中国科学: 信息科学 2022 年 第 52 卷 第 5 期: 784–803
SCIENTIA SINICA Informationis
《中国科学》杂志社
SCIENCE CHINA PRESS
评述

# 图数据中极大团枚举问题的求解: 研究现状与挑战

许绍显[1,2,3,4], 廖小飞[1,2,3,4], 邵志远[1,2,3,4*], 华强胜[1,2,3,4], 金海[1,2,3,4]

1. 华中科技大学计算机科学与技术学院, 武汉 430074
2. 大数据技术与系统国家地方联合工程研究中心, 武汉 430074
3. 服务计算技术与系统教育部重点实验室, 武汉 430074
4. 集群与网格计算湖北省重点实验室, 武汉 430074
* 通信作者. E-mail: zyshao@hust.edu.cn

收稿日期: 2021–05–10; 修回日期: 2021–07–06; 接受日期: 2021–09–01; 网络出版日期: 2022–04–14

国家自然科学基金 (批准号: 61972444, 61832006, 61825202, 61702202) 资助项目

**摘要** 随着大数据时代的到来, 图数据挖掘成为了一个热门的研究方向. 极大团枚举 (maximal clique enumeration, MCE) 作为图论中的一个基本问题, 在很多领域都有着广泛的应用. 然而, 鉴于极大团枚举问题本身的复杂性以及现实图数据规模的飞速增长, 在现实图数据上进行极大团枚举是很耗时的. 目前已经有大量的工作对该问题的求解算法进行改进, 或采用各种计算优化方法减少算法的运行时间. 本文就极大团枚举问题做了如下工作: 对现有的极大团枚举问题的研究工作进行了分类归纳; 对极大团枚举问题的研究现状进行了详细介绍; 对该问题进一步发展所面临的挑战和发展方向进行了讨论和展望.

**关键词** 极大团枚举, 图论, 图数据挖掘, 图划分, 并行计算

## 1 引言

在大数据时代, 随着计算机技术和互联网技术的飞速发展, 之前常用的数据结构和数据处理方式已经不能适应各行各业中产生的海量数据信息. 图作为一种更加灵活和复杂数据结构, 在描述个体属性、个体间关系及群体特征等方面有着极大的优势, 越来越受到人们的重视. 极大团枚举 (maximal clique enumeration, MCE) 问题是图论中的一个基本问题, 与图论中的许多关键问题都有着紧密的联系, 诸如着色问题 [1]、独立集问题 [2] 等, 而且在生物 [3~8]、化学 [9]、社交网络 [10~12]、无线通讯 [13] 等领域都有着极高的应用价值. 文献 [7] 在生物图上使用极大团枚举来发掘相关的基因. 文献 [8] 通过极大团枚举以及对极大团重叠部分和非重叠部分进行分析来预测蛋白质间的互相作用. 文献 [9] 用极大团枚举来发掘化学结构间的公共部分从而进行分析. 文献 [12] 利用极大团枚举在不同类型的图上进行

**引用格式:** 许绍显, 廖小飞, 邵志远, 等. 图数据中极大团枚举问题的求解: 研究现状与挑战. 中国科学: 信息科学, 2022, 52: 784–803, doi: 10.1360/SSI-2021-0155
Xu S X, Liao X F, Shao Z Y, et al. Maximal clique enumeration problem on graphs: status and challenges (in Chinese). Sci Sin Inform, 2022, 52: 784–803, doi: 10.1360/SSI-2021-0155

 

社区发现. 文献 [13] 提出了一种基于极大团的节点聚类协议, 其在负载均衡、鲁棒性等方面有着更好的效果. 文献 [14] 在空间地理图上使用极大团枚举算法发掘图中的并置模式 (co-location pattern). 文献 [15] 使用极大团的大小和数目等信息评估其算法在生成股票关联图 (stock-correlation networks) 上的效果.

长期以来, 人们对极大团枚举算法的研究取得了大量成果. 但数据规模的增长和各种并行技术的发展, 为极大团枚举问题带来了许多新挑战和新发展. 就极大团枚举问题, 本文做了如下工作:

- 对现有的极大团枚举问题的研究工作进行了分类归纳.
- 详细介绍了各种不同的枚举算法, 以及各种计算方面的优化手段.
- 对该问题进一步发展所面临的挑战和发展方向进行了讨论和展望.

本文的其余部分组织如下: 第 2 节介绍极大团枚举问题的定义; 第 3 节说明我们对各项工作的分类依据以及分类结果; 第 4 节详细介绍极大团枚举问题在算法上的发展历史; 第 5 节详细介绍极大团枚举算法在计算方面的各种优化手段; 最后, 在第 6 节对现有算法的不足以及该问题进一步发展所面临的挑战进行了讨论, 对该问题之后的发展方向进行了展望.

## 2 极大团枚举 (MCE) 问题定义

本节将介绍有关无向图、完全图、团的相关知识, 并给出极大团枚举问题以及一些衍生问题的定义.

**无向图 (undirected graph).** 一般用符号 $G=(V,E)$ 表示图. 其中顶点集 $V=\{v_1,v_2,\ldots,v_n\}$, 表示图中共有 $n$ 个顶点; 边集 $E=\{(v_i,v_j)\mid v_i,v_j\in V\}$ 是由顶点的二元组组成的集合, 对于图中任意两个顶点 $v_i,v_j\in V$, 若两点间有边连接, 则 $(v_i,v_j)\in E$, 否则 $(v_i,v_j)\notin E$. 无向图中, 边是没有方向之分的, 即若有 $(v_i,v_j)\in E$, 则必有 $(v_j,v_i)\in E$, 且 $(v_i,v_j)$ 与 $(v_j,v_i)$ 指同一条边.

**子图 (subgraph).** 对于图 $G=(V,E)$ 和 $G'=(V',E')$, 若 $V'\subseteq V$ 且 $E'\subseteq E$, 则称 $G'$ 是 $G$ 的子图.

**诱导子图 (induced graph).** 对于图 $G=(V,E)$ 和 $G'=(V',E')$, 若 $V'\subseteq V$, $E'=\{(v_i,v_j)\mid v_i,v_j\in V'$ 且 $(v_i,v_j)\in E\}$, 则称 $G'$ 是以 $V'$ 为顶点集的诱导子图, 即对于 $V'$ 中的任意两个顶点, 只要它们在 $G$ 中有边, 则在 $G'$ 中一定有边.

**无向完全图 (complete undirected graph).** 在无向图 $G=(V,E)$ 中, 若对于任意 $v_i,v_j\in V$, 都有 $(v_i,v_j)\in E$, 即任意两顶点间都有边连接, 则称 $G$ 为无向完全图. 若存在 $G'$ 是 $G$ 的子图, 且 $G'$ 是完全图, 则称 $G'$ 是 $G$ 的完全子图.

**团 (clique).** 在无向图 $G=(V,E)$ 中, 对于顶点集 $C\subseteq V$, 且对于任意 $v_i,v_j\in C$, 都有 $(v_i,v_j)\in E$, 则称 $C$ 为图 $G$ 中的一个团. 以 $C$ 为顶点集的诱导子图 $G'$ 为图 $G$ 的完全子图.

**极大团 (maximal clique, MC).** 在无向图 $G=(V,E)$ 中, 有团 $M$, 若不存在 $v\in(V\setminus M)$, 使得对任意 $u\in M$, 都有 $(v,u)\in E$, 则称团 $M$ 为极大团. 也就是说不存在一个团外的顶点与该团中的所有顶点都有一条边, 也即该团不能被更大的团所包含.

根据团和极大团的定义可以看出, 图 1(a) 中, $\{1,2,3,4\}$ 和 $\{4,5\}$ 为极大团; 图 1(b) 中, $\{1,2,3,4,5\}$, $\{1,6\}$, $\{2,7\}$, $\{5,8\}$, $\{6,7,8\}$ 为极大团. 图 1(a) 将在介绍 BK (Bron-Kerbosch) 算法时 (在 4.1.1 小节) 再次使用; 图 1(b) 将在介绍输出敏感算法时 (4.1.2 小节) 再次使用.

在有了极大团的概念后, 我们可以给出极大团枚举 (MCE) 问题的定义: 给定一个无向图, 枚举出图中全部的极大团.

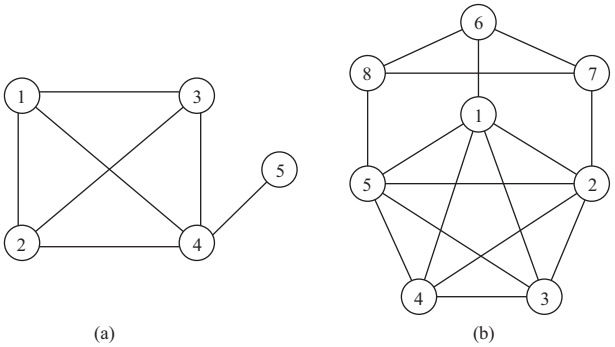


**图 1　两个简单的示例图**

**Figure 1**　Two simple example graphs. (a) a 2-MC graph; (b) a 5-MC graph

在 1965 年, MOON 和 MOSER [16] 严格证明了一个图可能包含的最多的极大团数目 $f(n)$ 和图的顶点数目 $n$ 有着直接的数学关系: 对于一个有 $n$ 个顶点的图, 最多有 $3^{(n/3)}$ 个极大团. 极大团问题的计数版本 (MC counting) 是一个 #P-complete 的问题 [17], 而枚举要比计数更难 [18].

除了极大团枚举问题外, 在图数据上还有一些相关的问题, 如极大独立集枚举问题、最大团问题、$k$ 团问题, 这里也给出他们的定义和部分相关文献.

一个图中的极大独立集与其补图中的极大团一一对应, 因此极大独立集枚举问题是极大团枚举问题的对偶问题.

**极大独立集 (maximal independent set).** 在无向图 $G = (V, E)$ 中, 若有顶点集 $U$, 对于任意 $v, u \in U$, 都有 $(v, u) \notin E$, 则称 $U$ 为独立集. 若对任意 $p \in (V \setminus U)$, 都存在 $v_i \in U$, 且 $(p, v_i) \in E$, 则称独立集 $U$ 为极大独立集. 极大独立集枚举问题 [19,20] 的定义为: 给定一个无向图, 枚举出图中全部的极大独立集.

**最大团 (maximum clique).** 最大团是图的所有团中包含顶点数最多的团, 最大团本身也一定是一个极大团. 对于图 1(b) 的例子, $\{1, 2, 3, 4, 5\}$ 为其最大团. 最大团问题 [21] 的定义为: 给定一个无向图, 找出图中的最大团.

**$k$ 团 ($k$-clique).** 包含 $k$ 个顶点的团. $k$ 团问题 [22,23] 的定义为: 给定一个无向图, 找到所有包含 $k$ 个顶点的团.

## 3　相关工作分类

基于本文所综述的文章, 我们将从算法 (algorithm) 和计算 (computation) 两大方面对当前的研究现状进行详细介绍. 在算法方面, 我们将介绍对枚举策略本身进行研究和改进的相关工作. 在计算方面, 我们将介绍通过各种计算优化手段 (如图划分、并行等), 提升给定 MCE 算法运行效率的相关工作. 本节中, 我们将首先说明我们对各项工作的分类依据以及分类结果, 然后对本文提到的、可计算时空复杂度的算法的复杂度进行一个汇总.

### 3.1　算法分类

算法方面涉及的工作及其分类如图 2 [24~51] 所示. 首先我们根据算法是否只针对特殊结构的图进行划分. 在特殊图中, 如平面图 (planar graph)、弦图 (chordal graph)、有维度限制的方形图 (graphs of bounded boxicity) 等, 它们的极大团数目上限是线性级的或多项式级的 [24~26], 而且借助其特殊结构,

可以更快地枚举出其所有极大团 [27~29]. 对于一般图的算法, 我们根据其针对的是静态图、动态图还是不确定图, 将相关工作进行划分. 静态图上的算法更加关注如何快速的将图中所有的极大团枚举出来; 动态图上的算法 [30~35] 则更关注在图发生变化时, 如何高效地对之前的枚举结果进行维护和更新; 而不确定图中, 每条边都有一个存在概率, 除了要枚举极大团外, 还要计算极大团存在的概率 [36,37].

在动态图上维护更新极大团枚举结果的工作由 Stix 在 2004 年首次提出 [30], 对于图中的一条新增边 $u \to v$, Stix 对包含 $u$ 的极大团和包含 $v$ 的极大团进行笛卡尔积 (Cartesian product), 生成新图中所有可能存在的极大团并一一验证. 2017 年, Sun 等 [33] 将其策略改进为对包含 $u$ 或 $v$ 的原有极大团进行扩展. Cheng 等 [31] (5.1.1 小节) 提到了如何在他们提出的特殊结构 $H^*$-graph 上更新极大团. Xu 等 [52] (4.1.1 小节) 也涉及了如何利用他们的枚举方法和数据结构来进行极大团更新. Das 等 [34,35] 则首先精确地抽取出受到更新影响的子图, 并在子图上应用枚举算法找到新的极大团, 同时, 他们理论分析了极大团集合由新增边导致的变化程度的界限以及他们算法的时间复杂度. 这里我们简要介绍了动态图上的极大团相关工作, 但其涉及到的方法和技术与极大团枚举本身有着较大的不同, 后文中我们将会把关注点放在一般静态图上的极大团枚举, 这也是极大团相关问题的基础和主要研究方向.

对一般静态图上的枚举算法, 我们可以根据其搜索策略将之分为广度优先搜索算法与深度优先搜索算法, 具体细节会在第 4 节进行详细介绍. 广度优先搜索算法目前有 Kose [38] 提出的一种从小团开始不断合并的算法, 以及 Yu 等 [39] 提出的一种类似遍历边的算法. Kose 和 Yu 提出的算法每次都会扩展所有能被扩展的团, 因此我们将之归入广度优先搜索算法.

深度优先搜索算法则可以根据其搜索树的构成进一步分为以图的顶点作为搜索树节点的算法, 以及以极大团作为搜索树节点的算法. 前者通过递归回溯的方式, 每次递归将一个点加入当前的团结构, 并且还需要其他两个集合分别保存能被加入当前团的点和已经加入过当前团的点, 分别代表着接下来需要探索的情况和已经被探索过的情况. 该类算法由 Bron 和 Kerbosch [40] 在 1973 年首次提出, 因此也被称为 BK 算法.

以极大团作为搜索树节点的算法则通过定义一种极大团间的父子关系构建极大团枚举树, 其可以保证两个连续输出之间的时间间隔 (也被成为延迟) 为多项式级别, 而总的枚举时间与最终输出的极大团个数有关, 因此被称为输出敏感算法. BK 算法和输出敏感算法都会对每一个可能的分支路径深入到不能再深入为止, 因此我们将其归入深度优先搜索算法.

### 3.2 计算优化分类

鉴于极大团枚举问题本身的难度, 其在算法上的每一次进步都非常不易. 相较而言, 利用图划分、并行、位运算等各种手段, 使算法的计算过程更加贴合现有的计算资源, 减少算法整体运行时间的工作会稍多一些, 我们将这些工作称为计算优化 (computation optimization). 根据各工作的主要特征, 将其分为 3 大类, 具体的分类情况如图 3 [31,53~68] 所示 (图中论文间的连线表示后者与前者的工作有相似或继承之处, 每个分类内, 各工作按照时间顺序排列, 有关的具体内容会在第 5 节中进行介绍).

下面我们依次说明各类别的含义: (1) 分而治之 (partition based optimization): 分而治之作为一种常见的缩减问题规模的手段, 其在极大团枚举问题上也被大量使用; 该类中的工作通过图划分技术, 将整个图上的极大团枚举问题, 划分为各个子图上的极大团枚举问题; 并且根据划分次数, 我们又将其细分为单级划分 (single-level decomposition)和多级划分 (multi-level decomposition), 单级划分指只会对输入图进行一次划分并得到一系列子图, 多级划分则会对第 1 次划分后得到各个子图再进行一次或多次划. (2) 运行时优化 (runtime optimization): 除了图划分外, 还有一些用工作偷取、数据并行原语等手段进行并行运算的工作, 我们将其归为运行时优化. (3) 频繁操作优化 (frequent operation

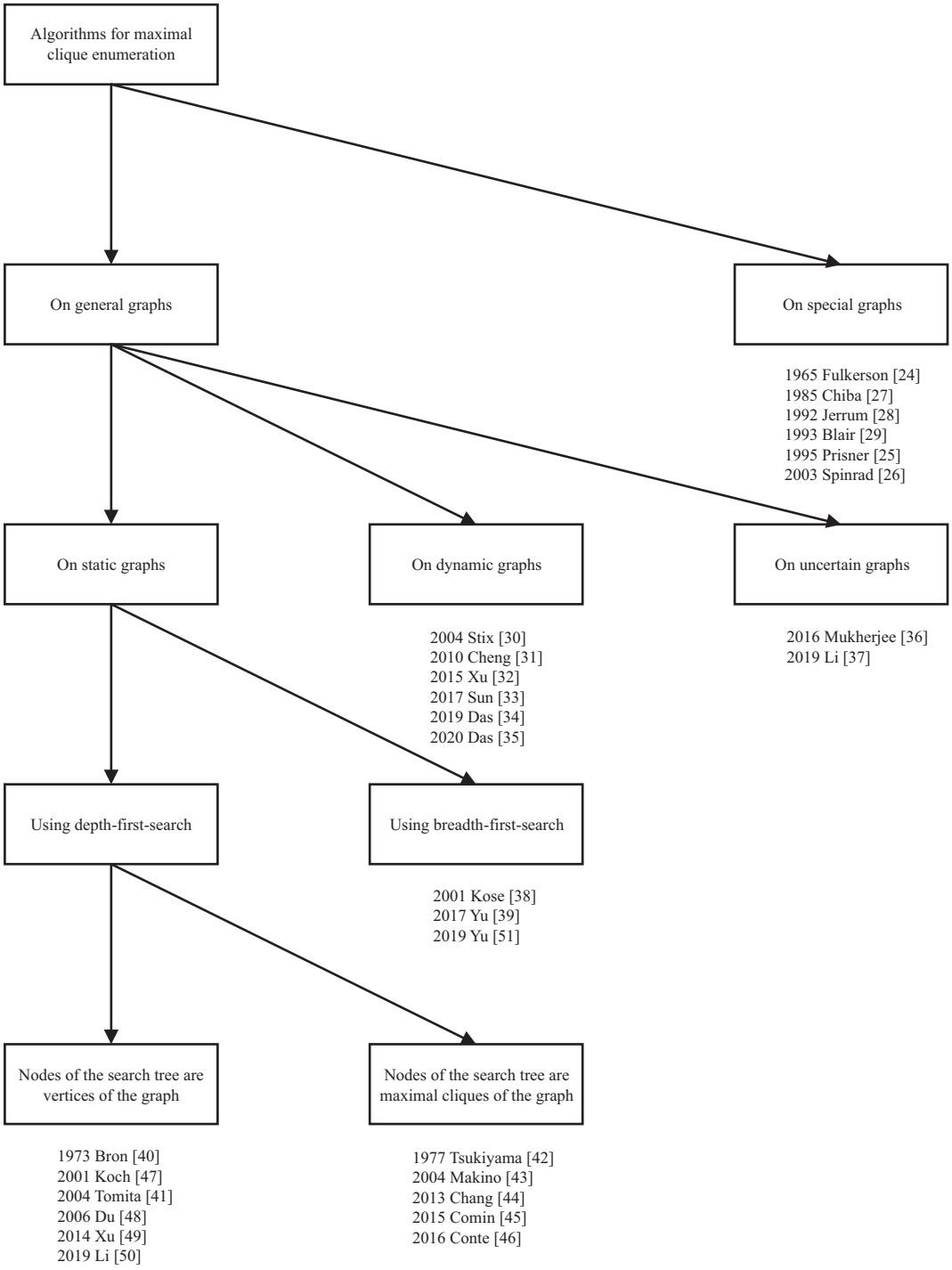


**图 2　MCE 算法的分类**

**Figure 2**　The classification of MCE algorithms

optimization): MCE 问题中最频繁的操作就是集合运算, 有一些工作利用用位邻接矩阵或位向量表示工作集, 减小了工作集的大小、提升了集合运算速度, 这些工作被归为频繁操作优化.

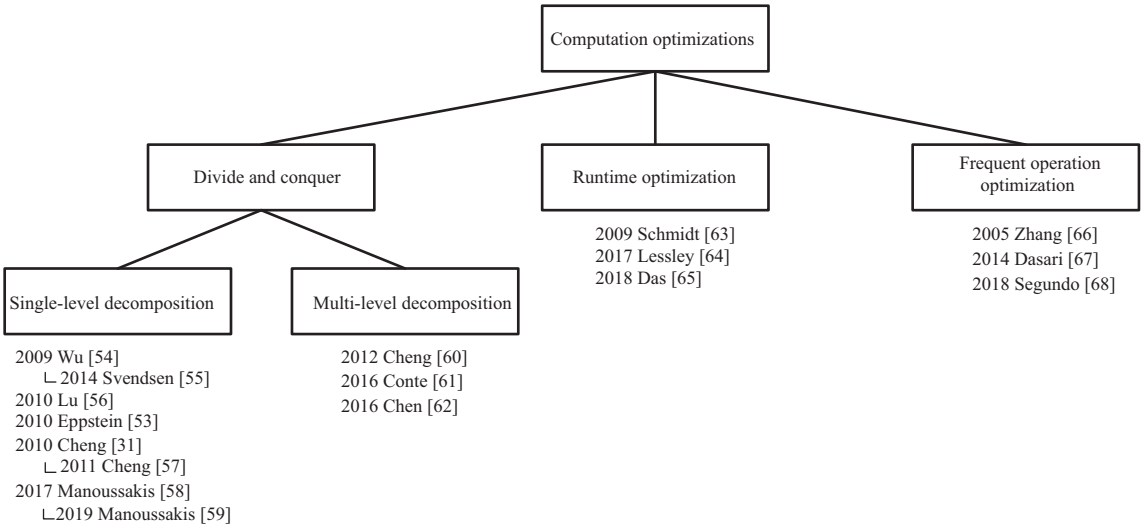


**图 3 MCE 计算优化工作的分类**

**Figure 3** The classification of computation optimizations on MCE

除了图 3 中的作为归类依据的主要特征外, 我们在表 1 [31,53~68] 中展现了各计算优化工作涉及的其他特征, 包括执行方式、图划分方式和顶点排序方式. 划分方式中, 按邻域划分指的是, 以一个顶点及其邻域为顶点集的诱导子图作为一个子问题; 其他划分方式也会在 5.1 小节中详细介绍. 顶点排序中, 退化度排序详见 5.1.1 小节中对 Eppstein [53] 工作的介绍, 其他排序方式有按顶点度数排序、按顶点三角形计数排序等.

### 3.3 算法时空复杂度

我们在表 2 [40~46,53,58,59] 中将本文提到的可计算时空复杂度的算法的复杂度进行一个总合, 由于很多工作只是进行了计算优化, 无法或没有给出时空复杂度的证明, 表中只列出了部分算法. 其中预处理时间 (preprocessing time) 指读入数据、构建所需数据结构的时间, 延迟 (delay) 指输出敏感算法中两个连续输出间的时间间隔, 枚举时间 (enumeration time) 指从开始枚举到枚举出所有极大团所需的时间, 存储空间 (space) 指除了输入图的存储外, 算法所需的额外存储空间.

## 4 算法的发展历程

有关极大团枚举的最早工作可以追溯到 20 世纪 50 年代, Harary 等 [69] 提出通过团结构分析目标的社会关系. 接下来许多前辈也进行了极大团相关的研究 [70~73]. 虽然他们研究的问题不同, 但本质都涉及到极大团枚举问题. 在此之后, 随着极大团在不同领域得到应用, 极大团枚举问题也成为了一个被广泛研究的问题. 本节中, 我们将依照 3.1 小节中的分类, 对极大团枚举算法的发展历程进行详细介绍.

### 4.1 深度优先搜索算法

基于深度优先搜索的 BK 系列算法和输出敏感算法是最早被提出的两种极大团枚举算法, 下面我们将分别对两者的枚举方法、算法特点, 以及相关的算法改进工作进行详细说明.

**表 1　各计算优化工作的执行方式、图划分方式, 以及顶点排序方式 (SMP 代表共享内存并行)**

**Table 1** The execution model, graph partition method and vertex ordering method of each computation optimization work (SMP stands for shared memory parallel)

| | Execution model | | | Graph partition method | | Vertex ordering method | |
|---|---|---|---|---|---|---|---|
| | Serial | SMP | Distributed | Neighborhood | Other means | Degeneracy | Other means |
| Wu [54] | | | √ | | √ | | |
| Svendsen [55] | | | √ | √ | | | √ |
| Lu [56] | | | √ | | √ | | |
| Eppstein [53] | √ | | | √ | | √ | |
| Cheng [31, 57] | √ | | | | √ | | |
| Manoussakis [58, 59] | √ | | | | √ | √ | |
| Cheng [60] | | | √ | | √ | | |
| Conte [61] | | | √ | | √ | | |
| Chen [62] | | | √ | √ | | | |
| Schmidt [63] | | √ | | | | | |
| Lessley [64] | | √ | | | | | |
| Das [65] | | √ | | √ | | √ | √ |
| Zhang [66] | | √ | | | | | |
| Dasari [67] | | √ | | √ | | √ | |
| Segundo [68] | √ | | | √ | | | √ |

### 4.1.1 BK 系列算法

**BK 算法.** 1973 年, Bron 和 Kerbosch [40] 提出了第 1 个递归回溯算法来枚举无向图中的极大团. 该算法使用了 3 个集合来辅助枚举: (1) $R$ 集 (结果集), 该集合保存了当前找到的团, 随着递归深入移入顶点, 或随着回溯移出顶点; (2) $P$ 集 (候选集), 该集合保存了所有可以加入当前团的顶点, 即 $P$ 中的顶点与 $R$ 中的每个顶点都相邻; (3) $X$ 集 (禁止集), 该集合保存了所有已经加入过 $R$ 的顶点, 即当前状态下所有包含 $X$ 中顶点的团都生成过了. 当 $P$ 和 $X$ 都为空时, $R$ 中的顶点就是构成一个未被发掘过的极大团, $P$ 为空表明不再有顶点能被加入 $R$, $X$ 为空保证了 $R$ 是极大的、不重复的.

BK 算法的伪代码如算法 1 所示, 其中 $N(v)$ 表示输入图 $G$ 中点 $v$ 的邻居 (不含 $v$). 算法每层递归中, 首先判断当前团 $R$ 是否为极大团, 如果是则进行输出 (line 1, 2); 否则依次对候选集 $P$ 中的点 $v$ 进行扩展 (line 4); 将 $v$ 加入 $R$, 并更新 $P$ 和 $X$, 保证其中的点与 $R$ 中每个点都相连, 扩展后的结果作为输入传入下层递归 (line 5); 递归返回后, 将点 $v$ 移入 $X$(line 6, 7), 代表该分支已探索完.

**Algorithm 1** BK$(R, P, X)$

```
1: if P = φ and X = φ then
2:     R is a maximal clique;
3: end if
4: for v ∈ P do
5:     BK(R ∪ v, P ∩ N(v), X ∩ N(v));
6:     P = P \ v;
7:     X = X ∪ v;
8: end for
```

**表 2 算法时空复杂度** [a)]

**Table 2** Time and space complexity of the algorithms[a)]

| | Preprocessing time | Delay | Enumeration time | Space |
|---|---|---|---|---|
| Bron-Kerbosch [40] | $O(m)$ | – | Unbounded | $O(n+q\Delta)$ |
| Tomita [41] | $O(m)$ | – | $O(3^n/3)$ | $O(n+q\Delta)$ |
| Eppstein [53] | $O(m)$ | – | $O(d(n-d)^{d/3})$ | $O(n+d\Delta)$ |
| Tsukiyama [42] | $O(n^2)$ | $O(nm)$ | $\alpha O(nm)$ | $O(n^2)$ |
| Makino [43] | $O(n^2)$ | $O(n^{2.37})$ | $\alpha O(n^{2.37})$ | $O(n^2)$ |
| Chang [44] | $O(m)$ | $O(\Delta h^3)$ | $\alpha O(\Delta h^3)$ | $O(m)$ |
| Comin [45] | $O(n^{5.37})$ | $O(n^{2.09})$ | $\alpha O(n^{2.09})$ | $O(n^{4.27})$ |
| Conte [46] | $O(mL)$ | $O(\min(mk,qk)L)$ | $\alpha O(\min(mk,qk)L)$ | $O(d)$ |
| Manoussakis [58] | $O(m+\text{poly}(d)n)$ | $O(\text{poly}(d))$ | $\alpha O(\text{poly}(d))$ | $O(\alpha q)$ |
| Manoussakis [59] | $O(m+\text{poly}(d)n)$ | $O(\text{poly}(d)qd\Delta)$ | $\alpha O(\text{poly}(d)qd\Delta)$ | $O(\text{poly}(k))$ |

a) $n$ = number of vertices, $m$ = number of edges, $\Delta$ = max degree, $\alpha$ = number of maximal cliques, $d$ = degeneracy of the graph, $q$ = largest clique size, $L=\log_{O(1)}(m+n)$, poly$(\cdot)$ = a polynomial of $\cdot$, $h$ = the smallest integer such that $|\{v|v\in V,|N(v)|\geqslant h\}|\leqslant h$.

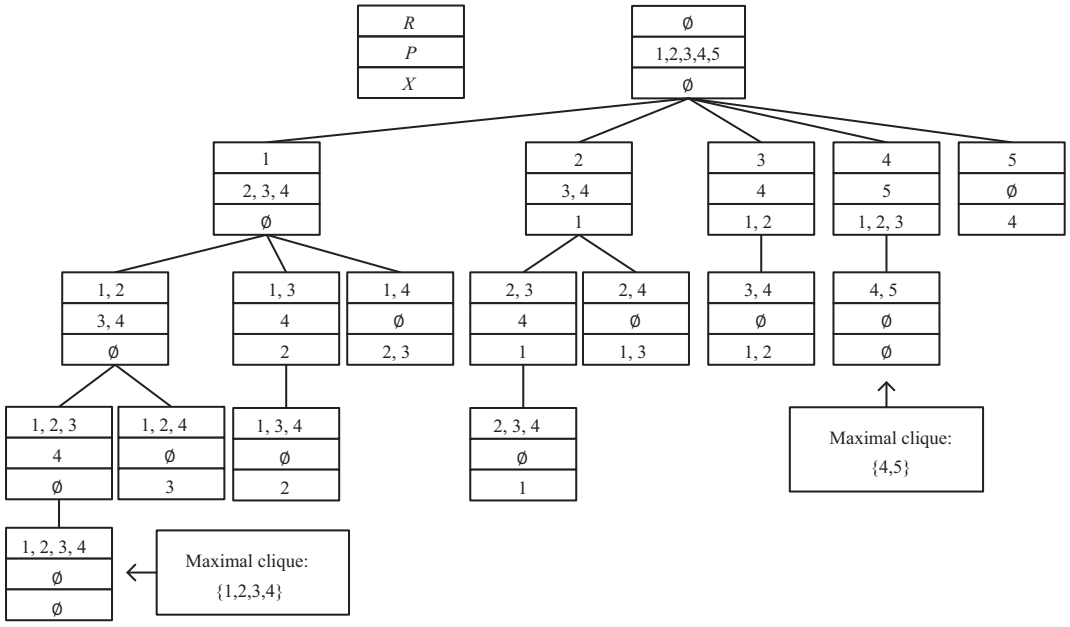


**图 4 BK 算法在图 1(a) 上的搜索树**

**Figure 4** The search tree of BK algorithm on Figure 1(a)

BK 算法在图 1(a) 上的搜索树如图 4 所示. 从中可以看出, 虽然 BK 算法的形式非常简洁, 但缺少剪枝手段, 会产生很多重复的搜索分支, 即使对于极为简单的图 1(a), 其搜索树仍然比较复杂.

**$\mathbf{BK_{pivot}}$算法.** 2001 年, Koch [47] 提出了一种选取中心点 (pivot) 的方法, 可以大大减少 BK 算法搜索树的分支. 其指出: 每层递归中, 可以在 $P$ 集中选取一个中心点 $u_t$, 在之后的扩展中 (算法 1, line 4) 就只需要处理 $u_t$ 以及与 $u_t$ 不相邻的点即可. 这一点很容易证明: 对于任意极大团 $C$, $C$ 要么包含

$u_t$ (和 $u_t$ 的邻居), 要么不包含; 对于包含 $u_t$ (和 $u_t$ 邻居) 的极大团, 可以在将 $u_t$ 加入 $R$ 后的分支中得到; 而对于其他的极大团, 可以在对 $u_t$ 的非邻居的扩展中得到.

2004 年, Tomita 等 [41] 给出了一种选取中心点的方式, 并证明了其算法 $\mathrm{BK}_{\mathrm{pivot}}$ 在最坏情况下的时间复杂度为 $O(3^{(n/3)})$. 算法中, 其选择 $P$ 和 $X$ 集的所有点中, 在 $P$ 集中的邻居数量最多的点作为中心点, 这样的选择可以让候选集中与中心点不相邻的点尽量少, 即选出中心点后的搜索分支尽量少. 例如图 4 中根节点处会选择顶点 4 作为中心点, 而图 1(a) 中所有点都与点 4 相邻, 因此接下来的 for 循环中就只需处理点 4 一个点, 使得原来的 5 个搜索分支减少为了 1 个.

2006 年, Du 等 [48] 利用三角形是所有大小大于 3 的极大团的基础结构, 用三角形邻域 $\sigma(v)$ (所有可以与 $v$ 和 $v$ 的邻居组成三角形的点) 代替了之前的邻域 $N(v)$, 对 $P$ 集做出了额外限制, 减少了其中的顶点数量, 对搜索树进一步剪枝. 作者还实现了算法的并行版本, 其将一个顶点及其邻域 $N(v)$ 作为一个基础任务, 每个处理器按照一个简单的序号映射关系负责多个基础任务, 这也是首个按邻域并行的算法. 但其并没有针对并行的特点进行额外优化, 负载不均衡等问题使得其在 30 个处理器上相比 1 个处理器只能达到平均 7 倍的加速.

2014 年, Xu 等 [49] 在 Eppstein [53] 工作 (在 5.1.1 小节进行详细介绍) 的基础上, 减小了算法中参与交集操作的顶点数目. 原有的 BK 算法在对点 $v$ 进行扩展时, 会用当前的 $P$ 集与 $v$ 的邻居做交集操作以形成新的 $P$ 集 (算法 1, line 4), 但是当我们有了特定的顶点顺序且是在某个顶点的邻域上进行枚举时, 该步操作的集合大小可以通过如下方法进行缩减: 假设在顶点 $v$ 的邻域上进行枚举, 在将点 $u$ 加入团进行扩展时, 只需要让 $P$ 集与点 $u$ 的邻居中序号大于 $v$ 的部分进行交集即可, 即 $\mathrm{candidate} \cap \{w | w \in N(u), \mathrm{ID}_w > \mathrm{ID}_v\}$. 除此之外, Xu 等还提出了一种基于 $k$-core [74] 的顶点排序方式. 其实验显示, 与退化度排序和顶点度排序相比, 该排序方式在多个数据集上都有着更好的效果 (总枚举时间更短). 最后, 作者以按邻域划分的方式构建了其算法的分布式版本.

2018 年, Segundo 等 [68] 发现, 用位向量表示集合, 虽然可以加速交集运算, 但会使 pivot 选取过程花费更长的时间. 为了更好地发挥位向量的作用, 其对顶点排序方式和 pivot 选取方式进行了修改. 其改用最大度优先排序: 在给顶点排序时, 每次删去度最大的顶点并同时删除其边, 顶点的删除顺序即为新的顶点排序. 在进行了排序后, 作者直接选取禁止集中的第 1 个顶点作为 pivot (在禁止集为空时, 选取候选集的第 1 个顶点). 因为按照排序策略, 越靠前的顶点度越大, 该算法期望一个在整个图中邻居更多的顶点, 也同样在候选集中有更多的邻居. 实验也表明, 虽然这样的策略可能选取的 pivot 不是最优的, 使后续要探索的分支增多, 但极大的优化了 pivot 的选取速度, 能够带来总体上的性能提升. 在 DIMACS 数据集中的 40 个图上, 作者提出的算法在 36 个例子上, 比 $\mathrm{BK}_{\mathrm{pivot}}$ 快了 1.2~20 倍.

**$\mathrm{BK}_{\mathrm{rcd}}$算法.** 2019 年, Li 等 [50] 观察到这样一个现象: 在采用了 Eppstein 提出的退化度排序并对输入图进行按邻划分后 (相关内容在 5.1.1 小节进行详细介绍), 得到的子图中存在着大量的稠密子图. 而之前的 BK 算法都采用自底向上的策略进行枚举, 即从零开始, 每次扩展一个点加入团. 这样的策略在稀疏图上更为高效, 而在这些稠密子图上, 会增加不必要的递归深度: 对于稠密子图, 可能只需要剔除少量的顶点就可以找到一个极大团, 因此一种自顶向下的枚举策略会更适用于这样的稠密子图. 基于此, Li 提出了 $\mathrm{BK}_{\mathrm{rcd}}$ 算法, 其伪代码如算法 2 所示.

与 $\mathrm{BK}_{\mathrm{pivot}}$ 相反, $\mathrm{BK}_{\mathrm{rcd}}$ 每次尝试从候选集中剔除一个邻居最少的点 (line 5), 从而期望剩余的部分成为一个团 (line 4), 并对被剔除的部分 (被剔除的点及其邻域) 递归的使用 $\mathrm{BK}_{\mathrm{rcd}}$ 以保证枚举出所有的极大团 (line 6). 与其他 BK 算法相比, $\mathrm{BK}_{\mathrm{rcd}}$ 还在循环结束后多了一步验证操作, 即验证 $P$ 是一个非空的、不重复的团 (line 10), 这样 $R$ 和 $P$ 就一起组成了一个极大团. 对于稠密图而言, 其初始结构更接近于团结构, 采用这种 “自顶向下” 的枚举策略可以更快地以更少的步数找到极大团. 为了使

**Algorithm 2** $\mathrm{BK}_{\mathrm{rcd}}(R,P,X)$

```
if P = φ and X = φ then
    R is a maximal clique;
end if
while P is not a clique do
    Choose v ∈ P, which minimize N(v) ∩ P;
    BK_rcd(R ∪ v, P ∩ N(v), X ∩ N(v));
    P = P \ v;
    X = X ∪ v;
end while
if P ≠ φ and N(P) ∩ X = φ then
    R ∪ P is a maximal clique;
end if
```

$\mathrm{BK}_{\mathrm{rcd}}$ 和 $\mathrm{BK}_{\mathrm{pivot}}$ 分别发挥各自的优势, 作者还分析计算了 $\mathrm{BK}_{\mathrm{rcd}}$ 的适用条件, 并构建了 $\mathrm{MCE}_{\mathrm{hybrid}}$ 算法: 对不同的子图根据判断条件应用 $\mathrm{BK}_{\mathrm{rcd}}$ 或者 $\mathrm{BK}_{\mathrm{pivot}}$.

#### 4.1.2 输出敏感算法

除了 BK 系列算法, 输出敏感算法也是一种深度优先搜寻算法, 但其通过定义一种极大团间的父子关系来构建极大团枚举树, 保证两个连续输出之间的时间间隔 (也被称为延迟) 为多项式级别. 这种算法的总枚举时间与最终输出的极大团个数有关, 因此被称为输出敏感算法. 首个输出敏感算法是 Tsukiyama 等 [42] 在 1977 年提出的, 其延迟为 $\mathrm{O}(nm)$, $n$ 为顶点数, $m$ 为边数.

在给出极大团的父子关系前, 我们首先需要介绍一个基础操作, 补全 (complete), 该操作被用于计算包含团 $S$ 的、字典顺序最大 (这里将字典序靠前的称为字典序大的, 例如 "123">"14") 的极大团, 简写为 $C(S)$. 该操作可以通过从 $S$ 中任选一个点 $v$, 按从小到大的顺序依次尝试将 $v$ 的邻居加入 $S$ 来构成新的团的方式完成.

对于极大团的父子关系, 我们首先给出根极大团的定义, 然后给出极大团间父关系的定义, 最后说明子极大团的计算方式. 根极大团 $K_0$: 图中字典序最大的极大团为根极大团, 根极大团没有父亲. 父关系: 若有极大团 $K$, $K$ 的父亲为 $P(K)=C(K_{\leqslant i-1})$, 其中 $K_{\leqslant i-1}=K\cap\{v_1,\ldots,v_{i-1}\}$, $i$ 要取使 $C(K_{\leqslant i-1})\neq K$ 的最大值, 这样的 $i$ 也被称为 $K$ 的父索引, 被记为 $i(K)$.

$K$ 的父亲 $P(K)$ 是通过 $K_{\leqslant i-1}$ 补全得到的, 反过来, $K$ 就可以通过 $P(K)_{\leqslant i}\cap N(v_i)\cup\{v_i\}$ 补全得到. 于是我们定义极大团 $K$ 的子节点为 $K[i]=C(K_{\leqslant i}\cap N(v_i)\cup\{v_i\})$, 并且为了让每个极大团都有唯一的父索引, 规定 $i$ 需要满足如下 4 个条件: (a) $v_i\notin K$; (b) $i>i(K)$; (c) 父极大团中序号小于等于 $i$ 且与 $v_i$ 相邻的点就是子极大团中所有序号小于等于 $i-1$ 的点, 即 $K[i]_{\leqslant i-1}=K_{\leqslant i}\cap N(v_i)$; (d) 由父极大团中序号小于 $i$ 且与 $v_i$ 相邻的点经过补全后得到的团, 和父极大团两者中序号小于 $i$ 的点是一致的, 即 $K_{\leqslant i}=C(K_{\leqslant i}\cap N(v_i))_{\leqslant i}$.

以图 1(b) 为例, 我们可以得到如图 5 所示的极大团枚举树. 其中 $\{1,2,3,4,5\}$ 为图 1(b) 中字典序最大的极大团, 作为枚举树的根节点. 计算 $K_0$ 的子节点时, 对于 $i=6$ 可得, $K_0[6]=C(\{1,2,3,4,5\}_{\leqslant 6}\cap N(6)\cup\{6\})=\{1,6\}$, $i$ 为 7 和 8 时同理. 计算 $K_1$ 的子节点时, 对于 $i=7$ 可得, $K_1[7]=C(\{1,6\}_{\leqslant 7}\cap N(7)\cup\{7\})=\{6,7,8\}$; 对于 $i=8$, $C(\{1,6\}_{\leqslant 8}\cap N(8)\cup\{8\})=\{6,7,8\}$, 而 $\{6,7,8\}_{\leqslant 8-1}\neq\{1,6\}_{\leqslant 8}\cap N(8)$, 不满足条件 (c).

**矩阵法.** 2004 年, Makino 等 [43] 对子节点 $i$ 值需要满足的 4 个条件中的 (c) 和 (d) 进行了细化扩

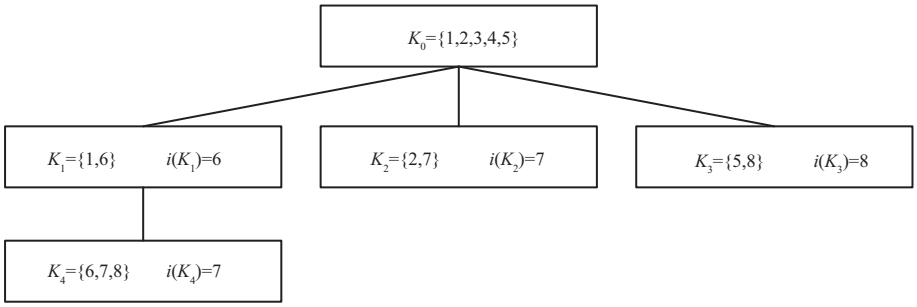


**图 5 图 1(b) 对应的极大团枚举树**

**Figure 5** The enumeration tree of maximal cliques corresponding to Figure 1(b)

展, 使得其可以通过矩阵运算得出符合条件的值, 保证了延迟为 $O(n^{2.37})$. 2015 年, Comin 等 [45] 改进了 Makino 的工作, 能够通过矩阵 (Makino 的矩阵为方阵, Comin 的矩阵为矩形矩阵) 一次计算多个极大团的所有子节点, 将延迟降低到了 $O(n^{2.09})$, 但是会占用更多的内存.

**枚举森林.** 2013 年, Chang 等 [44] 提出了枚举森林的概念, 来代替枚举树. 对于父极大团的定义, 其加入了 $K_{\leqslant i-1} \neq \phi$ 的限制, 即如果一个极大团中 id 最小的点没有 id 更小的邻居 (包括不在该极大团中的邻居) 时, 这个极大团就没有父节点. 这切断了部分极大团间的父子关系, 使得原有的枚举树成为了枚举森林. 这一改进大大降低了输出敏感算法的启动时间 (即寻找图中根极大团的时间). 除此之外, 作者还对高低度顶点采取了不同的处理方式. 通过这两种方式使得算法延迟达到 $O(\Delta h^3)$, $\Delta$ 是最大度数, $h$ 是使 $|\{v|v \in V, |N(v)| \geqslant h\}| \leqslant h$ 成立的最小的整数.

2016 年, Conte 等 [46] 在枚举森林的基础上, 通过非递归的深度搜索方式, 显著降低了算法的空间开销. 之前的输出敏感算法都是递归式的, 随着图规模的增长, 递归深度和递归的内存占用会急剧增加. 为了避免递归造成的内存爆炸, 作者不再采用递归的方式来保存搜索状态, 而是以时间换空间: 每次都根据极大团的父子关系, 计算出下一步的 (或需要回退到的) 搜索状态, 其中用到了逆向搜索 [75] 的相关思想. 这样的方式虽然引入了额外的计算开销, 但显著减小了程序的内存占用, 使其能处理更大规模的图. 作者在论文中指出, Eppstein 的算法 [53] (在 5.1.1 小节进行详细介绍) 比他们平均快 3.7 倍, 但同时要多消耗 878.9 倍的内存.

### 4.2 广度优先搜索算法

除了深度优先搜索的 BK 系列算法和输出敏感算法外, 也有一些工作采用广度优先搜索的方式来枚举极大团, 但相关工作要远少于深度优先搜索. 与深度优先搜索算法类似, 广度优先搜索算法也可以被进一步划分为以极大团为中心的算法和以顶点为中心的算法.

**以极大团为中心.** 2001 年, Kose 等 [38] 提出了第 1 个广度优先搜索的极大团枚举算法. 其用到了这样一个事实: 如果两个 $n$-clique 有 $n-1$ 个公共顶点, 则有可能合并为一个 $(n+1)$-clique. 举例来说, 对于两个有着 3 个公共顶点的 4-clique, $C_1\{1,2,3,4\}$ 和 $C_2\{1,2,3,5\}$, 如果存在 $C_3\{1,2,4,5\}$, 则 $C_1$, $C_2$, $C_3$ 可以合并成为一个 5-clique $1,2,3,4,5$. 于是, Kose 提出的算法从 2-clique 开始, 合并出所有 3-clique, 再合并出所有 4-clique, 一步步往下进行, 直到无法产生新的团为止. 该算法提供了一种极大团枚举算法的新思路, 避免了原始 BK 算法中的大量重复工作 (当时 $\text{BK}_{\text{pivot}}$ 算法还没被提出), 而且该算法能按照顶点数量从小到大的顺序输出所有的极大团, 但是其需要存储所有的 $k$-clique 用于枚举出 $(k+1)$-clique, 占用大量的内存.

**以顶点为中心.** 2017 年, Yu 等 [39] 提出另一种广度优先搜索的极大团枚举算法, 其以类似遍历边的方式构建极大团, 理论上在大型稀疏图上有 $O(|E|)$ 时间复杂度. 其设计了一种新的数据结构, 候选团映射表 (candidate map, CM), 用有序列表的方式保存所有的候选团. 在构建 CM 时, 对图中每个点 $v$, 遍历所有序号小于自身的邻居 $u$, 尝试将 $v$ 加入以 $u$ 为头顶点的所有团中, 或者用以 $u$ 为头顶点的团和 $v$ 的小邻居做交集来构建新的团. 在筛选和去重方面, Yu 采用树结构来存储已经枚举出的极大团. 如果两个极大团有着公共的起始部分, 则它们在存储树中会共享这部分的路径. 例如, 极大团 $1,2,3,4$ 和 $1,2,3,5$ 有着公共的起始部分 $1,2,3$, 则它们在存储树中会共享路径 $1\to 2\to 3$, 并在节点 3 后分别构建并列的叶子节点 4 和 5. 但鉴于构建 CM 的操作本身较为复杂, 作者指出这种方法仅适用于稀疏图, 如果一个点的邻居过多, 构建 CM 的时间会大幅提升. Yu 在 2019 年对自己的工作进行了改进 [51], 使用位集合来存储邻居, 用位操作来进行集合运算, 减小了存储和计算开销. 其还引入了顶点排序手段对搜索树进一步剪枝.

## 5 计算优化相关工作

除了改进算法本身, 通过各项计算优化手段降低算法运行时间也非常重要. 本节将会按照 3.2 小节给出的分类, 依次对分而治之、运行时优化、频繁操作优化的相关工作进行详细介绍.

### 5.1 分而治之 (partition based optimization)

在极大团枚举问题上, "分" 指的是将整个输入图进行划分, "治" 指的是在划分后的子图上应用已有的枚举算法. 根据划分次数, 相关工作可以被进一步分成单级和多级两类, 单级划分只会对输入图进行一次划分得到一系列子图, 多级划分则会对子图再进行一次或多次划分.

#### 5.1.1 单级划分

2009 年, Wu 等 [54] 通 MapReduce 将输入图划分成多个子图, 然后在子图上进行极大团枚举. 划分阶段, 对每个顶点 $k$, 获取 $k$ 的一跳邻居 $N(k)$ 以及两跳邻居 $N(N(k))$ 的信息, 以构建子图 $G^k_{\text{sub}}$, 为了去重, 在挖掘子图 $G^k_{\text{sub}}$ 时, 只输出以 $k$ 为最大顶点的极大团. 在子图中枚举极大团时, Wu 借用了 $\text{BK}_{\text{pivot}}$ 算法. 但其划分后的子图存在部分冗余信息, 会导致重复工作, 而且没有考虑负载均衡的问题, MapReduce 框架本身也会导致大量的中间结果和较大的通讯成本, 这些问题使得其在使用 64 个 reducers 时与 $\text{BK}_{\text{pivot}}$ 算法相比仅能实现数倍的加速效果.

2010 年, Lu 等 [56] 采用随机划分顶点集的方式, 提出了一种在非共享的机器集群上运行的分布式枚举算法. 其首先将顶点划分成多个固定大小的顶点集 $\{S_i\}$, 文中讨论了两种划分方式, 一是将相邻顶点尽可能划分在一起, 二是随机划分. 后者虽然会导致生成的诱导子图更大, 但前者会导致严重的负载不均衡, 相比而言, 随机划分的整体效果会更好. 在有了 $\{S_i\}$ 后, 就可以生成以 $S_i$ 及其邻域为顶点集的诱导子图 $G'(S_i)$, 并将子图送到不同的计算节点进行计算. 但分布式环境下得出的结果需要经过筛选, 这个后处理过程是非常耗时的, 甚至需要花费枚举阶段数倍至数十倍的时间.

2010 年, Eppstein 等 [53] 将退化度 (degeneracy) [76] 引入了极大团枚举问题, 改善了按邻域划分后, 子问题规模相差过大以及去重繁琐的问题. 退化度的定义是: 如果图的顶点存在一种序列, 使得以任意顶点及其前驱为顶点集的诱导子图中, 该顶点的度都不超过 $k$, 则 $k$ 能取到的最小值称为该图的退化度, 对应的序列即为退化顺序 (degeneracy ordering). 退化顺序可以在线性时间内快速的计算出来 [77]. 在计算出退化顺序后, 就可以将一个点的邻域分为两部分: $N^+(v)$ 表示 $v$ 的邻居中序号大于 $v$ 的部

分, $N^-(v)$ 表示 $v$ 的邻居中序号小于 $v$ 的部分. 接下来对每个顶点及其邻域应用 $\mathrm{BK}_{\mathrm{pivot}}$ 算法时, 就可以将 $N^+(v)$ 作为初始候选集, 将 $N^-(v)$ 作为初始禁止集. 而且经过这样的划分, 每个点的 $N^+(v)$, 相比 $N(v)$ 而言, 会变得均匀很多而且小很多.

2010 年, Cheng 等 [31] 提出了$H$*-graph的概念, 通过图划分的方式, 解决无法放入内存的大型图的极大团枚举问题. 对于图 $G(V,E)$, 定义顶点集 $H=\{v \mid v\in V, d(v)\geqslant h\}$ 且 $|H|=h$, 而且图中剩余顶点的度数都不超过 $h$. $H^+=H\cup N(H)$, $G_{H^+}$ 为 $H^+$ 的诱导子图, $G_{H^+}$ 中去掉 $N(H)$ 间的边就得到了 $G_{H^*}$. 经过实验, $H$*-graph 中边的数量仅为整个图的 12%∼15%, 而且该比例还会随着整体顶点数量增加而下降. 2011 年, Cheng 等改进了图划分的方式 [57], 对于静态图, 每次顺序选取一组顶点形成顶点集 $B$, 用以生成 $G_{B^*}$. $G_{B^*}$ 的生成方式与 $G_{H^*}$ 一致, 只不过将顶点集 $H$ 替换为了 $B$. 这样的方式, 避免了重复对图 G 进行遍历, 而且可以并行的生成所有的 $G_{B^*}$, 方便与现有的并行算法进行结合.

2014 年, Svendsen 等 [55] 使用特殊的顶点排序手段, 改善了 MapReduce 上子问题间的重复计算问题以及负载不均衡问题. 其首先规定, 在点 $v$ 及其邻域上进行枚举的 Reducer 只输出以 $v$ 为最小顶点的极大团. 在这样的规定下, 可以看到 $v$ 的序号越大, 对应 Reducer 中输出的极大团就越少; 所以对于一个包含更多极大团的 $N(v)$, $v$ 的序号应该更大以输出较少的极大团, 反之 $v$ 的序号应该更小. 但由于精确计算 $N(v)$ 中包含的极大团个数与极大团枚举问题本身差别不大, 所以作者采用了两种简单的排序方式进行近似: 按度排序和按三角形计数排序.

2017 年, Manoussakis [58] 通过退化度排序和按邻域划分, 提出了一种延迟只与退化度 $d$ 有关的输出敏感算法, 但其需要存储所有已经枚举出的极大团用于筛选去重. 其算法首先计算顶点的退化度排序, 然后生成以每个顶点和序号大于它的邻居作为点集的诱导子图, 在每个诱导子图上都采用已有的输出敏感算法进行极大团枚举, 最后再对枚举出的团进行筛选. 在筛选过程中, 其将枚举出的团视为一个由顶点编号构成的字符串, 并利用字符串问题中的后缀树 [78] 来判断该团是否为重复的或非极大的 (由顶点序号组成的字符串在后缀树中是否有匹配项). 作者还在 2019 年提出了另一种筛选机制 [59], 对于一个在以 $v_i$ 和其邻居中序号大于它的点作为点集的诱导子图中的极大团 $C$, 如果原图中存在一个序号小于 $v_i$ 且与 $C$ 中点都相邻的点, 则 $C$ 在原图中是非极大的. 新的筛选机制计算开销要比后缀树的方式大, 但并不要求存储已经枚举出的极大团, 可以针对任务规模选取不同的方式.

### 5.1.2 多级划分

2012 年, Cheng 等 [60] 提出了一种基于成本模型的嵌套划分方法, 改进了自己之前的工作, 在一定程度上兼顾了内存限制与枚举效率. 作者指出, 枚举过程中, 集合的交集操作是一个不可忽视的开销, 而划分后的子图越小, 在其上的交集开销也会更小, 但同时也会使得子图数量更多. 为此, 作者构建了一个成本模型用于确定合适的子图规模. 在有了成本模型后, 整个图划分过程被分为两层, 第 1 层为了减小 IO 开销, 在能放入内存的情况下构建尽可能大的子图, 第 2 层为了减小 CPU 开销, 将第 1 层构建的子图通过成本模型, 进一步划分成一系列更小的子图. 其实验表明, 该方法与 Eppstein 等 [53] 提出的方法相比, 消耗的内存要小数倍, 性能方面要比 Eppstein 慢一个数量级左右.

受到 Cheng 工作的影响, 有一些工作进一步挖掘了多级划分的优势. 2016 年, Conte 等 [61] 提出了一种递归式的两级划分方法. 第 1 级划分会将整个顶点集中的非高度顶点的抽取出来, 第 2 级将第 1 级抽取出来的顶点进一步划分成大小适合的块 (子图) 并在其上应用已有的枚举算法. 而对于剩余的高度顶点部分, 在将非高度顶点划分出去之后, 高度顶点的度数会大幅降低, 于是可以对原来的高度顶点部分再执行相同策略的划分和枚举, 不断递归直到处理完所有顶点. 除此之外, Conte 还以子图的不同特征 (包含顶点数、边数、退化度等) 作为输入, 以不同的图存储方式和不同算法在其上的枚举

时间作为结果训练出了一个决策树以期望对不同的子图应用最优的存储方式和枚举算法, 来进一步发挥各自的优势. 其实验结果也显示了该决策树是有效的.

同样在 2016 年, Chen 等 [62] 提出了一种嵌套邻域划分的极大团枚举算法. 其思路是, 首先以图中的每个点及其邻域形成诱导子图, 将这些子图放入一个待处理队列, 然后开始枚举过程. 枚举过程中, 每次从待处理队列中取出一个子图, 选取其中度数最小的点 $u$, 并将子图进一步划分为两部分: 一部分为该子图中 $u$ 和其邻居形成的诱导子图; 另一部分为该子图中去掉 $u$ 和 $u$ 的边后的剩余部分. 对于第 1 部分, Chen 又将其分为 3 种情况进行处理: 当其为一个极大团时, 进行输出; 当其包含的顶点数目小于一个阈值时, 对其进行极大团枚举; 若都不符合则将其放入待处理队列尾部. 对于剩余部分, 则继续选取其中度数最小的点并按照上面的方式划分, 直到某次划分后, 剩余部分为一个极大团. 输出该极大团后, 从队列中拿取新的任务 (子图) 进行下一轮计算.

### 5.2 运行时优化 (runtime optimization)

2009 年, Schmidt 等 [63] 提出了一种基于搜索树分解的并行方法, 是一个并行版的 $\mathrm{BK}_{\mathrm{pivot}}$ 算法, 并且使用了 "工作偷取 (work stealing)" 的动态负载均衡方法. 其使用了栈来模拟递归过程, 从而显式的保存搜索状态 (BK 算法的搜索树节点) 以实现并行和负载调度. 工作偷取的具体措施是: 空闲的线程会以随机轮询的方式拿取其他线程的栈底的一个或多个搜索树节点作为自己接下来的工作. 其实验表明, 该方式能达成近乎完美的负载均衡, 并且有着极好的扩展性, 但由于其复杂的动态负载均衡策略和栈操作的开销, 该方法的总运行时间可能并不理想.

2017 年, Lessley 等 [64] 用数据并行原语 (data-parallel primitives, DPPs) 在共享内存、多核架构下实现了 Kose 等 [38] 算法. 作者还通过 hash 操作加速了判断两个团是否共享顶点的过程: 将 $k$-cliques 合并成 $(k+1)$-cliques 时, 其首先对 $k$-cliques 的后 $k-1$ 个点进行 hash 运算; 有着相同 hash 值的团 (极) 可能会共享 $k-1$ 个顶点, 这样可以减少尝试合并团时的搜索范围. 因此选择的 hash 算法的优劣 (产生的 hash 碰撞越少越好) 会直接影响该枚举方法的性能. 除此之外, 作者还利用了 GPU 进行加速 (但作者并没有给出其在 GPU 上的任务分配方式和具体实现方法), 这一系列方法虽然极大的提高了 Kose 算法的枚举效率, 但仍没有解决其对于内存的极高需求.

2018 年, Das 等 [65] 提出了一种基于共享内存并行的极大团枚举方法, 对 $\mathrm{BK}_{\mathrm{pivot}}$ 进行了细粒度并行化. 将 $\mathrm{BK}_{\mathrm{pivot}}$ 并行化的第 1 个关键是选取 pivot, 这个是比较简单的, 原有的操作是依次计算候选集和禁止集中的点在候选集中的邻居个数, 其中是没有依赖关系的, 可以直接将循环展开进行并行计算. 第 2 个关键是扩展过程的并行化, 原有的操作是依次对 pivot 和 pivot 的非邻居进行扩展, 期间会更新候选集和禁止集, 无法直接并行. 文中采用的方法是, 首先对图中有所顶点进行排序, 这样每一步顶点都是按序存储的, 这样在扩展某个顶点时, 就可以将序号比该点小的点直接从候选集移入禁止集, 打破了原有的依赖关系, 就可以将循环展开进行并行计算. 这样就得到了并行的 $\mathrm{BK}_{\mathrm{pivot}}$. 文章还测试了不同顶点排序方式对于并行效果的影响.

### 5.3 频繁操作优化 (frequent operation optimization)

加速频繁操作是一种很直接的优化思路, 在极大团枚举问题中最频繁的操作就是集合运算. 采用基于位的数据结构可以使集合运算变为位运算, 是加速集合运算的一种有效手段, 其也被应用在极大团枚举问题中.

2005 年, Zhang 等 [66] 采用位向量的方式表示点的邻居和团的公共邻居, 优化了 Kose 等 [38] 算法的效率, 但并没与解决内存需求的问题.

**表 3　极大团枚举问题的常用测试数据集**

**Table 3**　Commanly used graph datasets in the MCE problem

| Graph | # vertex | # edge | # MC | Graph | # vertex | # edge | # MC |
|---|---|---|---|---|---|---|---|
| com-Youtube | 1134890 | 2987624 | 3265956 | cit-Patents | 3774768 | 16518947 | 14787032 |
| web-Google | 875713 | 4322051 | 1417580 | wiki-talk | 2294385 | 4659565 | 86333306 |
| as-skitter | 1696415 | 11095298 | 37322355 | email-EuAll | 265009 | 420045 | 377751 |
| roadNet-CA | 1965206 | 2766607 | 2537996 | | | | |

**表 4　最近的有代表性的 MCE 方法在表 3 所列数据集上的计算时间 (秒)**

**Table 4**　The computing time (s) of recent and typical methods for the MCE problem over the graphs in Table 3

| Dataset | PECO<br>Svendsen [55] | Xu [49] | Wu [54] | Hashing<br>Lessley [64] | ParMCE<br>Das [65] | CMC-bit<br>Yu [39] | $\text{MCE}_{\text{hybrid}}$<br>Li [50] |
|---|---|---|---|---|---|---|---|
| com-Youtube | | 7.02 | 982 | | | 58 | 3.34 |
| cit-Patents | 113 | 8.50 | 2868 | 3.27 | | 133 | 17 |
| web-Google | 126 | 5.84 | 1178 | | | 32 | 1.90 |
| wiki-talk | >10000 | 219 | >10000 | | 62 | | 241 |
| as-skitter | >8140 | 140 | >10000 | | 45 | | 128 |
| email-EuAll | | | | 2.24 | | 4.24 | 0.23 |
| roadNet-CA | | | | 0.27 | | 17 | 0.59 |

2014 年, Dasari 等 [67] 用局部位邻接矩阵代替邻接表, 减小了工作集的大小, 提高数据局部性, 并且加速了集合的交集运算. 在经过退化度排序后, 假设顶点 $v$ 有 $p$ 个大于它的邻居 $N^+(v)$ 和 $x$ 个小于它的邻居 $N^-(v)$, 由于不需要 $N^-(v)$ 间的边信息, 构建出的局部位邻接矩阵大小为 $(p+x)p$ 位. 此外还需要对 $N^+(v)$ 和 $N^-(v)$ 中的顶点序号进行一次映射, 将 $N^+(v)$ 中的序号映射到 $[0, p-1]$, $N^-(v)$ 中的序号映射到 $[p, p+x-1]$. 采用同样的映射, 为候选集也构建为一个 $p$ 位的位向量. 为了进一步减少工作集的大小, 当候选集中点的个数减到 128 和 32 个 (此为实验得出的经验值) 时, 会以相同方式重新构建局部位邻接矩阵.

## 6　总结与展望

我们在表 3 中给出了部分极大团枚举问题上最常用的测试数据集[1)], 它们都是来自不同领域的现实图数据. 表 4 中列出了近些年来的具有代表性的一些算法在表 3 的数据集上的性能表现, 但由于缺乏开源项目, 表 4 中各算法 (除了 $\text{MCE}_{\text{hybrid}}$) 的数据来源于其原论文. 对于 Li 等提出的 $\text{MCE}_{\text{hybrid}}$ 方法, 我们拿到了其代码[2)]并在一台 CPU 型号为 Intel Xeon Processor E5-2680 v4、内存 250 G 的 linux 服务器上进行了测试. 从表 4 中可以看出, 在一些中小规模的图上, 如 com-Youtube, web-Google 等, 现有的算法已经有着很好的效果, 能够很快地枚举出其中的所有极大团. 但对于大规模图, 尤其是包含大量极大团的大规模图, 如 wiki-talk, as-skitter 等, 现有的方法仍旧需要很长的计算时间, 而且考虑到现有的并行算法可能使用了几十倍于串行算法的计算资源, 其所能到达的加速效果并不理想. 下面, 我们结合前文的介绍和分析对极大团枚举问题的研究现状和未来发展进行总结和展望.

1) https://snap.stanford.edu/data/.

2) https://github.com/CGCL-codes/BKrcd.

纵观现有的几种极大团枚举算法, 虽然输出敏感算法在复杂度上有着理论上的优势, 但现实图绝大部分情况下不会达到 BK 系列算法的最坏情况, 在实践中的绝大部分情况下还是 BK 系列算法表现得更加优秀. 而对于 Kose 等 [38] 提出的算法, Lessley 等 [64] 的改进虽然使得其枚举时间得到了较大的缩减, 但完全没有解决该算法对于内存的极高需求, 特别是随着图规模的不断变大, 加之其共享内存的并行实现, 内存问题会变得更为突出.

结合现有算法的特点及应用场景的数据规模, 我们认为, MCE 问题的发展存在以下几点挑战:

(1) **测试数据集偏小.** 现有工作的测试数据集顶点规模一般不超过百万级, 而 Twitter 用户数已经超过了 5 亿, Facebook 的日活跃用户数已经超过了 10 亿, 每位用户都是社交图的一个顶点. 现有的工作, 尤其是 Kose 等 [38] 提出的方法, 受限于内存需求, 无法很好地处理大规模的现实图. Conte 等 [46] 以及 Cheng 等 [60] 的工作虽然采用不同的方式解决了内存使用的问题, 但是一个引入了额外的计算开销, 一个引入了复杂的图划分开销, 都对枚举时间带来了不小的影响.

(2) **并行的可扩展性不高.** 现有的并行算法受限于图划分方式、通讯开销、筛选开销、负载均衡问题等, 其可扩展性都不高. Wu 等 [54] 只能在 20 个 reducers 以下维持线性加速比, Cheng 等 [60] 方法的并行版本只能在 4 台机器以下的集群上维持线性加速比, Svendsen 等 [55] 只能在 16 个 reducers 以下维持线性加速比等. 这使得现有的并行算法并不能充分利用现有的强大的并行计算资源.

(3) **负载均衡问题.** 在并行计算中, 负载均衡是一个非常关键的问题, 然而极大团枚举的子问题规模是极难预先判断的. 现有的工作要么采用预先对顶点排序的静态方式, 使子问题规模尽量均衡; 要么采用动态负载均衡的方式, 在求解过程中不断平衡各个计算单元间的负载. 顶点排序中, Eppstein 等 [53] 引入的退化度排序虽然排序效果更好, 但其计算有着极高的数据依赖, 目前还没有人提出并行化的退化度排序算法, 现有的并行算法一般采用按顶点度排序的方式代替退化度排序. 而动态负载均衡虽然能实现较为完美的负载均衡效果, 但会带来很大的额外开销. 所以, 负载均衡问题也是极大团枚举中仍未被完全解决的一个关键问题.

结合极大团枚举的研究现状, 以及其他领域的一些研究热点, 我们对于极大团枚举问题接下来的发展方向, 有以下一些思考:

(1) **基于位的数据结构.** 采用局部位邻接矩阵 [66] 与位向量 [67] 的两个工作都取得了不错的效果, 同时 Segundo 等 [68] 也指出, 原有的退化度排序和 $\mathrm{BK}_{\mathrm{pivot}}$ 算法并不是完全适合基于位的数据结构, 他也只是给出了一种解决方式. 那么, 是否存在其他不同的解决方式甚至完全不同的算法, 能够从位数据结构中获得更大的收益? 位数据结构能否被应用在分布式环境下, 以减少其通讯开销和筛选开销?

(2) **分布式图计算系统.** 过去一些年里, 有很多大型分布式图计算系统被提出 [79], 例如以顶点为中心的 Pregel [80], PowerGraph [81], 以子图为中心的 GoFFish [82], NScale [83] 等, 这些系统在大型图的 PageRank, SSSP, BFS 等任务上发挥了巨大的作用. 那么, 极大团枚举问题能否从中受益呢?

(3) **图神经网络.** 近几年随着图神经网络的提出和发展 [84], 越来越多的基于图数据的分类和预测可以由图神经网络完成, 例如顶点分类、社会影响预测、分子性质预测、交通流量预测等. 对于极大团枚举问题, 我们是否能通过训练好的图神经网络来对子问题的运算时间进行预测呢? 如果我们能通过这种方式快速、较准确地获得子问题的运算时间, 是否能解决负载均衡的问题?

(4) **以边为中心.** BK 系列算法是从顶点的角度考虑如何发现一个极大团, 这是一种很直观的方式, 那么能否从边的角度出发来枚举极大团呢? 但不得不承认的是, 一般情况下图中的边数要大于甚至远大于顶点数, 而且从边出发进行极大团枚举是全新的角度, 很难预测其效果.

(5) **利用硬件加速器.** 随着 GPU 的发展, 许多领域都开始使用 GPU 加速运算 [85, 86], 但在极大团枚举方面, 相关研究还极为有限. Lessley 等 [64] 利用了 GPU 的数据并行原语加速极大团枚举, 并取得

了一定的效果. 除了 GPU 外, FPGA 在过去几年也得到了迅速发展 [87], 并且得到了广泛的应用 [88, 89]. GPU 以及 FPGA 能否被有效地利用在极大团枚举问题上, 我们认为这也是一个值得研究的方向.

总之, 极大团枚举问题作为图论中的一个基础问题, 在生物、化学、社交网络等领域都有着较为广泛的应用, 这些年来针对极大团枚举问题人们做出了广泛的探索尝试, 也取得了一系列的成果. 但随着社会的发展, 各种图数据的增长越来越快, 现有的算法已经难以处理当前的大型图, 该问题还有很多值得探索和研究的方向.

## 参考文献


1 Méndez-Díaz I, Zabala P. A branch-and-cut algorithm for graph coloring. Discrete Appl Math, 2006, 154: 826–847
2 Balliu A, Brandt S, Hirvonen J, et al. Lower bounds for maximal matchings and maximal independent sets. In: Proceedings of IEEE Annual Symposium on Foundations of Computer Science, Baltimore, 2019. 481–497
3 Tpfer A, Marschall T, Bull R A, et al. Viral quasispecies assembly via maximal clique enumeration. PLoS Comput Biol, 2014, 10: 3
4 Rokhlenko O, Wexler Y, Yakhini Z. Similarities and differences of gene expression in yeast stress conditions. Bioinformatics, 2007, 23: 184–190
5 Chen Y, Crippen G M. A novel approach to structural alignment using realistic structural and environmental information. Protein Sci, 2005, 14: 2935–2946
6 Zhang B, Park B H, Karpinets T, et al. From pull-down data to protein interaction networks and complexes with biological relevance. Bioinformatics, 2008, 24: 979–986
7 Matsunaga T, Yonemori C, Tomita E, et al. Clique-based data mining for related genes in a biomedical database. BMC Bioinform, 2009, 10: 1–9
8 Yu H, Paccanaro A, Trifonov V, et al. Predicting interactions in protein networks by completing defective cliques. Bioinformatics, 2006, 22: 823–829
9 Koichi S, Arisaka M, Koshino H, et al. Chemical structure elucidation from 13C NMR chemical shifts: efficient data processing using bipartite matching and maximal clique algorithms. J Chem Inf Model, 2014, 54: 1027–1035
10 Wen X, Chen W N, Lin Y, et al. A maximal clique based multiobjective evolutionary algorithm for overlapping community detection. IEEE Trans Evol Comput, 2016, 21: 363–377
11 Leskovec J, Huttenlocher D, Kleinberg J. Signed networks in social media. In: Proceedings of SIGCHI Conference on Human Factors in Computing Systems, Atlanta, 2010. 1361–1370
12 Lu Z, Wahlstrm J, Nehorai A. Community detection in complex networks via clique conductance. Sci Report, 2018, 8: 1–16
13 Biswas K, Muthukkumarasamy V, Sithirasenan E. Maximal clique based clustering scheme for wireless sensor networks. In: Proceedings of the 8th International Conference on Intelligent Sensors, Sensor Networks and Information Processing, Melbourne, 2013. 237–241
14 Bao X G, Wang L Z. A clique-based approach for co-location pattern mining. Inf Sci, 2019, 490: 244–264
15 Hosseini S S, Wormald N, Tian T. A weight-based information filtration algorithm for stock-correlation networks. Phys A-Stat Mech Appl, 2021, 563: 125489
16 Moon J W, Moser L. On cliques in graphs. Israel J Math, 1965, 3: 23–28
17 Valiant L G. The complexity of enumeration and reliability problems. SIAM J Comput, 1979, 8: 410–421
18 T'kindt V, Bouibede-Hocine K, Esswein C. Counting and enumeration complexity with application to multicriteria scheduling. 4OR, 2005, 3: 1–21
19 Conte A, Grossi R, Marino A, et al. Listing maximal independent sets with minimal space and bounded delay. In: Proceedings of International Symposium on String Processing and Information Retrieval, Palermo, 2017. 144–160
20 Firyulina O S. Finding all maximal independent sets of an undirected graph. Vestnik S Petersburg Univ Ser 10 Prikl Mat Inform Prots Upr, 2013, 1: 63–69
21 Singh K K, Pandey D A K. Survey of algorithms on maximum clique problem. Int Adv Res J Sci Eng Tech, 2015, 2: 15–20
22 Tsourakakis C. The k-clique densest subgraph problem. In: Proceedings of the 24th International Conference on World

Wide Web, Florence, 2015. 1122–1132
23 Danisch M, Balalau O, Sozio M. Listing k-cliques in sparse real-world graphs. In: Proceedings of the World Wide Web Conference, Lyon, 2018. 589–598
24 Fulkerson D, Gross O. Incidence matrices and interval graphs. Pac J Math, 1965, 15: 835–855
25 Prisner E. Graphs with few cliques. In: Proceedings of the 7th Conference on the Theory and Applications of Graphs, 1995. 945–956
26 Spinrad J P. Efficient Graph Representations. Providence: American Mathematical Society, 2003
27 Chiba N, Nishizeki T. Arboricity and subgraph listing algorithms. SIAM J Comput, 1985, 14: 210–223
28 Jerrum M. Large cliques elude the Metropolis process. Random Struct Alg, 1992, 3: 347–359
29 Blair J R S, Peyton B. An introduction to chordal graphs and clique trees. In: Graph Theory and Sparse Matrix Computation. Berlin: Springer, 1993. 1–29
30 Stix V. Finding all maximal cliques in dynamic graphs. Comput Optim Appl, 2004, 27: 173–186
31 Cheng J, Ke Y P, Fu A W C, et al. Finding maximal cliques in massive networks by h*-graph. In: Proceedings of ACM SIGMOD International Conference on Management of Data, Indianapolis, 2010. 447–458
32 Xu Y Y, Cheng J, Fu A W C. Distributed maximal clique computation and management. IEEE Trans Serv Comput, 2015, 9: 110–122
33 Sun S L, Wang Y M, Liao W L, et al. Mining maximal cliques on dynamic graphs efficiently by local strategies. In: Proceedings of the 33rd International Conference on Data Engineering, San Diego, 2017. 115–118
34 Das A, Svendsen M, Tirthapura S. Incremental maintenance of maximal cliques in a dynamic graph. VLDB J, 2019, 28: 351–375
35 Das A, Sanei-Mehri S V, Tirthapura S. Shared-memory parallel maximal clique enumeration from static and dynamic graphs. ACM Trans Parall Comput, 2020, 7: 1–28
36 Mukherjee A P, Xu P, Tirthapura S. Enumeration of maximal cliques from an uncertain graph. IEEE Trans Knowl Data Eng, 2017, 29: 543–555
37 Li R H, Dai Q, Wang G, et al. Improved algorithms for maximal clique search in uncertain networks. In: Proceedings of the 35th International Conference on Data Engineering, Macao, 2019. 1178–1189
38 Kose F, Weckwerth W, Linke T, et al. Visualizing plant metabolomic correlation networks using clique-metabolite matrices. Bioinformatics, 2001, 17: 1198–1208
39 Yu T, Liu M C. A linear time algorithm for maximal clique enumeration in large sparse graphs. Inf Process Lett, 2017, 125: 35–40
40 Bron C, Kerboscht J. Finding all cliques of an undirected graph. Commun ACM, 1973, 16: 575–580
41 Tomita E, Tanaka A, Takahashi H. The worst-case time complexity for generating all maximal cliques. In: Proceedings of the 10th International Computing and Combinatorics Conference, Jeju Island, 2004. 161–170
42 Tsukiyama S, Ide M, Ariyoshi H, et al. A new algorithm for generating all the maximal independent sets. SIAM J Comput, 1977, 6: 505–517
43 Makino K, Uno T. New algorithms for enumerating all maximal cliques. In: Proceedings of the 9th Scandinavian Workshop on Algorithm Theory, Humlebaek, 2004. 260–272
44 Chang L, Yu J X, Qin L. Fast maximal cliques enumeration in sparse graphs. Algorithmica, 2013, 66: 173–186
45 Comin C, Rizzi R. An improved upper bound on maximal clique listing via rectangular fast matrix multiplication. 2015. ArXiv:1506.01082
46 Conte A, Grossi R, Marino A, et al. Sublinear-space bounded-delay enumeration for massive network analytics: maximal cliques. In: Proceedings of the 43rd International Colloquium on Automata, Languages, and Programming, Rome, 2016
47 Koch I. Enumerating all connected maximal common subgraphs in two graphs. Theor Comput Sci, 2001, 250: 1–30
48 Du N, Wu B, Xu L T, et al. A parallel algorithm for enumerating all maximal cliques in complex network. In: Proceedings of the 6th IEEE International Conference on Data Mining-Workshops, Hong Kong, 2006. 320–324
49 Xu Y Y, Cheng J, Fu A W C, et al. Distributed maximal clique computation. In: Proceedings of IEEE International Congress on Big Data, Anchorage, 2014. 160–167
50 Li Y N, Shao Z Y, Yu D X, et al. Fast maximal clique enumeration for real-world graphs. In: Proceedings of the 24th International Conference on Database Systems for Advanced Applications, Chiang Mai, 2019. 641–658

51 Yu T, Liu M C. A memory efficient maximal clique enumeration method for sparse graphs with a parallel implementation. Parall Comput, 2019, 87: 46–59

52 Xu Y Y, Cheng J, Fu A W C, et al. Distributed maximal clique computation. In: Proceedings of IEEE International Congress on Big Data, Anchorage, 2014. 160–167

53 Eppstein D, Lffler M, Strash D. Listing all maximal cliques in sparse graphs in near-optimal time. In: Proceedings of International Symposium on Algorithms and Computation, Jeju Island, 2010. 403–414

54 Wu B, Yang S Q, Zhao H Z, et al. A distributed algorithm to enumerate all maximal cliques in MapReduce. In: Proceedings of the 4th International Conference on Frontier of Computer Science and Technology, Shanghai, 2009. 45–51

55 Svendsen M, Mukherjee A P, Tirthapura S. Mining maximal cliques from a large graph using MapReduce: tackling highly uneven subproblem sizes. J Parall Distrib Comput, 2015, 79-80: 104–114

56 Lu L, Gu Y H, Grossman R. dMaximalCliques: a distributed algorithm for enumerating all maximal cliques and maximal clique distribution. In: Proceedings of International Conference on Data Mining Workshops, Sydney, 2010. 1320–1327

57 Cheng J, Ke Y P, Fu A W C, et al. Finding maximal cliques in massive networks. ACM Trans Database Syst, 2011, 36: 1–34

58 Manoussakis G. An output sensitive algorithm for maximal clique enumeration in sparse graphs. In: Proceedings of the 12th International Symposium on Parameterized and Exact Computation, Vienna, 2017

59 Manoussakis G. A new decomposition technique for maximal clique enumeration for sparse graphs. Theor Comput Sci, 2019, 770: 25–33

60 Cheng J, Zhu L H, Ke Y P, et al. Fast algorithms for maximal clique enumeration with limited memory. In: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Beijing, 2012. 1240–1248

61 Conte A, de Virgilio R, Maccioni A, et al. Finding all maximal cliques in very large social networks. In: Proceedings of the 19th International Conference on Extending Database Technology, Bordeaux, 2016. 173–184

62 Chen Q, Fang C, Wang Z, et al. Parallelizing maximal clique enumeration over graph data. In: Proceedings of the 21st International Conference on Database Systems for Advanced Applications, Dallas, 2016. 249–264

63 Schmidt M C, Samatova N F, Thomas K, et al. A scalable, parallel algorithm for maximal clique enumeration. J Parall Distrib Comput, 2009, 69: 417–428

64 Lessley B, Perciano T, Mathai M, et al. Maximal clique enumeration with data-parallel primitives. In: Proceedings of the 7th Symposium on Large Data Analysis and Visualization, Phoenix, 2017. 16–25

65 Das A, Sanei-Mehri S V, Tirthapura S. Shared-memory parallel maximal clique enumeration. In: Proceedings of the 25th International Conference on High Performance Computing, Bengaluru, 2018. 62–71

66 Zhang Y, Abu-Khzam F N, Baldwin N E, et al. Genome-scale computational approaches to memory-intensive applications in systems biology. In: Proceedings of ACM/IEEE Conference on Supercomputing, Seattle, 2005

67 Dasari N S, Desh R, Zubair M. pbitMCE: a bit-based approach for maximal clique enumeration on multicore processors. In: Proceedings of the 20th IEEE International Conference on Parallel and Distributed Systems, Hsinchu, 2014. 478–485

68 San Segundo P, Artieda J, Strash D. Efficiently enumerating all maximal cliques with bit-parallelism. Comput Oper Res, 2018, 92: 37–46

69 Harary F, Ross I C. A procedure for clique detection using the group matrix. Sociometry, 1957, 20: 205–215

70 Bonner R E. On some clustering techniques. IBM J Res Dev, 1964, 8: 22–32

71 Bednarek A R, Taulbee O E. On maximal chains. Rev Roumaine Math Pures Appl, 1966, 11: 23–25

72 Augustson J G, Minker J. An analysis of some graph theoretical cluster techniques. J ACM, 1970, 17: 571–588

73 Mulligan G D, Corneil D G. Corrections to Bierstone's algorithm for generating cliques. J ACM, 1972, 19: 244–247

74 Khaouid W, Barsky M, Srinivasan V, et al. K-core decomposition of large networks on a single PC. Proc VLDB Endow, 2015, 9: 13–23

75 Avis D, Fukuda K. Reverse search for enumeration. Discrete Appl Math, 1996, 65: 21–46

76 Lick D R, White A T. k-Degenerate graphs. Can J Math, 1970, 22: 1082–1096

77 Batagelj V, Zaversnik M. An O(m) algorithm for cores decomposition of networks. 2003. ArXiv:cs/0310049

78 Ukkonen E. On-line construction of suffix trees. Algorithmica, 1995, 14: 249–260
79 Batarfi O, Shawi R E, Fayoumi A G, et al. Large scale graph processing systems: survey and an experimental evaluation. Cluster Comput, 2015, 18: 1189–1213
80 Malewicz G, Austern M H, Bik A J C, et al. Pregel: a system for large-scale graph processing. In: Proceedings of ACM SIGMOD International Conference on Management of data, Indianapolis, 2010. 135–146
81 Gonzalez J E, Low Y, Gu H, et al. Powergraph: distributed graph-parallel computation on natural graphs. In: Proceedings of the 10th USENIX Symposium on Operating Systems Design and Implementation, Hollywood, 2012. 17–30
82 Simmhan Y, Kumbhare A, Wickramaarachchi C, et al. Goffish: a sub-graph centric framework for large-scale graph analytics. In: Proceedings of European Conference on Parallel Processing, Porto, 2014. 451–462
83 Quamar A, Deshpande A, Lin J. NScale: neighborhood-centric large-scale graph analytics in the cloud. VLDB J, 2016, 25: 125–150
84 Wu Z H, Pan S R, Chen F W, et al. A comprehensive survey on graph neural networks. IEEE Trans Neural Netw Learn Syst, 2021, 32: 4–24
85 Shi X H, Zheng Z G, Zhou Y L, et al. Graph processing on GPUs: a survey. ACM Comput Sur, 2018, 50: 1–35
86 Navarro C A, Hitschfeld-Kahler N, Mateu L. A survey on parallel computing and its applications in data-parallel problems using GPU architectures. Commun Comput Phys, 2014, 15: 285–329
87 Kuon I, Tessier R, Rose J. FPGA architecture: survey and challenges. FNT Electron Des Autom, 2007, 2: 135–253
88 Besta M, Stanojevic D, Licht J D F, et al. Graph processing on FPGAs: taxonomy, survey, challenges. 2019. ArXiv:1903.06697
89 Guo K Y, Zeng S L, Yu J C, et al. A survey of FPGA-based neural network accelerator. 2018. ArXiv:1712.08934

# Maximal clique enumeration problem on graphs: status and challenges

Shaoxian XU[1,2,3,4], Xiaofei LIAO[1,2,3,4], Zhiyuan SHAO[1,2,3,4*], Qiangsheng HUA[1,2,3,4] & Hai JIN[1,2,3,4]

1. *School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan* 430074*, China*;
2. *National Engineering Research Center for Big Data Technology and System, Wuhan* 430074*, China*;
3. *Key Laboratory of Services Computing Technology and System, Ministry of Education, Wuhan* 430074*, China*;
4. *Key Laboratory of Cluster and Grid Computing, Hubei Province, Wuhan* 430074*, China*
* Corresponding author. E-mail: zyshao@hust.edu.cn

**Abstract** In the era of big data, graph mining has become a popular research topic. The maximal clique enumeration (MCE), as a basic problem in graph theory, has been widely used in different fields. However, considering the complexity of the MCE problem and the rapid growth in the scale of real-world graphs, enumerating the maximal cliques in real-world graphs is time-consuming. A large number of researches have been performed to improve the algorithm for the MCE problem and to reduce the execution time by applying various computational optimizations. For the MCE problem, this survey has conducted the following works: existing research works on the MCE problem are classified, the research status of the MCE problem is introduced in detail, and the challenges and future directions of the MCE problem are discussed.

**Keywords** maximal clique enumeration, graph theory, graph mining, graph partition, parallel computing