计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.181102109

基于局部有限搜索的无向图近似最大团快速求解算法

钟茂生[1,2]　江　超[2]　陶　兰[2]　何　雄[2]　罗远胜[3]
1 江西师范大学计算机信息工程学院　南昌 330022
2 华东交通大学信息工程学院　南昌 330013
3 江西财经大学网络信息管理中心　南昌 330013
(zhongmaosheng@sina.com)

摘　要　无向图最大团求解是一个著名的NP-完全问题,解决该问题的经典算法基本上都采用完全精确搜索策略。鉴于NP-完全问题本身所固有的复杂性,这些算法或许仅适用于某些特殊的小规模图,对于具有大规模顶点和边的复杂图还是显得无力,难以适用。针对完全精确搜索策略下的无向图最大团求解算法的大部分时间都用于对图进行额外而无效的查找的问题,采用分划递归技术将图划分为邻接子图和悬挂子图,然后对邻接子图进行递归求解,而对悬挂子图则通过设置搜索范围控制函数进行局部有限搜索。在DIMACS数据集上将所提算法与当前主要的最大团求解算法进行对比实验,结果表明,文中提出的局部有限搜索求解策略能在75%的基准数据上获得最大团,剩下不能得到最大团的数据实际上也可以获得接近于最大团的近似最大团,但算法的平均求解时间仅为目前最大团精确求解算法的20%左右。因此,在很多最大团非精确要求的场景中,所提算法具有极高的应用价值。

关键词:近似最大团;求解算法;邻接子图;悬挂子图;局部有限搜索
中图法分类号　TP301.6

Quick Algorithm to Find an Approximate Maximum Clique in Undirected Graph by Using Local-limited Search Strategy

ZHONG Mao-sheng[1,2],JIANG Chao[2],TAO Lan[2],HE Xiong[2] and LUO Yuan-sheng[3]
1 School of Computer & Information Engineering,Jiangxi Normal University,Nanchang 330022,China
2 School of Information Engineering,East China Jiaotong University,Nanchang 330013,China
3 Center of Network Information Management,Jiangxi University of Finance and Economics,Nanchang 330013,China

Abstract　Finding the maximum clique is a well-known NP-complete problem,and the classical algorithms of finding the maximum clique basically use the exact search strategy. Because of the complexity of the NP-complete problem,these algorithms may only be applicable to some special small scale graphs,which are difficult to applied to the complex graphs with large scale vertices and edges. In this paper,aiming at the problem of executing much redundant and ineffective search after finding a maximum clique by using these classical algorithms,this paper proposed a new algorithm based on the partition & recursive and the local-limited search strategy to find a maximum clique in an undirected graph,that is,partitioning a graph into an adjacent sub-graph and suspended sub-graph,then finding recursively the maximum clique of the adjacent sub-graph and that of parts of sub-graph of the suspended sub-graph by setting a search range control function. The experiments in a benchmark data set DIMACS show that this algorithm can find the maximum clique in most of the experimental data,can find the approximate maximum clique in other experimental data that is very close to the maximum clique,and it is much faster than these classical algorithm. Therefore,in many cases with non-precise requirements for the maximum clique,this algorithm has better application value.

Keywords　Approximate maximum clique,Finding algorithm,Adjacent sub-graph,Suspended sub-graph,Local-limited search

到稿日期:2018-11-15　返修日期:2019-05-08　本文已加入开放科学计划(OSID),请扫描上方二维码获取补充信息。
基金项目:国家自然科学基金(61877031,61462027,61562031);江西省教育厅科技计划项目(GJJ170206)
This work was supported by the National Natural Science Foundation of China (61877031,61462027,61562031) and Science and Technology Project of Jiangxi Education Departmen (GJJ170206).
通信作者:钟茂生(zhongmaosheng@sina.com)

1 引言

团(Clique)是 Luce 等为了研究矩阵分析方法在群组结构(Group Structure)中的应用时提出的一个概念[1]。当无向图中一个顶点子集的导出子图为完全子图时，该顶点子集称为团；无向图中的一个完全子图不包含在该无向图的更大完全子图中时，称该完全子图对应的顶点子集为极大团(Maximal Clique)；无向图中含有顶点数最多的团称为该图的最大团(Maximum Clique)。显然，图的最大团必然是一个极大团，而极大团则未必是最大团。

从无向图中查找最大团，是一个著名的 NP-完全问题[2]，与之相关的还有图的最大独立集问题。最大团问题在多项式时间内可以转换为许多知名的难题，如最小顶点覆盖问题、Hamilton 圈问题、Hamilton 轨问题、货郎担问题等，因此最大团问题的求解在理论和实际应用中都具有重大的意义。多年来，国内外学者对其求解算法进行了大量研究。但是，由于 NP-完全问题本身具有复杂性，现有的算法或者仅适用于某些特殊的图，或者具有指数时间复杂度。本文使用一个具有 500 个顶点、93 800 条边、边密度为 0.752、最大团为 50 个顶点的基准 DIMACS 实验数据 p_hat500-3.clq[1)]，在 Inter(R) Core(TM) i5-4200U CPU @ 1.60 GHz、8.00 GB 内存、Windows 8 的计算环境下进行的实验表明，即使用目前性能最好的 MCSb1 算法[3]，其查找时间也达到了 4 479 s，而一般的完全精确查找 MC 算法[3]的查找时间超过了 24 H，MC 改进算法 MCQ1[3]，MCQ2[3]和 MCQ3[3]的查找时间则分别为 29 413 s，20 897 s 和 24 464 s。显然，现有的团查找算法对于具有大规模顶点和边的复杂图(如社交网络)还是显得无力，难以适用。

由于团决策问题(即确定图中是否包含一个比给定值更大的团的问题)本身也是一个 NP-完全问题，因此对于任意一个无向图，要查找图中的最大团，目前的大部分算法也只有采用完全精确搜索策略，这也是导致现有算法效率不高的根本原因。然而，使用分划递归策略进行无向图最大团查找的实验(见 4.3 节中的表 1)表明，对于一个顶点度数相对均匀的无向图，算法在前面较短的时间内就已经找到了最大团，只是由于无法确定剩余的子图是否存在比当前更大的团而在剩余子图中进行多余而费力的查找，这就导致算法的大部分时间都是在进行额外的查找，从而降低了算法的效率。

基于上述原因，本文提出了一种基于局部有限搜索的近似最大团快速求解算法。该算法的主要思想是：将无向图划分为若干子图，并根据实际应用中对最大团精度的要求，设置控制函数，实现对其中部分子图的递归求解。实验表明，该算法能在大部分图上获得最大团，在少部分不能得到最大团的图上大概也可以获得少 1～2 个顶点的很接近于最大团的近似最大团，但其求解时间远短于精确算法的求解时间。因此，在最大团非精确要求的场景中，本文算法具有很好的应用价值。

本文第 2 节介绍了最大团问题的相关研究工作；第 3 节给出了子图划分相关的定义、定理和理论证明；第 4 节给出了相应的算法、算法实现；第 5 节给出了标准数据集上的实验结果；最后总结全文。

2 相关工作

鉴于最大团问题的难解性和重要性，研究人员先后提出了一系列基于枚举搜索策略的求解算法，这些算法可以分为确定性算法和启发式算法。Harary 和 Ross 首次提出采用枚举方式来求解任意图的最大团问题的确定性算法；此后，研究人员又提出了最小化状态数[4]、聚类[5]、生成极大链[6]、分枝限界法[7-14]、动态(约束)规划[15-16]、max-SAT 编码[17-18]等众多求解最大团问题的确定性算法，这些算法的最好复杂度约为 $O(2^{n/3})$[19]。从 20 世纪 80 年代末开始，研究者尝试采用启发式算法来求解最大团，主要包括贪婪算法[20]、遗传算法[21-23]、蚁群算法[24-25]等。启发式算法在时间性能上优于确定性算法，但其有时也只找到了近似最大团。近年来，随着并行处理技术的发展，研究者又考虑将并行处理技术引入最大团问题的求解中，以期实现搜索的并行处理[26-30]。国内关于最大团问题的求解研究包括区间图法[31]、GA 和 DNA 法[32]、MEC 方法[33]、平均度排序法[34]、分治算法[35-36]等。总体来看，使用确定性搜索算法求解最大团时，其时间复杂度为指数级，启发式搜索算法的时间复杂度低于确定性算法的时间复杂度，引入并行处理技术只是通过改进运行模式来加快求解速度。

3 定义与定理

给定无向图 $G=(V,E)$，其中，V 是非空集合，称为顶点集，$V=\{1,2,\cdots,n\}$；E 是 V 中元素构成的无序二元组的集合，称为边集，无向图中的任意边 e 均为顶点的无序对，通常用“( )”表示。如果顶点集 $U\subseteq V$，且对任意两个顶点 $v_i,v_j\in U$，均有 $(v_i,v_j)\in E$，则称顶点集 U 对应的子图 G_U 是 G 的完全子图。G 的完全子图 G_U 是 G 的团，当且仅当 G_U 的顶点集 U 不包含在 G 的更大完全子图中。G 的最大团是指 G 中所含顶点数最多的团。

根据上述定义，若要求解最大团，则必须首先搜索极大团，然后找出顶点数最多的极大团作为图的最大团。针对极大团搜索，本文采用分划递归策略，即将图递归地分划为子图，然后从子图中搜索极大团。为了进行子图的划分，下面引入几个新的定义。

定义 1 图 G 上顶点 v_k 的邻接顶点集 $U_{v_k}^+$，定义为图 G 上顶点 v_k 以及与 v_k 有边相连的所有顶点 v_j 构成的顶点集合，即 $U_{v_k}^+=\{v_j\mid(\forall v_j)((v_j=v_k)\vee(v_j\in V\wedge(v_k,v_j)\in E))\}$。

定义 2 图 G 上顶点 v_k 的悬挂顶点集 $U_{v_k}^-$，定义为图 G 上与顶点 v_k 没有边相连的全部顶点 $v_j(v_j\in V-U_{v_k}^+)$，以及与 v_k 有边相连，但与 $V-U_{v_k}^+$ 上的某一顶点也有边相连的全部顶点 $v_j(v_j\in U_{v_k}^+)$ 所构成的顶点集合，即 $U_{v_k}^-=\{v_j\mid(\forall v_j)(v_k,v_j)\notin E\vee(\forall v_j)(\exists v_p)(v_j\in U_{v_k}^+\wedge v_p\in V-U_{v_k}^+\wedge(v_j,v_p)\in E)\}$。

1) http://www.dcs.gla.ac.uk/~pat/jchoco/clique/dimacs

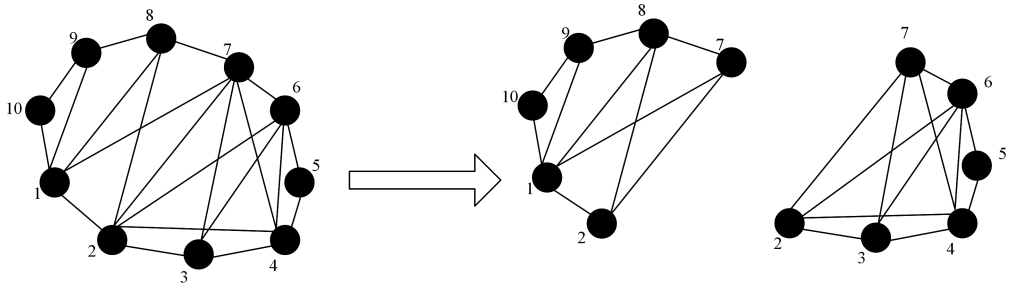
定义 3 图 G 上顶点 v_k 的划分边界顶点集 $U^*_{v_k}$，定义为 v_k 的邻接顶点集 $U^+_{v_k}$ 与 v_k 的悬挂顶点集 $U^-_{v_k}$ 的交集，即 $U^*_{v_k}=U^+_{v_k}\cap U^-_{v_k}$，此时顶点 v_k 称为 $U^*_{v_k}$ 的划分控制点。

定义 4 图 G 上顶点 v_k 的邻接子图 $G_{U^+_{v_k}}$，定义为顶点 v_k 的邻接顶点集 $U^+_{v_k}$ 在图 G 上导出生成的子图。

定义 5 图 G 上顶点 v_k 的悬挂子图 $G_{U^-_{v_k}}$，定义为顶点 v_k 的悬挂顶点集 $U^-_{v_k}$ 在图 G 上导出生成的子图。

定义 6 图 G 上顶点 v_k 的划分边界子图 $G_{U^*_{v_k}}$，定义为顶点 v_k 的划分顶点集 $U^*_{v_k}$ 在图 G 上导出生成的子图，也即邻接子图 $G_{U^+_{v_k}}$ 和悬挂子图 $G_{U^-_{v_k}}$ 的相交部分构成的子图。

下面通过例子来进一步解释上述定义。如图 1(a)中的无向图 G，以顶点 1 为示例。图 G 上顶点 1 的邻接顶点集 $U^+_1=\{1,2,7,8,9,10\}$；图 G 上顶点 1 的悬挂顶点集 $U^-_1=\{2,3,4,5,6,7\}$；图 G 上顶点 1 的划分顶点集 $U^*_1=\{2,7\}$；图 G 上顶点 1 的邻接子图 $G_{U^+_1}$ 和悬挂子图 $G_{U^-_1}$ 如图 1(b)和图 1(c)所示，其边界子图 $G_{U^*_1}$ 为顶点集{2,7}所导出的子图。



(a)无向图 G (b)邻接子图 $G_{U^+_1}$ (c)悬挂子图 $G_{U^-_1}$

图 1 无向图 G 及其邻接子图 $G_{U^+_1}$ 和悬挂子图 $G_{U^-_1}$

Fig. 1 Sketch figure of an undirected graph, adjacent sub-graph and suspended sub-graph

根据上述定义，下面给出与无向图极大团相关的几个引理和定理。

引理 1 对于图 G 上的两个顶点 v_i 和 v_j，如果 $v_i\in U^-_{v_k}-U^*_{v_k}$，$v_j\in U^+_{v_k}-U^*_{v_k}$，则 v_i 和 v_j 没有连接边。

引理 1 可根据邻接顶点集 $U^+_{v_k}$ 和悬挂顶点集 $U^-_{v_k}$ 的定义证明得出。以图 1(a)中的图 G 为例，$U^-_1-U^*_1=\{3,4,5,6\}$，$U^+_1-U^*_1=\{1,8,9,10\}$，集合{3,4,5,6}上的顶点与集合{1,8,9,10}上的顶点之间没有连接边。

引理 2 图 G 上顶点 v_k 的邻接子图 $G_{U^+_1}$ 所蕴含的每个极大团 C_i 一定包含顶点 v_k。

引理 2 可由邻接顶点集 $U^+_{v_k}$ 的定义证明得出。以图 1(a)所示的图 G 为例，顶点 1 的邻接子图 $G_{U^+_1}$ 中蕴含的 3 个极大团 $C_1(1,2,7,8)$，$C_2(1,8,9)$和 $C_3(1,9,10)$均包含了顶点 1。

引理 3 图 G 上包含顶点 v_k 的极大团 C_j，一定不包含 $V-U^+_{v_k}$ 集合中的顶点。

引理 3 可由邻接顶点集 $U^+_{v_k}$ 的定义证明得出。以图 1(a)中所示的图 G 为例，G 中包含顶点 1 的极大团有 3 个，即 $C^1_1(1,2,7,8)$，$C^1_2(1,8,9)$和 $C^1_3(1,9,10)$，图 G 上顶点 1 的邻接顶点集 $U^+_1=\{1,2,7,8,9,10\}$。因此，$V-U^+_1=\{3,4,5,6\}$，即团 C^1_1，C^1_2 或 C^1_3 中，没有一个极大团会包含顶点集 $V-U^+_1$ 中的某一个顶点。

推论 1 图 G 上包含顶点 v_k 的任何完全子图，一定不包含 $V-U^+_{v_k}$ 集合中的顶点。

推论 1 可由引理 3 推理得出。

引理 4 图 G 上包含顶点集合 $V-U^+_{v_k}$ 中某个顶点的极大团一定不包含 $V-U^-_{v_k}$ 集合中的顶点。

引理 4 可由邻接顶点集 $U^+_{v_k}$ 和悬挂顶点集 $U^-_{v_k}$ 的定义证明得出。以图 1(a)中所示的图 G 为例，G 中 $U^+_1=\{1,2,7,8,9,10\}$，$V-U^+_1=\{3,4,5,6\}$，$U^-_1=\{2,3,4,5,6,7\}$，$V-U^-_1=\{1,8,9,10\}$。包含顶点集{3,4,5,6}中部分顶点的团有 $C_1(2,3,4,6,7)$和 $C_2(5,6,7)$，这两个团都没有包含集合{1,8,9,10}中的任何顶点。

推论 2 图 G 上包含顶点集合 $V-U^+_{v_k}$ 中某个顶点的完全子图一定不包含 $V-U^-_{v_k}$ 集合中的顶点。

推论 2 可由引理 4 推理得出。

定理 1 以某个顶点 v_k 作为划分控制点，将图 G 划分为邻接子图 $G_{U^+_1}$ 和悬挂子图 $G_{U^-_1}$，令 $C^{v_k}=\{C_i\mid C_i$ 为子图 $G_{U^+_1}$ 或子图 $G_{U^-_1}$ 中至少包含 $V-U^-_{v_k}$ 或 $V-U^+_{v_k}$ 中某个顶点的所有极大团$\}$，$C=\{C_j\mid C_j$ 为图 G 所蕴含的极大团$\}$，那么 $C^{v_k}=C$。

证明：(1)假设 C_i 为子图 $G_{U^+_1}$ 或子图 $G_{U^-_1}$ 所蕴含的任意一个极大团，即 $C_i\in C^{v_k}$。当 C_i 为子图 $G_{U^+_1}$ 所蕴含的极大团时，根据推论 2，C_i 对应的顶点集 U_{C_i} 一定包含顶点 v_k，且 $U_{C_i}\subseteq U^+_{v_k}$。由于 $G_{U^+_1}$ 为图 G 的子图，因此顶点集 U_{C_i} 在图 G 上导出生成一个完全子图。根据推论 1，图 G 上包含顶点 v_k 的完全子图一定不会有 $V-U^+_{v_k}$ 集合中的顶点，也即顶点集 U_{C_i} 在图 G 上导出生成的是一个极大完全子图，因此 C_i 也是图 G 中的一个极大团，故 $C_i\in C$。而当 C_i 为子图 $G_{U^-_1}$ 所蕴含的包含 $V-U^+_{v_k}$ 中某个顶点的极大团时，顶点集 $U_{C_i}\subseteq U^-_{v_k}$。由于 $G_{U^-_1}$ 为图 G 的子图，因此顶点集 U_{C_i} 在图 G 上导出生成了一个完全子图。根据推论 2，该完全子图一定不会包含 $V-U^-_{v_k}$ 中的顶点，即顶点集 U_{C_i} 在图 G 上导出生成的是一个极大完全子图，因此 C_i 也是图 G 中的一个极大团，故 $C_i\in C$。综上得到：$C^{v_k}\subseteq C$。

(2)假设 C_j 为图 G 中的任意一个极大团，即 $C_j\in C$。由于 $C_j\in C$，即 C_j 是图 G 的一个完全子图，且图 G 中没有更大的完全子图包含该子图，其对应的顶点集为 U_{C_j}。当 $v_k\in U_{C_j}$ 时，根据引理 3，U_{C_j} 一定不包含 $V-U^+_{v_k}$ 集合中的顶点，因此 U_{C_j} 只包含在顶点集 $U^+_{v_k}$ 中，即 $U_{C_j}\subseteq U^+_{v_k}$。当图 G 划分为邻接

子图 $G_{U_1^+}$ 和悬挂子图 $G_{U_{v_k}^-}$ 后，顶点集 U_{C_j} 在邻接子图 $G_{U_1^+}$ 上导出产生的子图与在原图 G 上导出产生的 C_j 是同一个子图，因此该子图也成为 $G_{U_1^+}$ 的一个极大团，即 $C_j \in C^{v_k}$。而当 $v_k \notin U_{C_j}$ 时，根据极大团的定义，U_{C_j} 中至少存在一个顶点 $v_l \in U_{C_j}$，但 $(v_k, v_l) \notin E$，即 $v_l \in V - U_{v_k}^+$，根据引理 4，顶点集 U_{C_j} 中不包含 $V - U_{v_k}^-$ 集合中的顶点，即 $U_{C_j} \subseteq U_{v_k}^-$。因此，当图 G 被划分为邻接子图 $G_{U_{v_k}^+}$ 和悬挂子图 $G_{U_1^-}$ 后，顶点集 U_{C_j} 在悬挂子图 $G_{U_1^-}$ 上导出产生的子图与在原图 G 上导出产生的 C_j 是同一个子图，故该子图也成为 $G_{U_{v_k}^-}$ 的一个极大团，即 $C_j \in C^{v_k}$。综上得到：$C \subseteq C^{v_k}$。

根据上述 $C^{v_k} \subseteq C$ 和 $C \subseteq C^{v_k}$，即可得 $C^{v_k} = C$。定理 1 成立。证毕。

定理 1 表明，如果以某个度数小于 $n-1$ 的顶点 v_k 为划分控制点，将图 G 划分为邻接子图 $G_{U_1^+}$ 和悬挂子图 $G_{U_1^-}$，那么 $G_{U_1^+}$ 和 $G_{U_1^-}$ 上所蕴含的至少包含 $V - U_{v_k}^-$ 或 $V - U_{v_k}^+$ 中某个顶点的所有极大团 C^{v_k}，与原图 G 所蕴含的所有极大团 C 是相同的。因此，为了找到图 G 中的最大团，可以从邻接子图 $G_{U_1^+}$ 和悬挂子图 $G_{U_1^-}$ 所蕴含的所有极大团中找出顶点数最多的极大团作为最大团。

4 算法设计

4.1 基本思想

根据第 3 节中的定理 1，当用某个非 $n-1$ 度的顶点 v_k 作为划分控制点，将无向图 G 划分为邻接子图 $G_{U_1^+}$ 和悬挂子图 $G_{U_{v_k}^-}$ 后，子图上所蕴含的至少包含 $V - U_{U_1^-}$ 或 $V - U_{v_k}^+$ 中某个顶点的所有极大团与原图 G 所蕴含的所有极大团是相同的。因此，若要求解出无向图 G 的（近似）最大团，可以使用分划和递归策略，并设置有限搜索区域来进行局部搜索求解。其基本思想是：如果图 G 不是完全图，则任意取 G 中的某个非 $n-1$ 度顶点 v_k 作为划分控制点，将图 G 划分为邻接子图 $G_{U_1^+}$ 和悬挂子图 $G_{U_1^-}$，然后对 $G_{U_1^+}$ 进行递归搜索以求解（近似）最大团，而对 $G_{U_1^-}$ 则设置搜索范围控制函数来进行局部有限搜索，以求解出其中的（近似）最大团。

4.2 基于局部有限搜索的近似最大团求解算法

假设用邻接矩阵 $\boldsymbol{M}$ 来表示无向图 $G\langle V, E\rangle$，根据上述思想，本文设计算法 1 来求解无向图 G 中的（近似）最大团。算法 1 中，首先扫描图 G 对应的邻接矩阵 $\boldsymbol{M}_V$ 中非对角线元素是否均为 1，如果是，则 G 为完全图，G 就是最大团；如果 G 为非完全图，则从对应的顶点集 V 中找出一个非 $n-1$ 度顶点 v_k 来将顶点集 V 划分为 $U_{v_k}^+$ 和 $U_{v_k}^-$ 两个顶点子集。然后，对顶点集 $U_{v_k}^+$ 导出的子图 $G_{U_1^+}$ 递归求最大团；而对于顶点集 $U_{v_k}^-$，选择 $V - U_{v_k}^+$ 中的顶点 v_i 将其划分为邻接顶点集 $U_{new-v_i}^+$ 和悬挂顶点集 $U_{new-v_i}^-$，对邻接顶点集 $U_{new-v_i}^+$ 导出的子图 $G_{U_{new-v_i}^+}$ 递归求最大团，而对悬挂顶点集 $U_{new-v_i}^-$ 则继续从 $V - U_{v_k}^+$ 中选择剩余顶点进行划分。循环执行对剩余悬挂顶点集的划分，直到某个时刻 $U_{new-v_i}^-$ 中的顶点个数少于或等于当前最大团为止。算法 1 通过引入搜索范围控制函数 $f(x)$ 来实现局部有限搜索。$f(x)$ 可取常量、线性函数、平方根函数、对数函数。

算法 1 无向图近似最大团局部搜索求解算法

输入：无向图 G 的邻接矩阵 $\mathbf{M}_V$，当前已搜索得到的近似最大团 c_{max}，顶点集 V

输出：无向图 G 中的近似最大团 c_{max}

```
FindMaximumClique (M_V, V, c_max)
{   //求解图 G 的近似最大团
    if (M_V 中非对角线元素均为 1)
    {
      if (|V|>|c_max|) c_max=V;
    }
    else
    {
    (1)以非 n-1 度顶点 v_k 将图 G 划分为子图 G_{U_vk^+} 和 G_{U_vk^-}，对应
       顶点子集为 U_{v_k}^+ 和 U_{v_k}^-;
    (2)if ( |U_{v_k}^+| > |c_max| )  FindMaximumClique (M_{U_vk^+}, U_{v_k}^+,
       c_max);//在子图 G_{U_vk^+} 中求解最大团
    (3)if ( |U_{v_k}^-| > |c_max| && |V-U_{v_k}^+| >0)
       {
         U_cur^- = U_{v_k}^-;
         U_part^- = V - U_{v_k}^+;
         for (int i=0;i<f(|V+U_{v_k}^+|);i++) //f(x)为搜索范围
       控制函数，用于实现局部有限搜索
         {
           v_i = Get(V-U_{v_k}^+, i);//从集合 V-U_{v_k}^+ 中取第 i 个顶点
       赋给 v_i
           if ( |U_cur^-| > |c_max| && |U_part^-| >0)
           {
             if (v_i ∈ U_cur^-)
             {
               以 v_i 为划分点，将集合 U_cur^- 划分为邻接顶点集
               U_{new-v_i}^+ 和悬挂顶点集 U_{new-v_i}^-;
            if ( |U_{new-v_i}^+| > |c_max| ) FindMaximumClique
       (M_{U_{new-v_i}^+}, U_{new-v_i}^+, c_max);
             U_part^- = U_cur^- - U_{new-v_i}^+;
             U_cur^- = U_{new-v_i}^-;
           }
         }
         else break;
    }
  }
}
Return c_max
}
```

5 实验及结果

为了验证算法 1 的性能，用 Java 语言实现了相应的算法程序，并将其与文献[3]中列出的所有算法进行对比实验（文献[3]对当前主要的精确求解最大团的算法进行了实验对比研究）。实验采用的数据为基准 DIMACS 实验数据，文献[3]

中的所有对比程序的下载地址为 http://www.dcs.gla.ac.uk/~pat/maxClique/distribution/，对比程序由 Java 语言编写，实验环境均为 Inter(R) Core(TM) i5-4200U CPU @ 1.60GHz、8.00GB 内存、Windows 8 的操作系统，算法运行之前将所有顶点按度的值从大到小进行排序。

算法 FindMaximumClique 通过引入搜索范围控制函数 $f(x)$来实现局部有限搜索(x 为对应子图的顶点个数)。当使用控制函数 $f(x)=x$ 时，算法 FindMaximumClique 实际上执行的是完全精确搜索。因此，我们首先用该算法来找出每个图中第一次获得当前最大团的实际使用时间以及最后完全搜索整个图所用的时间。算法在部分 DIMACS 数据集上执行的结果如表 1 所列。表 1 中，第一列到第三列分别表示 DIMACS 数据名称、图的定点数、该图的最大团数。第四列表示算法在执行搜索后得到的当前最大团及所用时间(单位为 s)。以“16/0.006;17/0.008;18/0.011;19/0.020;20/1.120;21/8.819”为例，它表示在 brock200_1 数据搜索时，搜索到大小为 16 的团时使用了 0.006 s，搜索到大小为 17 的团时使用了 0.008 s，而搜索得到大小为 21 的团时使用了 8.819 s。第五列表示算法执行结束得到的最大团和运行总时间(单位为 s)，如 brock200_1 数据中，算法执行结束时得到的最大团为 21，算法运行的总时间为 154.574 s。第六列表示算法从搜索到当前图的最大团开始直到算法结束运行所需的额外搜索时间(单位为 s)。以 brock200_1 数据为例，算法第一次得到 21 的最大团时使用了 8.819 s，而总时间为 154.574 s，因此其额外搜索时间为 154.574－8.819＝145.755 s。第六列为时间利用率，是指算法搜索到当前图的最大团需要的时间与算法运行结束所需的搜索时间之比，如 brock200_1 数据中，其时间利用率为(8.819/154.574)×100%＝5.71%。

表 1 算法 FindMaximumClique 进行完全搜索获得当前时刻最大团的用时及算法运行结束的用时

Table 1 Time of obtaining current maximum clique and end time of running algorithm on FindMaximumClique

数据名称	顶点数	最大团	算法执行过程中搜索得到的当前最大团及相应的运行时间	最大团/算法运行结束用时	额外搜索时间	时间利用率/%
brock200_1	200	21	16/0.006;17/0.008;18/0.011;19/0.020;20/1.120;21/8.819	21/154.574	145.76	5.71
brock200_2	200	12	7/0.002;8/0.004;9/0.006;10/0.008;12/0.013	12/0.183	0.170	7.10
brock200_3	200	15	10/0.002;12/0.005;13/0.011;14/0.424;15/1.628	15/2.017	0.389	80.71
brock200_4	200	17	13/0.002;14/0.005;15/0.021;16/0.060;17/2.758	17/7.702	4.944	35.81
hamming6-2	64	32	32/0.003	32/0.021	0.018	14.29
hamming8-4	256	16	16/0.002	16/15.148	15.146	0.01
johnson16-2-4	120	8	8/0.002	8/5.507	5.505	0.04
johnson8-4-4	70	14	14/0.002	14/0.042	0.040	4.76
keller4	171	11	8/0.002;9/0.005;10/0.008;11/0.012	11/2.936	2.924	0.41
MANN_a9	45	16	16/0.002	16/0.341	0.339	0.59
p_hat1000-1	1000	10	7/0.004;8/0.007;9/0.010;10/0.014	10/9.723	9.709	0.14
p_hat300-1	300	8	7/0.002;8/0.006	8/0.049	0.043	12.24
p_hat300-2	300	25	23/0.003;24/0.005;25/0.016	25/8.488	8.472	0.19
p_hat500-1	500	9	6/0.003;7/0.006;8/0.009;9/0.014	9/0.455	0.441	3.08
p_hat700-1	700	11	7/0.003;8/0.005;9/0.008;10/0.290;11/0.573	11/1.735	1.162	33.03
sanr200_0.7	200	18	14/0.002;15/0.005;16/0.009;17/0.031;18/0.151	18/26.210	26.059	0.58
sanr400_0.5	400	13	10/0.002;11/0.005;12/0.008;13/3.010	13/12.902	9.892	23.33

从表 1 中的结果可以看出，对于一个顶点度数相对均匀的无向图，在对其进行完全精确搜索求解最大团时，算法在前面比较短的时间就已经得到了最大团，只是由于无法确定剩余的子图中是否存在更大的团而执行了很多额外的搜索，致使完全精确搜索的时间很长，时间利用率很低。表 1 中，时间利用率低于 10%的搜索将近占 2/3，而对于 hamming8-4 图的最大团搜索，其利用率只有 0.01%，这意味着近 99.99%的时间在对剩余子图进行额外的搜索。当算法 FindMaximumClique 中的控制函数 $f(x)$分别取 1，$0.1x$，$\log_2(x)$和 $\mathrm{sqrt}(x)$时，表示根据 $f(x)$的值从 $V-U_{v_k}^{+}$的部分顶点生成的邻接子图中进行局部搜索，因此加快了算法的执行速度。

算法程序在 DIMACS 数据集上运行的实验结果如表 2 所列，表中的算法 FindMaximumClique 列对应的数据格式为“C/T”，其中 C 表示所找到的(近似)最大团大小，T 表示需要的时间。

从表 2 中可以看出：

(1)当控制函数 $f(x)=1$ 时，运行是最快的，找到的大部分结果也是图中实际的最大团大小，而部分近似结果虽非实际最大团，但比实际最大团少 2 个顶点左右；

(2)当控制函数 $f(x)=0.1x$ 时，在 $f(x)=1$ 的基础上，又在 brock200_2 和 p_hat500-1 上找到了最大团，在 brock200_1 和 p_hat700-1 上找到顶点数更多的近似最大团，但整体的搜索时间比 $f(x)=1$ 长；

(3)当控制函数 $f(x)=\log2(x)$时，在 $f(x)=0.1x$ 的基础上，又在 brock200_1 和 sanr200_0.7 上找到了最大团，在 brock200_3，p_hat300-3 和 p_hat500-3 上找到顶点数更多的近似最大团，但整体搜索时间在 $f(x)=0.1x$ 的基础上快速增加；

(4)控制函数 $f(x)=\mathrm{sqrt}(x)$时，在每个数据上找到的近似最大团与 $f(x)=\log2(x)$时相同，但搜索时间再次增加。

总体来看，算法 FindMaximumClique 可根据实际应用中对最大团精度的要求，通过设置搜索范围控制函数 $f(x)$实现对无向图的局部有限搜索，但即使设置 $f(x)=1$(即每次只对悬挂子图中的一个子图进行搜索)，实际上也可以获得很接近最大团的近似最大团，并且极大地缩短了搜索求解的时间。

表 2 几种主要的最大团求解算法与本文算法在 DIMACS 数据集上的实验对比

Table 2 Comparison of several main algorithms for maximum clique with proposed algorithm on DIMACS dataset

数据名称	顶点数	最大团	*MC*	*MC*0	*MCQ*1	*MCSa*1	*MCSb*1	*BBMC*1	FindMaximumClique			
									$f(x)=1$	$f(x)=0.1x$	$f(x)=\log_2(x)$	$f(x)=\mathrm{sqrt}(x)$
brock200_1.clq	200	21	**305.483**	127.977	5.454	4.173	2.798	5.876	19/**0.033**	20/0.115	21/1.585	21/1.595
brock200_2.clq	200	12	**0.188**	0.094	0.016	0.032	0.032	0.031	9/**0.002**	12/0.005	12/0.006	12/0.009
brock200_3.clq	200	15	**2.375**	1.158	0.112	0.111	0.109	0.142	13/**0.002**	13/0.016	14/0.040	14/0.062
brock200_4.clq	200	17	**8.516**	3.938	0.439	0.423	0.361	0.579	16/**0.003**	16/0.020	16/0.108	16/0.156
c-fat200-1.clq	200	12	0	0	0	0	0	0	12/0.001	12/0.001	12/0.001	12/0.000
c-fat200-2.clq	200	24	0	0	0	0	0	0	24/0.000	24/0.000	24/0.000	24/0.000
c-fat200-5.clq	200	58	**0.109**	0.032	0	0	0	0	58/0.000	58/0.000	58/0.000	58/0.000
c-fat500-1.clq	500	14	0	0	0	0	0	0	14/0.001	14/0.001	14/0.001	14/0.000
c-fat500-10.clq	500	126	**3590.0**	1329.02	0.016	0.016	0.094	0.015	126/**0.001**	126/0.001	126/0.001	126/0.001
c-fat500-2.clq	500	26	0	**0.016**	0	0	0	0	26/0.002	26/0.001	26/0.001	26/0.000
c-fat500-5.clq	500	64	**0.516**	0.501	0.501	0	0.016	0	64/0.001	64/0.001	64/0.001	64/0.000
hamming6-2.clq	64	32	**0.218**	0.078	0	0	0	0	32/0.001	32/0.001	32/0.005	32/0.016
hamming6-4.clq	64	4	0	0	0	0	0	0	4/0.000	4/0.000	4/0.000	4/0.000
hamming8-4.clq	256	16	**32.642**	15.221	0.328	0.375	0.375	0.469	16/**0.005**	16/0.074	16/0.128	16/0.219
johnson16-2-4.clq	120	8	**4.594**	1.579	0.251	0.297	0.312	0.734	8/**0.001**	8/0.003	8/0.038	8/0.032
johnson8-2-4.clq	28	4	0	0	0	0	0	0	4/0.000	4/0.000	4/0.000	4/0.000
johnson8-4-4.clq	70	14	**0.078**	0.031	0	0	0	0	14/0.000	14/0.001	14/0.004	14/0.000
keller4.clq	171	11	**3.767**	1.563	0.064	0.063	0.063	0.093	11/**0.002**	11/0.016	11/0.055	11/0.078
MANN_a9.clq	45	16	**0.875**	0.312	0	0	0	0	16/0.015	16/0.015	16/0.008	16/0.016
p_hat1000-1.clq	1000	10	**6.609**	3.391	1.329	1.328	1.391	2.424	9/**0.028**	10/0.236	10/0.052	10/0.094
p_hat300-1.clq	300	8	0.032	0.015	0.016	0	0.016	0	7/0.000	8/0.004	8/0.003	8/0.000
p_hat300-2.clq	300	25	**91.458**	39.174	0.142	0.063	0.094	0.094	25/**0.010**	25/0.038	25/0.126	25/0.171
p_hat300-3.clq	300	36	**>24 h**	>24 h	54.146	11.331	17.019	14.971	34/**0.158**	34/2.417	35/48.375	35/76.218
p_hat500-1.clq	500	9	**0.312**	0.157	0.063	0.079	0.062	0.111	8/**0.007**	9/0.019	9/0.013	9/0.033
p_hat500-2.clq	500	36	**>24 h**	>24 h	15.626	2.626	4.097	3.439	34/**0.052**	34/0.999	35/2.488	35/4.313
p_hat500-3.clq	500	50	**>24 h**	>24 h	**29413.49**	**1287.73**	**4479.86**	**1762.83**	**48/2.200**	48/318.128	49/5395.545	49/16502.7
p_hat700-1.clq	700	11	**1.141**	0.564	0.234	0.234	0.234	0.344	8/**0.013**	9/0.068	9/0.013	9/0.046
sanr200_0.7.clq	200	18	**39.174**	17.954	1.204	1.111	0.939	1.563	17/**0.004**	17/0.035	18/0.352	18/0.454
sanr400_0.5.clq	400	13	**15.112**	7.485	2.048	2.126	1.735	3.095	12/**0.006**	12/0.088	12/0.063	12/0.157

结束语 无向图最大团求解是一个著名的 NP-完全问题，经典求解算法基本上都采用完全精确搜索策略。但由于 NP-完全问题本身具有复杂性，这些算法或许仅适用于某些特殊的小规模图，对于具有大规模顶点和边的复杂图（如社交网络）还是显得无力，难以适用。本文针对完全精确搜索策略下的无向图最大团求解算法的大部分时间都是在对图进行多余而无效的查找的问题，采用分划递归策略，通过设置搜索范围控制函数来对划分后的子图进行部分搜索，所提算法可根据实际应用中对最大团的精度要求来选择相应的搜索范围控制函数。实验表明，按照本文的分划递归策略，即使设置搜索范围控制函数 $f(x)=1$（即每次只对悬挂子图中的一个子图进行搜索），也可以获得很接近于最大团的近似最大团，并且极大地缩短了求解时间。因此，在最大团非精确要求的场景中，所提算法具有很好的应用价值。

参 考 文 献

[1] LUCE R D, PERRY A D. A method of matrix analysis of group structure [J]. Psychometrika, 1949, 14(2): 95-116.

[2] GAREY M R, JOHNSON D S. Computers and Intractability: A guide to the Theory of NP-Completeness [M]// Computers and Intractability: A Guide to the Theory of NP-Completeness. New York: Freeman, 1979.

[3] PATRICK P. Exact Algorithms for Maximum Clique: A Computational Study[J]. Algorithms, 2012, 5(4): 545-587.

[4] PAULL M C, UNGER S H. Minimizing the Number of States in Incompletely Specified Sequential Switching Functions[J]. IRE Transactions on Electronic Computers, 1959, EC-8(3): 356-367.

[5] Bonner R E. On some clustering techniques[J]. IBM Journal of Research and Development, 1964, 8(1): 22-32.

[6] BEDNAREK A R, TAULBEE O E. On maximal chains [J]. Revue Roumaine des Mathematiques Pures et Appliquees, 1966, 11(1): 23-25.

[7] PARDOLOS P M, RODGERS G P. A branch and bound algorithm for the maximum clique problem[J]. Computer & OR, 1992, 19: 363-375.

[8] TOMITA E, SEKI T. An efficient branch-and-bound algorithm for finding a maximum clique and computational experiments [M]// Discrete Mathematics and Theoretical Computer Science. Berlin: Springer, 2003: 278-289.

[9] KONC J. An improved branch and bound algorithm for the maximum clique problem[J]. MATCH Commun. in Mathematical and in Computer Chemistry, 2007, 58(3): 569-590.

[10] SEGUNDO P S, TAPIA C. A new Implicit Branching Strategy for Exact Maximum Clique[C]// XXII IEEE International Conference onTools with Artificial Intelligence (ICTAI). IEEE, 2010: 352-357.

[11] TOMITA E, SUTANI Y, HIGASHI T, et al. A Simple and Faster Branch-and-Bound Algorithm for Finding Maximum Clique [M]// WALCOM: Algorithms and Computation. Berlin: Springer, 2010: 191-203.

[12] PATTABIRAMAN B, PATWARY M M A, GEBREMEDHIN A H, et al. Fast Algorithms for the Maximum Clique Problem on Massive Graphs with Applications to Overlapping Community Detection[J]. Internet Mathematics, 2015, 11(4/5): 421-448.

[13] MIKHAIL B, BORIS G, EVGENY M, et al. Improvements to MCS algorithm for the maximum clique problem[J]. Journal of Combinatorial Optimization, 2014, 27(2): 397-416.

[14] MCCREESH C, PROSSER P. Reducing the Branching in a Branch and Bound Algorithm for the Maximum Clique Problem [C] // International Conference on Principles and Practice of Constraint Programming. Cham: Springer, 2014.

[15] ÖSTERGÅRD, PATRIC R J. A fast algorithm for the maximum clique problem[M]. Elsevier Science Publishers B. V., 2002.

[16] RÉGIN J C. Using Constraint Programming to Solve the Maximum Clique Problem[C] // International Conference on Principles & Practice of Constraint Programming. Springer Berlin Heidelberg, 2003.

[17] LI C M, QUAN Z. An Efficient Branch-and-Bound Algorithm Based on MaxSAT for the Maximum Clique Problem[C] // Twenty-fourth Aaai Conference on Artificial Intelligence. DBLP, 2010.

[18] LI C M, FANG Z, XU K. Combining MaxSAT Reasoning and Incremental Upper Bound for the Maximum Clique Problem [C]// 2013 IEEE 25th International Conference on Tools with Artificial Intelligence. IEEE, 2014.

[19] BOMZE I M, BUDINICH M, PARDALOS P M, et al. The maximum clique problem [M] // Handbook of Combinatorial Optimization 4. Kluwer Academic Publishers, 1999: 1-74.

[20] FEO T A, RESENDE M G C, SMITH S H. A Greedy Randomized Adaptive Search Procedure for Maximum Independent Set [J]. Operations Research, 1994, 42(5): 860-878.

[21] BUI T N, EPPLEY P H. A Hybrid Genetic Algorithm for the Maximum Clique Problem[C] // International Conference on Genetic Algorithms. DBLP, 1995.

[22] MARCHIORI E. Genetic, Iterated and Multistart Local Search for the Maximum Clique Problem[C] // Applications of Evolutionary Computing, EvoWorkshops 2002. Berlin: Springer, 2002.

[23] ZHANG Q, SUN J, TSANG E. An Evolutionary Algorithm With Guided Mutation for the Maximum Clique Problem[J]. IEEE Transactions on Evolutionary Computation, 2005, 9(2): 192-200.

[24] BUI T N, JR J R R. Finding Maximum Cliques with Distributed Ants[J]. Gecco Lecture Notes in Computer Science, 2004, 3102: 24-35.

[25] YIN H, SONG H. The Research of Ant Colony Optimization Algorithm for Maximum Clique Problem[J]. Journal of University of South China(Science and Technology), 2017, 31(3): 81-85.

[26] SEGUNDO P S, DIEGO R L, JIMÉNEZ A. An exact bit-parallel algorithm for the maximum clique problem[J]. Computers & Operations Research, 2011, 38: 571-581.

[27] XIANG J, GUO C, ABOULNAGA A. Scalable maximum clique computation using MapReduce[C] // 2013 IEEE 29th International Conference on Data Engineering (ICDE). IEEE, 2013.

[28] MCCREESH C, PROSSER P. Multi-Threading a State-of-the-Art Maximum Clique Algorithm[J]. Algorithms, 2013, 6(4): 618-635.

[29] GU J H, HUO S J, WU J Y, et al. Parallel multi-layer graph partitioning method for solving maximum clique problems[J]. Journal of Computer Applications, 2018, 38(12): 3425-3432.

[30] MCCREESH C, PROSSER P. The Shape of the Search Tree for the Maximum Clique Problem, and the Implications for Parallel Branch and Bound[J]. Acm Transactions on Parallel Computing, 2014, 2(1): 1-27.

[31] ZHONG S, XIE L. An Algorithm Computing the Maximum Clique in a Graph[J]. Journal Of Software, 1999, 10(3): 288-292.

[32] LI Y. Genetic algorithm in DNA computing: A solution to the maximal clique problem[J]. Chinese Science Bulletin, 2004, 49(9): 967.

[33] ZHOU X D, SUN C Y, LI W J. Solving Maximum Clique Problem by MEC[C] // Advance of China Artificial Intelligence (2003). Beijing: Beijing University of Post and Telecommunication, 2003.

[34] JIA X F, GUO T H, XU X X. A New Algotithm for the Maximum Clique Problem[J]. Journal of North University of China (Natural Science Edition), 2006, 27(2): 180-182.

[35] HE K, ZOU S H, ZHOU J R. Enumerating maximal cliques in large sparse graphs[J]. Journal of Huazhong & Technology (Natural Science Edition), 2017, 45(12): 1-6.

[36] HUANG F, NING A B, LIU Z M, et al. Measure and Conquer Approach for the Maximum Vertex Weightet Clique Problem [J]. Mathematical Theory and Applications, 2017, 37(2): 97-104.

ZHONG Mao-sheng, Ph. D, professor, master supervisor. His research interests include graph theory, algorithm design, social network and natural language processing.