提出了在动态图上维护最大独立集的近似算法, Liu 等人[35]基于 Pay-and-Recycle 模型, 提出了包含指定点集的类最大独立集算法. 此外, 在前人的研究中, 还提出了许多有效的剪枝规则, 如折叠和镜像规则, 二度路径剪枝规则等[36,37]. 这些剪枝策略有助于在不牺牲解决方案质量的情况下减少问题的规模.

然而, 目前国内外还没有出现如本文定义的超图最大独立集的搜索算法, 这也说明本文的研究是十分有意义的.

2 基础知识

假如 $G=(V, E)$ 是一个超图, $V(|V|=n)$ 表示图 G 对应的点集, $E(|E|=m)$ 表示图 G 对应的边集.

定义 1. 超图独立集和最大独立集. 超图独立集 S 是超图 G 中的一个超点子集, 使得超图独立集 S 中的任意两个不同的超点都不能被超图 G 中的同一超边所包含. 如果一个超图独立集的点数是图 G 中最多的, 则称为超图 G 的一个最大超图独立集. 用 $\alpha(G)$ 表示超图 G 最大独立集中的超点个数.

如图 1 所示, 图中有 3 条超边, 超边 a 包含点 0, 1, 2; 超边 b 包含点 2 和 3; 超边 c 包含点 0 和 3. 根据本文超图独立集定义, 图中共有 5 个独立集, 分别为{0}, {1}, {2}, {3}, {1, 3}, 其中独立集{1, 3}是点数最多的独立集, 所以是最大独立集.

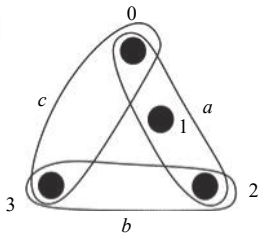

图 1 超图独立集

定义 2. 超边的大小和超点的度. 一条超边 e 的大小是超边 e 中包含的超点的个数, 记为 $size(e)$. 一个超点 u 的邻接边集包含所有超点 u 参与形成的超边, 记为 N_u. 一个超点 u 的度是超点 u 参与形成的超边的数量, 记为 $degree(u)=|N_u|$. 超点的平均度表示为 D, 超边的平均大小表示为 S.

3 超图邻接关系存储

本文采用改进的压缩存储格式 (improved compressed storage format, ICSF)[38]来保存超图的点边邻接关系和边点邻接关系. 图 2 给出了图 1 对应的 ICSF 存储格式, 这种邻接关系存储的格式没有用到指针和链表, 避免了复制的指针操作和链表的维护工作, 结构比较简单, 方便维护, 并有效地避免了存储空间的浪费.

在 ICSF 存储格式中, 共包含 xadj、adjncy、eptr、eind 这 4 个数组. 其中 xadj 和 adjncy 保存超点对于超边的邻接关系, eptr 和 eind 保存超边对于超点的包含关系. xadj 数组的长度为超图中的超点总数, 其保存的是 adjncy 数组的序号. 其中 xadj[i] 保存 i 号超点在 adjncy 数组中对应的起始位置. adjncy 数组的长度为所有超点的邻接超边的个数之和, 其中保存的是超边序号. eptr 数组的长度为超图中的超边总数, 其保存的是 eind 数组的序号. 其中 eptr[j] 保存 j 号超边在 eind 数组中对应的起始位置. adjncy 数组的长度为所有超边中包含的超点的个数之和, 其中保存的是超点序号.

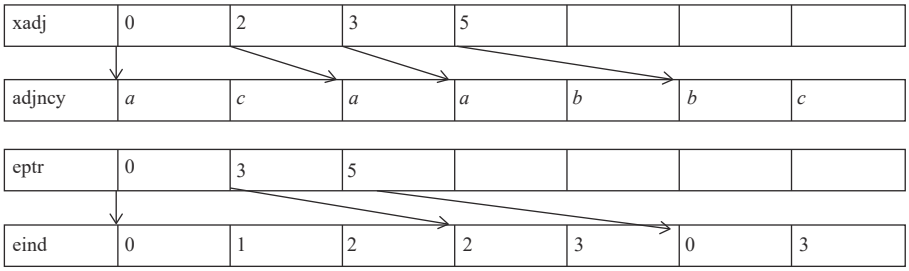

图 2 存储结构

假设超点序号从 0 开始, 则第 i 个超点的邻接超边的序号保存于 adjncy[xadj[i]] 到 adjncy[xadj[i+1]–1], 第 i 个超点的度为 xadj[i+1]–xad[j]; 类似的, 假设超边序号从 0 开始, 则第 j 条超边中包含的超点保存于 eind[eptr[j]] 到 eind[eptr[j+1]–1], 第 j 条超边的大小为 eptr[j+1]–eptr[j].

如图 2 所示, 2 号超点参与组成了 a 和 b 两条超边, 则在数组 adjncy 中, 2 号超点的邻接边序号从 xadj[2]=3 开始, 到 xadj[2+1]–1=4 结束. 类似的, a 号超边 (在数组 eptr 中用 0 号超边来表示) 包含了 0, 1 和 2 这 3 个超点, 则在数组 eind 中, a 号超点的邻接边序号从 eptr[0]=0 开始, eptr[0+1]–1=2 结束.

4 超图独立集的性质分析

首先本文基于超图的性质和超图独立集的定义, 提出以下两个性质.

性质 1. 对于同一超边上的 n 个点, 最多只能选择一个点加入独立集.

证明: 根据本文超图独立集定义, 一个独立集中任何两个超点都不能同时参与形成同一条超边, 则对于一个超点 $u\in e$, 若超点 u 是一个超图独立集中的一个点, 假设有另一超点 $v\in e$ 也是该超图独立集中的一个点, 这对于超边 e, 它存在两个超点 u, v 都在独立集中. 显然这与定义不相符, 所以假设不成立.

性质 2. 对于同一条超边上的点, 与其选择该超边上与其他超边有邻接关系的点加入最大独立集, 不如选择该超边上与其他超边没有有邻接关系的点. 一旦选择了该超边上与其他超边有邻接关系的点, 那么其他超边上的点就也不能选了.

证明: 假设对于超图 G, 其最大独立集为 S. 则对于一条超边上的两点 u, $v\in e$, 若 $degree(u)>1$, $degree(v)=1$. 则 $\exists e'\in E$, $e'\neq e$, $u\in e'$. 若选择超点 u 加入独立集 S, 则由超边 e 和 e'中的其他超点就都不会再有机会加入独立集了. 此时 $\alpha(G)=\alpha(G\backslash e\backslash e')+1$. 若选择超点 v 加入独立集 S, 则只有超边 e 中的其他点不能加入独立集. 此时超图独立集大小 $\alpha(G)=\alpha(G\backslash e)+1$. 显然 $\alpha(G\backslash e)\geq\alpha(G\backslash e\backslash e')$. 由此可知, 选择超点 v 的结果不差于选择超点 u, 则性质 2 得证.

5 超图最大独立集的基础算法

首先, 本文提出一种比较简单的基于贪心策略的超图最大独立集算法, 如算法 1 所示, 该算法主要根据性质 1 的思想来求解超图最大独立集.

算法 1. 超图独立集基础算法.

输入: 超图 $G=(V,E)$;
输出: 超图 G 的最大独立集.

1. **for** $v\in V$ of G **do** //初始化状态标记
2. $State[v]=INITIAL$
3. **for** $v\in V$ of G **do** //遍历 $INITIAL$ 超点
4. **if** $State[v]=INITIAL$ **then**
5. $State[v]=IS$

6. **for** $u \in N_v$ **do** //标记 ~ *IS* 超点
7. **if** *State*[*v*] = *INITIAL* **then**
8. *State*[*v*] =~ *IS*
9. **return** 所有状态为 *IS* 的超点

首先遍历图 *G* 的所有超点 *v*, 为所有的超点赋予一个状态值 *State*[*v*]. 初始时, 所有的超点的状态都为 *INITIAL*. 然后遍历超图 *G* 中的所有超点, 若当前访问的超点 *u* 的状态为 *INITIAL*, 则将当前被访问超点 *u* 的状态置为 *IS*, 同时遍历超点 *u* 的所有的邻接超边中包含的所有超点, 如果这些点状态为 *INITIAL*, 则将其置为~*IS*. 重复本操作, 直到超图 *G* 中的所有超点的状态都不为 *INITIAL*, 此时输出所有状态为 *IS* 的超点, 即为算法搜索到的一个近似的超图最大独立集.

算法完成后的所有的点的状态都为 *IS* 或~*IS*. 对于所有状态为 *IS* 的点, 每一轮循环中被置为 *IS* 的点都不会与在他之前就已被置为 *IS* 的点有共同的超边. 因为如果存在这样的超点, 则该超点被枚举为 *IS* 时, 他的邻接超点已被置为 *IS*, 失去了再被置为 *IS* 的机会. 因此算法结束后所有的 *IS* 状态的点一定是一个超团独立集. 对于所有状态为~*IS* 的点, 每一个~*IS* 状态的点, 至少存在一个与他相邻的点已被置为 *IS*, 否则这个点就不会被置为~*IS*. 则此时~*IS* 状态的点都不能再加入当前所有 *IS* 状态的点组成的超图独立集.

然而, 这种贪心的搜索策略可能在一些情况下将非最优的超点加入独立集. 如图 3 所示, 原图中共有 4 个点, 且 4 个点的度都为 3. 按照算法 1 枚举图中所有超点, 假如从超点 1 开始枚举, 则超点 1 的状态先被置为 *IS*, 此时搜索超点 1 的所有邻居超点, 即 2, 3, 4 这 3 个超点. 将 3 个超点的状态都置为~*IS*, 此时图中所有点的状态已经都不为 *INITIAL*, 则可输出本次超图最大独立集搜索结果, 即{1}.

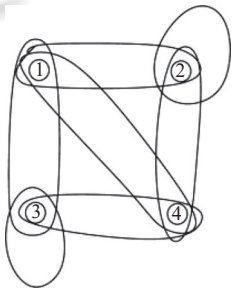

图 3 基础算法的异常示例

可是, 如若改换算法 1 第 3 行的超点遍历顺序, 首先遍历超点 2. 则超点 2 的状态先被置为 *IS*, 此时搜索超点 2 的所有邻居超点, 即 1, 4 两个超点, 则将这两个超点的状态都置为~*IS*. 此时超点 3 的状态仍为 *INITIAL*, 则第 2 次循环遍历超点 3, 搜索超点 3 的所有邻居超点, 即 1, 4 两个超点. 但超点 1, 4 的状态在上一轮循环中都已被置~*IS*, 则本轮不再有 *INITIAL* 状态的超点, 则循环直接结束. 此时可输出本次超图最大独立集搜索结果, 即{2, 3}.

显然第 2 次执行算法搜索到的独立集的大小大于第 1 次搜索到的独立集的大小. 这主要是因为算法 1 中超点的遍历顺序不同导致的, 一种好的遍历次序可以让算法 1 找到更接近于真实结果的超图最大独立集.

6 超图最大独立集的改进算法

本节中, 本文改进了 2-uniform 图上的剪枝方式[37], 提出了超图近似最大独立集的搜索框架, 3 种精确剪枝和一种近似剪枝策略. 如算法 2 所示, 给出了超图近似最大独立集的搜索框架.

算法 2. 超图独立集剪枝框架.

输入: 超图 *G* 以及若干精确和近似剪枝;
输出: 超图 *G* 的最大独立集.

1. **while** $\exists u \in V$, $degree(u) \geqslant 1$ **do**
2. **if** 对超点 u 可应用精确剪枝 **then**
3. 对超点 u 应用精确剪枝
4. **else**
5. 对超点 u 应用近似剪枝
6. **return** 所有剩余超点组成最大独立集

在算法 2 超图独立集剪枝框架中, 首先需要输入原始超图和若干精确剪枝和近似剪枝策略. 然后不断枚举超图 G 中的超点, 可采用精确剪枝时, 优先使用精确剪枝; 精确剪枝不能使用时, 采用近似剪枝. 近似剪枝后, 可能又会出现可应用近似剪枝的条件. 这样近似剪枝与精确剪枝相结合, 直到最后精确剪枝和近似剪枝都不能在应用, 此时所有剩余的超点的度都不大于 1. 则所有剩余的超点可以组成一个近似的最大超图独立集.

下面本文结合超图的特性和本文中超图最大独立集的定义, 提出以下几种剪枝策略.

● 剪枝 1. one-degree 点精确剪枝. 对于原始超图 G 中的所有的度为 1 的超点 u, 删除超点 u 及其所在的整条边 e 后, 原图 G 中的最大独立集的大小减少 1. 即$\forall u \in V$, 若 $degree(u)=1$, 而且 $u \in e$, $|\alpha(G\backslash e)|=|\alpha(G)-1|$. 换个角度说, 即可将超点 u 加入最大独立集中, 然后删掉超点 u 所在的超边 e, 此时, 假设 $G\backslash e$ 中的最大独立集为 S', 则原图 G 一定存在一个最大独立集为 $S' \cup \{u\}$.

one-degree 点剪枝的具体情况由图 4 所示, 图中绿色超点为 one-degree 超点. 若在该图上应用剪枝 1, 则可将绿色超点直接加入最大独立集中, 同时删除绿色超点所在超边及边上所有点.

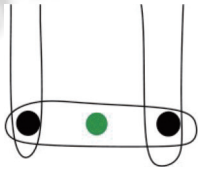

图 4 one-degree 点剪枝

证明: (1) 证明选择 one-degree 超点 u 加入最大独立集不会改变最大独立集大小.

假设 $G'=G\backslash e$, $\alpha(G\backslash e)$ 为图 G' 上的最大独立集大小. 由于 one-degree 点只存在于超边 e 中, 则$\forall v \in \alpha(Ge)$, 超点 u 和超点 v 不能共同存在于同一超边, 则由性质 1, 一条超边最多能有一个超点加入最大独立集, 可得 $\alpha(Ge) \cup \{u\}$ 为原图 G 的一个最大独立集.

(2) 证明选择 one-degree 超点 u 加入最大独立集不差于选择超边 e 中的其他点.

对于 $v \in e$, $u \neq v$, 有两种情况, 即 $degree(v)=1$ 或 $degree(v) \geqslant 1$.

① 若 $degree(v)=1$, 则超点 v 和超点 u 一样都是 one-degree 超点, 本质上将超点 v 和超点 u 中的任意一个超点加入最大独立集都是可以的, 因此选择超点 u 加入最大独立集和选择超点 v 加入最大独立集一样好.

② 若 $degree(v)>1$, 超点 v 与其他超边还有邻接关系, 则由性质 2, 选择超点 u 加入最大独立集一定不比选择超点 v 加入最大独立集差.

● 剪枝 2. high-degree 近似点剪枝. 在图 G 中, 如果一个高度超点 u ($degree(u)>1$) 的 $degree(u)$ 条邻接边都不是只包含 u, 则可删掉超点 u. 高度超点的删除次序应为所有超点 $degree$ 值降序.

high-degree 近似点剪枝的具体过程如图 5 所示, 图中绿色超点为 high-degree 超点. 若在该图上应用剪枝 2, 则可将绿色超点直接删除, 同时遍历绿色超点的所有邻接边, 所有超边的 $size$ 大小减 1.

证明: high-degree 近似点剪枝这种不精确的剪枝规则是基于高度的超点不太可能在一个最大独立集中的直觉设计的; 例如, 如果一个高度的超点被加入独立集合中, 那么它所有的邻接点就会失去加入超图最大独立集机会. 此外, 对于一个不能应用其他精确剪枝的图, 通常在删除一个或几个高度的超点后, 可以重新应用其他精确剪枝规则.

对于两个 high-degree 超点, 在使用 high-degree 近似点剪枝时应优先删掉其中度数更高的超点, 因为高度超点

被删除后, 会影响其相邻的超边, 进而影响其相关的超点. 在这一过程中, 还可能较低度超点的邻接边的关系发生变化, 无需再使用 high-degree 近似点剪枝.

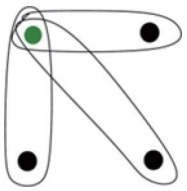

图 5 high-degree 点剪枝

● 剪枝 3. one-size 边精确剪枝 *a*. 在图 G 中, 如果一个高度的超点 u($degree(u)$>1) 的 $degree(u)$ 条邻接边中, 一部分只包含超点 u, 一部分不是只包含超点 u, 则删掉超点 u 的只包含超点 u 的邻接超边, 图 G 的最大独立集大小不变.

one-size 边精确剪枝 a 的具体过程由图 6 所示, 图中绿色超点为 high-degree 超点, 该点的 degree 为 3. 该点的 3 条邻接超边中有两条超边不是只包含绿色超点, 有一条超边只包含绿色超点, 这条边的 *size* 为 1. 若在该图上应用剪枝 3, 则可直接删除这条 *size* 为 1 的超边.

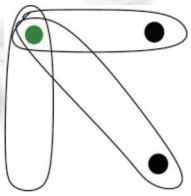

图 6 one-size 边剪枝 a

证明: one-size 的超边只包含一个超点 u, 他并没有提供与超点 u 相关的邻接信息. 删掉超点 u 的只包含 u 的部分邻接边并没有影响点 u 和图 G 中其他超点的邻接关系, 所以对原图的最大独立集大小没有影响.

● 剪枝 4. one-size 边精确剪枝 b. 在图 G 中, 如果一个高度的超点 u($degree(u)$>1) 的 $degree(u)$ 条邻接边都是只包含超点 u, 则删掉超点 u 的任意 $degree(u)$–1 条邻接边, 图 G 的最大独立集大小不变.

one-size 边精确剪枝 b 的具体过程由图 7 所示, 图中绿色超点为 high-degree 超点. 该点的 3 条邻接超边都只包含绿色超点一个点, 他们的 *size* 均为 1. 若在该图上应用剪枝 4, 则可删除其中的两条超边.

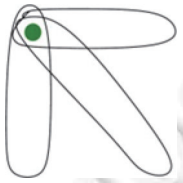

图 7 one-size 边剪枝 b

证明: 删掉点 u 的只包含 u 的邻接边并没有影响点 u 和图 G 中其他点的邻接关系, 所以对原图的最大独立集大小没有影响. 然而需要注意的一点是, 不能删除所有 *size* 为 1 的超边. 因为, 若删掉所有 *size* 为 1 的超边, 超点 u 也会被删除.

综合 4 种剪枝策略, 下面提出一种改进的超图独立集算法, 即算法 3. 算法第 3–6 行遍历所有度为 1 的超点, 通过遍历 1 度点的邻接边, 删除邻接边上的所有超点, 将遍历到的度为 1 的超点加入超图独立集 H. 之后按超点度数从高到低访问所有超点, 对于遍历到的一个超点 v, 如算法 3 的第 9–11 行所示, 若他的所有邻接边的 *size* 都为 1, 则所有邻接超边都只包含超点 v 这一个超点, 则可应用 one-size 边精确剪枝 b, 删除 N_v 中的所有超边, 但注意要

剩一条边. 如算法 3 的第 12, 13 行所示, 若它的所有邻接超边的 *size* 全部大于 1, 则所有邻接超边都不只包含超点 v, 则可应用剪枝 2, 删除遍历 N_v 中的所有超边, 从这些超边的包含点中删除超点 v, 并更新这些超边的 *size*. 如算法 1 的第 14–16 行所示, 若它的所有邻接边的 *size* 中有部分为 1, 则部分邻接超边只包含超点 v 这一个超点, 则可应用剪枝 2, 删除 N_v 中的所有 *size* 为 1 超边, 之后更新超点 v 的度, 同时更新超点 v 在 T_G 中的位置. 当所有剩余的超点的 *degree* 都不大于 1 时, 遍历所有剩余的超边, 此时所有剩余的超边彼此之间没有邻接关系, 都是由若干度为 1 的超点组成的. 则此时在每条剩余超边中选取一个点加入超图独立集, 与应用剪枝 1 时已加入超图独立集的点合并, 得到最终的超图最大独立集.

算法 3. 超图独立集改进算法.

输入: 超图 $G=(V,E)$;
输出: 超图 G 的最大独立集.

1. $N_u=\{e\subseteq E\mid u\in e\}, H=\varnothing$
2. T_G 是 $v\in G$ 的按 *degree* 降序序列
3. **for** $v\in V$ **do** //剪枝 1
4. **if** $degree[v]=1$ **then**
5. 从 E 中删除 N_u
6. $H=H\cup\{v\}$
7. **while** $\exists v\in V$, $degree[v]>1$ **do**
8. 从 T_G 中取最高度超点 v
9. **if** $\forall e\in N_v$, $size(e)=1$ **then** //剪枝 4
10. 从 E 中删除 N_u, 但要留一条超边
11. $degree(v)=1$ 并更新 T_G
12. **else if** $\forall e\in N_v$, $size(e)>1$ **then** //剪枝 2
13. 删掉 v 并更新 N_v 中边的 *size*
14. **else** //剪枝 3
15. $degree(v)-=\left|N_{v\&size(e)=1}\right|$
16. 删除 $N_{v\&size(e)=1}$ 更新 T_G
17. **for** $e\in E$ **do** //遍历剩余超边
18. u 为 e 中的一个超点
19. $H=H\cup\{u\}$
20. **return** H

以图 8 为例, 图中超图共有 10 个超点, 8 条超边. 度为 1 的超点有 3 个, 分别为超点 2, 4, 6; 度为 2 的超点有 4 个, 分别为超点 3, 5, 9, 0; 度为 3 的超点有 4 个, 分别为超点 3, 5, 9, 10. 若使用算法 3 处理该图, 首先枚举图中的度为 1 的超点, 即 3 个红色超点. 根据 one-degree 点精确剪枝规则, 可将 2, 4, 6 这 3 个点加入超图独立集, 同时删除这 3 个超点所在的边. 这个操作会导致图中 4 个绿色超点, 即 1, 3, 5, 7 被删除. 上述操作完成后, 原图中还剩下 8, 9, 0 这 3 个点. 按照 *degree* 由高到低次序访问, 首先访问度为 3 的蓝色超点 8. 超点 8 有 3 条邻接超边, 其中 1 条 *size* 为 1, 2 条 *size* 为 2, 则可应用 one-size 边精确剪枝 a, 将超点 8 的 *size* 为 1 的邻接超边删除. 此时原图中依然还剩下 8, 9, 0 这 3 个点, 但 3 个点的 *degree* 都是 2. 若再次取最高度节点取到了超点 8, 此时超点 8 有两条邻接超边, *size* 均为 2, 则可应用 high-degree 点近似剪枝, 删除蓝色超点 8. 接下来, 再次取最高度节点取到了超点 9, 此时超点 9 有两条邻接超边, 其中一条 *size* 为 1, 另一条 *size* 为 2, 则可应用 one-size 边精确剪枝 a, 将超点 9 的 *size* 为 1 的邻接超边删除. 之后再次取最高度超点 0, 与超点 9 相同, 也可应用 one-size 边精确剪枝 a, 将超点 0 的 *size*

为 1 的邻接超边删除. 此时, 图中仅剩余一条由超点 0 和超点 9 组成的超边, 且已经没有度大于 1 的超点. 此时可任取超点 0 和超点 9 中的一个加入 H. 算法结束, 最终得到的近似超图最大独立集为{2, 4, 6, 0}或{2, 4, 6, 9}.

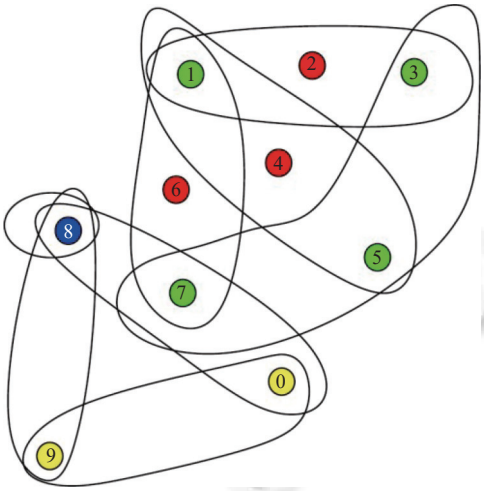

图 8 改进算法执行流程

7 算法分析

对于基础算法, 算法需要枚举所有 *INITIAL* 状态的超点, 并遍历修改他们邻接点的状态. 枚举所有 *INITIAL* 状态的超点时, 前面被枚举的超点可能会影响后面超点的状态, 实际上这一步的枚举次数是少于 n 的. 而对于超图中的一个 *INITIAL* 状态的超点 v, 他的平均邻接点的数量为 $D\times S$. 则在最坏情况下, 算法 1 的时间复杂度不超过 $O(n\times D\times S)$.

对于改进算法, 该算法需要计算按超点度的降序序列, 这一步的时间复杂度为 $O(n\log_2 n)$. 然后算法执行 one-degree 点剪枝, 需要遍历所有的度为 1 的超点, 并删除该超点所在的超边, 及边上的所有点. 则对于每个度为 1 的超点, 它所在的超边的大小平均为 S, 对于边上的每个超点, 要遍历平均 D 条邻接边, 并遍历这些邻接边, 从中删除相关的超点还需要再遍历 S 次. 则 one-degree 点剪枝最终的最坏时间复杂度为 $O(n_{degree=1}\times D\times S^2)$.

接下来, 考虑对于 high-degree 超点的剪枝, 剪枝 2 与剪枝 3 和剪枝 4 是一个相辅相成的组合体. 对于两种 one-size 边剪枝, 需要删除高度超点的邻接边中 *size* 为 1 的边, 对于每个高度超点, 删除邻接边最多需要查找 D 次. 而对于 high-degree 近似点剪枝, 需要遍历高度点的平均 D 条超边, 并从每条平均长度为 S 的超边中删除高度点, 需要的时间复杂度为 $O(D\times S)$. 由于精确剪枝的复杂度低于近似剪枝, 则在对于整个 high-degree 超点的剪枝过程, 最坏时间复杂度为 $O(n_{degree>1}\times D\times S)$.

改进算法最后需要遍历所有剩余超边, 最坏情况下会遍历 m 次. 则对于整个算法的时间复杂度为 $O(n\log_2 n+n_{degree=1}\times D\times S^2+n_{degree>1}\times D\times S)$.

在空间复杂度方面, 基础算法和改进算法的空间复杂度区别不大, 两者都是用 ICSF 存图, ICSF 空间大小为 $O(n\times D+m\times S)$. 与基础算法相比, 改进算法只是多了一个维护 *degree* 排序的序列, 因此改进算法空间复杂度为 $O(n\times D+m\times S+n)$.

8 实验分析

本节进行大量实验来证明改进的超图近似最大独立集算法的有效性. 本章实验环境硬件为 Intel(R) Xeon(R) Gold5218R CPU@2.10 GHz, 内存为 96 GB, 硬盘为 500 GB, 操作系统为 Ubuntu 9.4.0, 开发软件为 g++, 开发语言

为 C/C++. 如表 1 所示, 展示了本节使用的 10 个真实世界的数据集[10], 这些数据集的规模大小, 稠密程度各不相同. 这些数据集均可则在 https://www.cs.cornell.edu/~arb/data/下载.

表 1 数据集

数据集	简称	$\|E\|=m$	$\|V\|=n$	S	D	数据集	简称	$\|E\|=m$	$\|V\|=n$	S	D
contact-primary-school	Primary	106 879	242	2.09	922	DAWN	DAWN	2 272 433	2 558	1.58	1 406
contact-high-school	High	172 035	327	2.05	1 076	NDC-substances	NDC	112 405	5 556	1.85	38
email-Eu	Email	234 760	1 005	2.33	544	tags-stack-overflow	Stack	14 458 875	49 998	2.96	859
congress-bills	Bills	260 581	1 718	3.66	555	coauth-MAG-Geology	MAG	1 590 335	1 261 129	2.77	4
tags-math-sx	Math	822 059	1 629	2.19	1 106	coauth-DBLP	DBLP	3 700 067	1 930 378	2.78	5

下面展示不同超图近似最大独立集算法在各数据集上的效果. 其中 BA 对应算法 1 是基于贪心的基本算法, MA2 对应算法 3, 是结合了 4 种剪枝策略的改进近似算法, MA1 表示在算法 3 中不执行 3–6 行, 即不采用 one-degree 点剪枝, 只采用其他 3 种剪枝的算法.

表 2 展示了 3 种算法在 10 个超图数据集上搜索出的近似超图最大独立集的点数情况. 从表中可以看出, 在所有的 10 个超图数据集中, MA1 算法和 MA2 算法查找到的超图最大独立集结果均优于 BA 算法. 而 MA1 和 MA2 两个算法相比, 其找到的最大独立集的点数则区别不大. 当超图数据集比较小时, 找到的超图独立集的总点数也比较少, 此时两种算法的结果相同或相差极小, 如数据集 contact-primary-school, contact-high-school, email-Eu 和 congress-bills. 然而当超图数据集比较大时, 找到的超图独立集的总点数也比较多, MA2 算法找到的最大独立集明显优于 MA1 算法, 例如在数据集 coauth-DBLP 中, MA1 找到的最大独立集的大小为 949 239, 而 MA2 算法为 952 366. 而且从图中可以看出, 超图独立集的总点数越多, MA2 算法的结果优势越明显.

表 2 最大独立集点数

数据集	BA	MA1	MA2
contact-primary-school	12	16	16
contact-high-school	38	45	45
email-Eu	229	300	299
congress-bills	41	73	76
tags-math-sx	413	519	522
DAWN	1 298	1 417	1 411
NDC-substances	2 825	3 067	3 074
tags-stack-overflow	16 686	23 242	23 252
coauth-MAG-Geology	525 316	627 979	629 283
coauth-DBLP	751 018	949 239	952 366

图 9 展示了 3 种算法在 10 个超图数据集上搜索出的近似超图最大独立集的运行时间及内存占用. 从图中可以看出, 超图数据集中的点数越多, 平均超边大小越大, 点的平均 *degree* 越大, 则算法完成搜索超图近似最大独立集的时间越长. 此外, BA 算法在所有数据集上的搜索时间都是最短的, 即使是对于数据规模最大的 coauth-DBLP 数据集, 仍然可以在 0.5 s 左右完成搜索. 而 MA1 算法和 MA2 算法的运行时间则明显长于 BA 算法. 在超图的规模较小时, 如数据集 contact-high-school, email-Eu 和 congress-bills, MA1 算法快于 MA2 算法. 而在超图的规模较大时, 如数据集 tags-stack-overflow, coauth-MAG-Geology 和 coauth-DBLP, MA2 算法用时则明显少于 MA1 算法. 甚至对于数据集 coauth-DBLP, MA2 算法只需要 310 s, 而 MA1 算法需要 573 s, MA2 比 MA1 算法快了一倍. 在内存占用方面, BA 算法只需保存超边和超点间的邻接关系, 其内存占用很小. 而 MA1 和 MA2 算法加入了新的数据结构, 需要多维护一个所有超点按 *degree* 降序的有序序列, 其内存占用约为 BA 算法的一倍左右. 而 MA1 和 MA2 这两种改进算法相比, 由于采用了相同的数据结构, 其内存占用几乎相同.

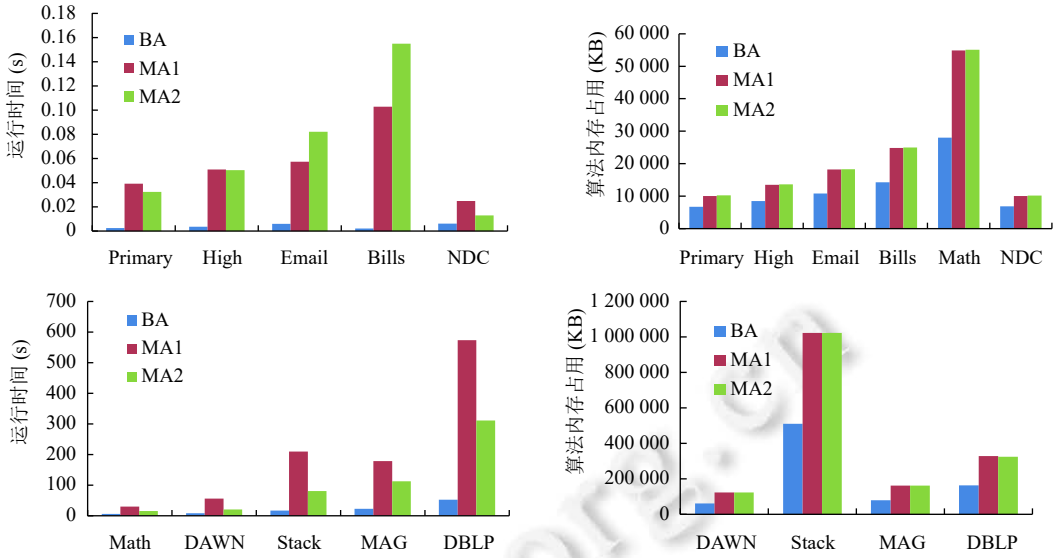

图 9 运行时间及内存占用

表 3 展示了 4 种剪枝策略在 10 个超图数据集上的应用效果. 剪枝 1 在较为稠密的图上的效果有限, 比如 contact-primary-school 和 contact-high-school, 这两个图中没有 one-degree 点, 则剪枝 1 失效. 而在较为稀疏的 coauth-DBLP 和 coauth-MAG-Geology 图上, 剪枝 1 可以在不损失准确度的情况下, 快速将 one-degree 点加入独立集并将包含 one-degree 点的边删除来减小后续的搜索空间. 剪枝 2 是 4 种剪枝中唯一的一种近似剪枝, 为了保重算法的效率和准确性, 剪枝 2 既需要删除高度点来为精确的剪枝 3 和剪枝 4 创造使用条件, 又需要尽量少的使用, 避免产生更多的误差. 在多数情况下, 剪枝 3 的删边数量要高于剪枝 4, 这是由于剪枝 3 的使用条件在搜索时更容易满足. 剪枝 3 和 4 可以精确的剪边和降低高度点的度, 使得剪枝 2 删点时可以选择更准确的高度点.

表 3 4 种剪枝的效果

数据集	剪枝1删边	剪枝2删点	剪枝3删边	剪枝4删边
contact-primary-school	0	224	106 838	25
contact-high-school	0	278	171 848	142
email-Eu	46	597	219 076	1 556
congress-bills	1	1 626	259 939	176
tags-math-sx	36	957	659 429	1 194
DAWN	296	747	575 795	3 822
NDC-substances	1 063	706	32 178	35 484
tags-stack-overflow	2 045	21 331	8 247 828	60 194
coauth-MAG-Geology	516 790	26 348	134 611	168 942
coauth-DBLP	683 676	89 382	693 472	444 166

图 10 展示了 3 种算法搜索到的最大独立集点数与真实结果的比值. 从图中可以看出在所有搜索过程中的, 3 种算法的准确率都超过了 50%. 同时在所有超图数据集上, MA1 和 MA2 算法的准确率都大幅超过了 BA 算法, 在数据集 congress-bills 上准确率的差异甚至达到了 40% 左右. 而 BA 算法在不同数据集上的表现不太稳定, 差异较大, 在 congress-bills 上的准确率只有不足 55%, 而在数据集 DAWN 上则可超过 90%. MA1 和 MA2 算法则比 BA 算法稳定的多, 在所有的数据集上的准确率都超过了 90%, 甚至在一些数据规模较小的数据集上可以接近 100%, 比如数据集 contact-primary-school.

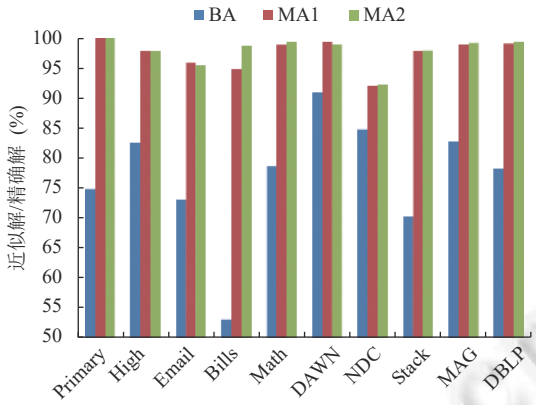

图 10 准确率

总的来说, BA 算法只能搜索到一个不太准确的超图最大独立集, 但其运行时间非常短. MA1 和 MA2 算法的运行时间要长一些, 但可以保证有很高的准确率. 在小规模图上, one-degree 点剪枝比 high-degree 剪枝耗时, 但在大规模图上, one-degree 点剪枝可以在保证精确的情况下, 更快的削减原图的大小, 从而加速 high-degree 剪枝的效率. 因此 MA1 算法适合在较小规模的图上搜索, 其准确率与 MA2 算法接近, 但时间更短. 而包含 one-degree 点剪枝的 MA2 算法更适合在较大规模的图, 此时其准确率和运行效率均优于 MA1 算法.

9 结论及未来工作

本文基于普通图上的独立集定义提出一种面向超图的独立集和最大独立集的定义. 本文又通过观察分析, 提出超图最大独立集搜索的两个原则, 并基于此, 提出了一种基于贪心策略的超图最大独立集基础算法. 进一步, 本文分析了普通图上的最大独立集搜索方法, 提出了一种超图上的近似最大独立集搜索架构, 即精确剪枝与近似剪枝相结合, 以精确剪枝缩小图的规模, 以近似剪枝加快搜索速度. 此外, 本文还基于该架构提出了 3 种精确剪枝策略和一种启发式的近似剪枝策略, 将超图上的近似最大独立集搜索架构和 4 种剪枝策略相结合, 提出了一种改进的超图近似最大独立集算法. 最后通过实验验证, 证明了改进算法虽然速度慢于基础算法, 但可以在线性时间内找到比较接近于真实结果的近似最优解.

本文虽然实现了一种改进的超图最大独立集算法, 但是本文提出的精确剪枝策略仍主要是面向 one-degree 超点和 one-size 超边. 而一些现有的普通图的近似最大独立集算法已经使用 two-degree 剪枝等更高阶的剪枝策略来实现更高的准确率和速度. 下一步的工作将进一步分析超图独立集搜索过程中的高阶剪枝策略, 提高算法效率, 同时将进一步研究算法近似率的理论保证.

References:

[1] Berman P, Fürer M. Approximating maximum independent set in bounded degree graphs. In: Proc. of the 5th Annual ACM-SIAM Symp. on Discrete Algorithms. Arlington: SIMA, 1994. 365–371.

[2] Fu AWC, Wu HH, Cheng J, Wong RCW. IS-Label: An independent-set based labeling scheme for point-to-point distance querying. Proc. of the VLDB Endowment, 2013, 6(6): 457–468. [doi: 10.14778/2536336.2536346]

[3] Halldórsson MM, Radhakrishnan J. Greed is good: Approximating independent sets in sparse and bounded-degree graphs. Algorithmica, 1997, 18(1): 145–163. [doi: 10.1007/BF02523693]

[4] Robson JM. Algorithms for maximum independent sets. Journal of Algorithms, 1986, 7(3): 425–440. [doi: 10.1016/0196-6774(86)90032-5]

[5] Puthal D, Nepal S, Paris C, Ranjan R, Chen JJ. Efficient algorithms for social network coverage and reach. In: Proc. of the 2015 IEEE Int'l Congress on Big Data. New York: IEEE, 2015. 467–474. [doi: 10.1109/BigDataCongress.2015.75]

[6] Basagni S. Finding a maximal weighted independent set in wireless networks. Telecommunication Systems, 2001, 18(1–3): 155–168.

[doi: 10.1023/A:1016747704458]

[7] Lee G, Ko J, Shin K. Hypergraph motifs: Concepts, algorithms, and discoveries. Proc. of the VLDB Endowment, 2020, 13(12): 2256–2269. [doi: 10.14778/3407790.3407823]

[8] Yoon SE, Song H, Shin K, Yi Y. How much and when do we need higher-order information in hypergraphs? A case study on hyperedge prediction. In: Proc. of the 2020 Web Conf. Taipei: ACM, 2020. 2627–2633. [doi: 10.1145/3366423.3380016]

[9] Kook Y, Ko J, Shin K. Evolution of real-world hypergraphs: Patterns and models without oracles. In: Proc. of the 2020 IEEE Int'l Conf. on Data Mining (ICDM). Sorrento: IEEE, 2020. 272–281. [doi: 10.1109/ICDM50108.2020.00036]

[10] Benson AR, Abebe R, Schaub MT, Jadbabaie A, Kleinberg J. Simplicial closure and higher-order link prediction. Proc.of the National Academy of Sciences of the United States of America, 2018, 115(48): E11221–E11230. [doi: 10.1073/pnas.1800683115]

[11] Finocchi I, Finocchi M, Fusco EG. Clique counting in mapreduce: Algorithms and experiments. ACM Journal of Experimental Algorithmics, 2015, 20: 1–20. [doi: 10.1145/2794080]

[12] Hu XC, Tao FY, Chung CW. I/o-efficient algorithms on triangle listing and counting. ACM Trans. on Database Systems, 2014, 39(4): 27. [doi: 10.1145/2691190.2691193]

[13] Estrada E, Rodríguez-Velázquez JA. Subgraph centrality and clustering in complex hyper-networks. Physica A: Statistical Mechanics and its Applications, 2006, 364: 581–594. [doi: 10.1016/j.physa.2005.12.002]

[14] Gregori E, Lenzini L, Mainardi S. Parallel k-clique community detection on large-scale networks. IEEE Trans. on Parallel and Distributed Systems, 2013, 24(8): 1651–1660. [doi: 10.1109/TPDS.2012.229]

[15] Do MT, Yoon SE, Hooi B, Shin K. Structural patterns and generative models of real-world hypergraphs. In: Proc. of the 26th ACM SIGKDD Int'l Conf. on Knowledge Discovery & Data Mining. Virtual Event: ACM, 2020. 176–186. [doi: 10.1145/3394486.3403060]

[16] Benson AR, Kumar R, Tomkins A. Sequences of sets. In: Proc. of the 24th ACM SIGKDD Int'l Conf. on Knowledge Discovery & Data Mining. London: ACM, 2018. 1148–1157. [doi: 10.1145/3219819.3220100]

[17] Battiston F, Cencetti G, Iacopini I, Latora V, Lucas M, Patania A, Young JG, Petri G. Networks beyond pairwise interactions: Structure and dynamics. Physics Reports, 2020, 874: 1–92. [doi: 10.1016/j.physrep.2020.05.004]

[18] Carraghan R, Pardalos PM. An exact algorithm for the maximum clique problem. Operations Research Letters, 1990, 9(6): 375–382. [doi: 10.1016/0167-6377(90)90057-C]

[19] Pardalos PM, Rodgers GP. A branch and bound algorithm for the maximum clique problem. Computers & Operations Research, 1992, 19(5): 363–375. [doi: 10.1016/0305-0548(92)90067-F]

[20] Östergård PRJ. A fast algorithm for the maximum clique problem. Discrete Applied Mathematics, 2002, 120(1–3): 197–207. [doi: 10.1016/S0166-218X(01)00290-6]

[21] Li CM, Quan Z. An efficient branch-and-bound algorithm based on maxsat for the maximum clique problem. Proc. of the AAAI Conf. on Artificial Intelligence, 2010, 24(1): 128–133. [doi: 10.1609/aaai.v24i1.7536]

[22] Xiao MY, Nagamochi H. Exact algorithms for maximum independent set. Information and Computation, 2017, 255: 126–146. [doi: 10.1016/j.ic.2017.06.001]

[23] Akiba T, Iwata Y. Branch-and-reduce exponential/FPT algorithms in practice: A case study of vertex cover. Theoretical Computer Science, 2016, 609: 211–225. [doi: 10.1016/j.tcs.2015.09.023]

[24] Karp RM. Reducibility among combinatorial problems. In: Proc. of the 1972 Symp. on the Complexity of Computer Computations. New York: Plenum Press, 1972. 85–103. [doi: 10.1007/978-1-4684-2001-2_9]

[25] Jagota A, Sanchis LA. Adaptive, restart, randomized greedy heuristics for maximum clique. Journal of Heuristics, 2001, 7(6): 565–585. [doi: 10.1023/A:1011925109392]

[26] Pullan W. Phased local search for the maximum clique problem. Journal of Combinatorial Optimization, 2006, 12(3): 303–323. [doi: 10.1007/s10878-006-9635-y]

[27] Wu QH, Hao JK, Glover F. Multi-neighborhood tabu search for the maximum weight clique problem. Annals of Operations Research, 2012, 196(1): 611–634. [doi: 10.1007/s10479-012-1124-3]

[28] Zhou XD, Wang LA, Chen L. Research on using heuristic algorithms to solve MCP. Computer Engineering and Design, 2007, 28(18): 4329–4332 (in Chinese with English abstract). [doi: 10.3969/j.issn.1000-7024.2007.18.002]

[29] Feng Y. A heuristic algorithm for maximum independent set problem in graph theory. Journal of the Hebei Academy of Sciences, 2021, 38(3): 9–13 (in Chinese with English abstract). [doi: 10.16191/j.cnki.hbkx.2021.03.002]

[30] Liu Y, Lu J, Yang H, Xiao XK, Wei ZW. Towards maximum independent sets on massive graphs. Proc. of the VLDB Endowment, 2015, 8(13): 2122–2133. [doi: 10.14778/2831360.2831366]

[31] Lamm S, Sanders P, Schulz C, Strash D, Werneck RF. Finding near-optimal independent sets at scale. In: Proc. of the 2016 Meeting on

Algorithm Engineering and Experiments (ALENEX). San Francisco: SIAM, 2016. 138–150. [doi: 10.1137/1.9781611974317.12]

[32] Andrade DV, Resende MGC, Werneck RF. Fast local search for the maximum independent set problem. Journal of Heuristics, 2012, 18(4): 525–547. [doi: 10.1007/s10732-012-9196-4]

[33] Dahlum J, Lamm S, Sanders P, Schulz C, Strash D, Werneck RF. Accelerating local search for the maximum independent set problem. In: Proc. of the 15th Int'l Symp. on Experimental Algorithms. St. Petersburg: Springer, 2016. 118–133. [doi: 10.1007/978-3-319-38851-9_9]

[34] Gao XY, Li JZ, Miao DJ. Dynamic approximate maximum independent set on massive graphs. In: Proc. of the 38th IEEE Int'l Conf. on Data Engineering (ICDE). Kuala Lumpur: IEEE, 2022. 1835–1847. [doi: 10.1109/ICDE53745.2022.00183]

[35] Liu XC, Zheng WG, Chen ZY, He ZY, Wang XS. Querying maximum quasi-independent set by pay-and-recycle. In: Proc. of the 38th IEEE Int'l Conf. on Data Engineering (ICDE). Kuala Lumpur: IEEE, 2022. 859–871. [doi: 10.1109/ICDE53745.2022.00069]

[36] Fomin FV, Grandoni F, Kratsch D. A measure & conquer approach for the analysis of exact algorithms. Journal of the ACM, 2009, 56(5): 25. [doi: 10.1145/1552285.1552286]

[37] Chang LJ, Li W, Zhang WJ. Computing a near-maximum independent set in linear time by reducing-peeling. In: Proc. of the 2017 ACM Int'l Conf. on Management of Data. Chicago: ACM, 2017. 1181–1196. [doi: 10.1145/3035918.3035939]

[38] Leng M, Sun L, Bian J, Ma Y. An O(m) algorithm for cores decomposition of undirected hypergrap. Journal of Chinese Computer Systems, 2013, 34(11): 2568–2573 (in Chinese with English abstract). [doi: 10.3969/j.issn.1000-1220.2013.11.031]

附中文参考文献:

[28] 周旭东, 王丽爱, 陈崚. 启发式算法求解最大团问题研究. 计算机工程与设计, 2007, 28(18): 4329–4332. [doi: 10.3969/j.issn.1000-7024.2007.18.002]

[29] 冯云. 一种求解图论中最大独立集问题的启发式算法. 河北省科学院学报, 2021, 38(3): 9–13. [doi: 10.16191/j.cnki.hbkx.2021.03.002]

[38] 冷明, 孙凌宇, 边计年, 马昱春. 一种时间复杂度为O(m)的无向超图核值求解算法. 小型微型计算机系统, 2013, 34(11): 2568–2573. [doi: 10.3969/j.issn.1000-1220.2013.11.031]

徐兰天(1997－), 男, 硕士, 主要研究领域为图数据挖掘.

戴永恒(1983－), 男, 博士, 工程师, 主要研究领域为领域建模, 专家知识结构化, 融合的机器学习.

李荣华(1985－), 男, 博士, 教授, CCF 专业会员, 主要研究领域为大规模图数据管理与挖掘, 社交网络分析与挖掘.

王国仁(1966－), 男, 博士, 教授, CCF 杰出会员, 主要研究领域为不确定数据管理, 非结构化数据管理.