第 36 卷 第 6 期 计 算 机 工 程 2010 年 3 月
Vol.36 No.6 Computer Engineering March 2010
·软件技术与数据库· 文章编号：1000—3428(2010)06—0039—03 文献标识码：A 中图分类号：TP301.6

低度图的最大团求解算法

王青松，范铁生
(辽宁大学信息学院，沈阳 110036)

摘 要：在图的最大团问题中，当图的顶点数不大于阈值 m 时，很容易求解其最大团问题，求解算法的时间复杂度为 $O(d)$。给出一种求解低度图的最大团的确定性算法。该算法通过对图按顶点逐步分解实现分别计算，较好地解决低度图的最大团问题。算法时间复杂度为 $O(d \cdot n^3)$。其中，n 表示图的顶点数，图中顶点的最大度小于 m 或者图可以通过逐个删除度小于 m 的顶点而使所有顶点的度都小于 m。
关键词：最大团问题；图论；图论算法；NP 问题；独立集

Solution Algorithm for Maximum Clique in Low-degree Graphs

WANG Qing-song, FAN Tie-sheng
(Information Institute of Liaoning University, Shenyang 110036)

【Abstract】In the Maximum Clique Problem(MCP), setting m as a threshold means that it is easy to compute MCP of a graph whose vertices are not greater than m, and the time complexity is $O(d)$. An exact algorithm to compute MCP in low-degree graphs is presented. The algorithm solves MCP successfully by dividing the vertices of the graph gradually and computing MCP separately. The algorithm is easy to be realized, and the time complexity is $O(d \cdot n^3)$. n represents graph vertices, the maximum degree of vertex in graph is lower than m, or the graph can make all the vertex degree lower than m by gradually deleting the vertices which are lower than m.
【Key words】Maximum Clique Problem(MCP); graph theory; graph theory algorithms; NP problem; independent set

1 概述

求解图的最大团问题是一个著名的 NP-完全问题。与之等价的问题是图的最大独立集问题。最大团问题在多项式时间内可以转化为许多知名的难题，如最小顶覆盖问题、Hamilton 圈问题、货郎担问题，因此，在理论和应用上都具有重大意义[1]。

求解最大团的难度在于问题规模，如果图的顶点数很少，那么用现有的一些确定性算法很容易求解。但是，图的顶点增多时，现有的算法不能有效解决这些 NP-完全问题，所以，指数时间的算法往往使许多稍微复杂的 NP 问题以现有计算机速度根本无法解[2]。虽然文献[3]提出用 DNA 计算解决最大独立集问题，但从计算理论上还未实现根本性的突破，因为目前采用的 DNA 计算模型基本上都采用以空间换时间的思路，这种方法仍然无法无法应对指数性的组合爆炸。文献[4]采用启发式算法解决最大团问题，但启发式算法的唯一缺点是不一定能找到最优值，有时只能找到近优值。

解决独立集问题最重要的一步是计算低度图，因此，低度图的独立集问题成了 NP 问题精确算法结果改进的瓶颈，受到特别关注和重点研究[5]。但是目前像文献[5]所描述的解决低度图独立集问题算法的时间复杂度为 $O(k^n)(1<k\leqslant 2)$，而且该算法仅对 3 度图特别有效。求解最大团确定性算法最早由文献[6]提出，但是算法的时间复杂度是指数阶的。文献[1]提出了一种确定性求解最大团算法，在实现时需要把简单图转化为某个相关的区间图，要求每个顶点对应的区间数量不能太多，而且只是在满足一定条件下才可以在多项式的时间内求解。

假设求解不大于 m 个顶点的图的最大团的时间是快的，相应求解算法的时间复杂度为 $O(d)$。本文通过对图按顶点逐步分解，分解后的各个子图的顶点数都不超过 m，再对各个子图分别计算来求解图的最大团。本文给出一种求解图的最大团的确定性算法，算法实现简单，时间复杂度是 $O(d \cdot n^3)$。其中，n 是图的顶点数，G 中顶点的最大度小于 m 或者可以通过逐个删除度小于 m 的点而使所有点的度都小于 m。所以，本算法以较低的时间复杂度较好地解决了低度图的最大团问题，而且对图的限制较宽松，m 越大，算法适用的范围越大，而且本算法也适合一些满足条件的高度图。因为对图 G 有限制，所以算法的缺点是不能有效地求解所有图的最大团问题。总的说来，本算法特别适用于顶点特别多的低度图的最大团求解。

2 最大团问题

在图论中，团(或称集团)是指图的顶点集的一个子集，使得其导出子图为完全图，如果一个团不是任何其他团的子集，则称该团为极大团(maximal clique)。一个图中含有顶点数最多的团称为该图的最大团(maximum clique)[1]。

为了叙述方便，假设待求解的图用 G 表示，G 是无向简单图，其顶点数用 n 表示，最大团的顶点集合用 V 表示，m 为阈值，当图的顶点数不超过 m 时，很容易求解图的最大团，相应求解算法的时间复杂度设为 $O(d)$。

根据最大团的定义可知，如果 V 中含有 G 的某个顶点 v，那么 V 中其他点一定是 v 的邻接点。所以，将 G 中各个顶点

基金项目：辽宁大学“211”三期工程基金资助项目
作者简介：王青松(1974－)，男，讲师、硕士，主研方向：图论及应用；范铁生，副教授
收稿日期：2009-08-25 **E-mail：**wqsteacher@sohu.com

及邻接点依次从 G 中分离出来，求得分离后各个子图的最大团，进而求得 G 的最大团的方法是可行的。如果将 v 及 v 的邻接点从 G 中分解出来形成 G 的子图 G_v，且 G_v 的顶点数不超过 m，那么求解 G_v 的最大团就很容易。

所以，当 G 的顶点数 n 大于 m 时，可以从 G 中寻找度数小于 m 的点 v 进行分解，形成 G 的子图 G_v，G_v 由 v 及 v 的邻接点在 G 中导出。然后从 G 中删除 v，因为 v 的度小于 m，所以 G_v 的顶点数不会超过 m，求得 G_v 的最大团是快速的。如果 v 属于 V，那么 G_v 的最大团就是 G 的最大团。如果 v 不属于 V，那么删除 v 不会影响 G 的最大团求解。删除 v 使 G 的规模缩小且减少了进一步求解最大团重复计算的次数。重复这样的步骤，如果每一步都能从 G 中导出顶点数不超过 m 的子图且使 G 的顶点数减少一个，那么在有限的步骤内就很容易解决 G 的最大团问题。当然要比较求得的各个子图的最大团，顶点数最多的最大团一定是 G 的最大团。

任何可以用计算机求解的问题所需的计算时间都与其规模有关。问题规模越小，所需的计算时间往往越少，从而也较容易处理。要想直接解决一个较大问题，有时是相当困难的，特别是对 NP 问题的解决。求解最大团的确定性算法的时间复杂度一般都是 $O(n\times 2^n)$，所以，当 n 特别大时，最大团问题很难解决[7]。但是，如果将一个顶点数特别多的图分解成 $k(1\leqslant k\leqslant n\text{-}m+1)$个子图，而每个子图的顶点数都不超过 m，那么求解各个子图的最大团是容易的，通过对各个子图求解的最大团进行比较，最终求得图的最大团，这样通过“分治”的策略就很容易地解决了一个大而难的问题。

3 最大团的求解算法及分析

最大团的顶点集 V 初始为空，m 是阈值，b 是算法结束的标志，其初值为 false。求解 G 的最大团问题的算法描述如下：

(1)如果$|G|>m$，即 G 的顶点数大于 m，则转到步骤(3)；

(2)将 b 置为 true，对 G 直接求得最大团，其顶点集合用 D 表示，转到步骤(8)；

(3)从 G 中任选一个度数小于 m 的顶点 v，转到步骤(4)，如果找不到符合条件的 v，则转到步骤(2)(此时对 G 无法进行有效的分解)；

(4)导出 G 的子图 G_v，G_v 是由 v 及 v 的邻接点在 G 中所导出；

(5)在 G 中删除 v；

(6)如果$|G_v|\leqslant|V|$(即生成的 G_v 的顶点数不大于当前最大团集合 V 的元素个数)，那么转到步骤(1)；

(7)求解 G_v 的最大团，G_v 的最大团顶点集合用 D 表示；

(8)如果$|D|>|V|$，则用 D 中数据替换 V 中数据；

(9)如果 b 等于 true，算法结束，否则，转到步骤(1)。

算法结束的条件是 b 为 true，b 被赋值为 true 有 2 种情况：(1)在算法第(1)步时$|G|\leqslant m$，在第(2)步对 G 已经直接求解最大团，无须继续分解，算法可正常结束，V 便是图 G 的一个最大团的顶点集合，此时算法是高效的。算法每一步都要选择一个度数小于 m 的顶点 v，求得含 v 的最大团后，在 G 中会删除 v，所以，要求 G 中顶点的最大度小于 m 或者可以通过逐个删除度小于 m 的点，从而使所有顶点的度都小于 m。(2)通过算法的第(3)步可知，如果$|G|>m$ 且所有节点的度数都超过 m，那么不对 G 实施分解(因为分解后的子图的顶点数仍会超过 m)，而是直接求解 G，此时算法是低效的。

通过算法的描述可知，如果一个图的顶点的最大度小于 m 或者可以通过逐个删除图中度小于 m 的顶点而使所有顶点的度都小于 m，那么算法每次都能选择一个度数小于 m 的顶点，在 n 大于 m 的情况下，算法对 G 进行 $n\text{-}m$ 次分解，共产生 G 的 $n\text{-}m+1$ 个子图。因为每次选择的顶点的度数都小于 m，所以每个子图的顶点数都不会超过 m，每次求解的最大团的时间复杂度是 $O(d)$。算法所涉及的其他操作都是容易实现的，如果图用邻接矩阵存储，那么在算法中寻找度数小于 m 的顶点 v 以及导出子图 G_v 等操作都需要对图的邻接矩阵进行遍历，时间复杂度是 $O(n^2)$[8]，所以，算法总的时间复杂度是 $O(d\cdot n^3)$。这个时间复杂度打破了原有问题的 NP-Hard 结构，很好地解决了一些图，特别是低度图的最大团问题。

本文算法的效率是显而易见的。举例，如果图 G 有 100 个顶点，m 为 20，求解不大于 20 个顶点的图的最大团的时间复杂度设为 $O(d)$。假设 G 中节点最大度数小于 20 或者 G 可以通过逐个删除度小于 20 的点而使所有点的度都小于 20。用传统确定性算法进行求解的时间是指数阶的，求解 G 的最大团的计算次数约为 100×2^{100}，而用本文算法的计算次数约为 $100\times100\times100\times d\approx100\times2^{14}\times d$，显然，本文算法的求解效率较高。

4 计算过程示例

求解图 1 的最大团，设 m 为 4，即求解不大于 4 个顶点的图的最大团问题是快速的，当然实际中 m 可以更大一些。最大团的顶点集合 V 初始为空，即$|V|=0$。G 的顶点个数为 9，顶点的最大度为 6。因为 G 的顶点数是 m 的 2 倍多，所以可以认为直接对 G 求最大团是困难或低效的。

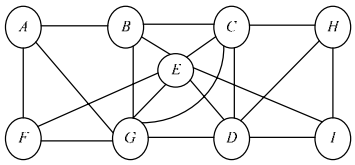

图 1　无向图 G

应用本文算法求得 G 的最大团的实现步骤如下：

(1)任选一度数小于 $m(m=4)$的节点，假定所选节点为 A。

(2)求由 A 及 A 的邻接点，即$\{A, B, F, G\}$所导出的子图 G_1(如图 2 所示)的最大团。因为 G_1 的顶点数没有超过 $m(m=4)$，所以很容易求得 G_1 的最大团的顶点集为$\{A, B, G\}$，当然 G_1 的最大团的顶点集不唯一。最大团的顶点数是 3。

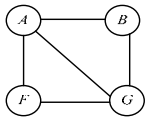

图 2　G 的子图 G_1

(3)因为 $3>|V|$，所以将 V 置成$\{A, B, G\}$。

(4)从 G 中删除 A(如图 3 所示)。

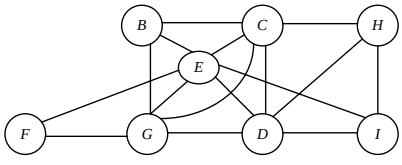

图 3　删除 A 后的图 G

重复上述步骤直至结束，上面算法可继续按照如下步骤进行：

(1)从图 3 中选中F，由$\{F,E,G\}$导出子图G_2，因为$|G_2|\leqslant|V|$，所以无须求G_2的最大团。从图 3 中删除F(如图 4 所示)，继续重复上述步骤。

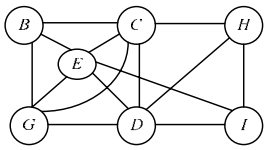

图 4　依次删除 A 和 F 后的图 G

(2)从图 4 中选择 B，由$\{B, C, E ,G\}$所导出子图的最大团顶点集合为$\{B, C, E, G\}$，所求得的最大团顶点个数为 4，超过$|V|$，所以，V 被置换为$\{B, C, E, G\}$，再删除 B，如此重复。因为每一步都能从删除相应节点后的 G 中选出一个度数小于 m 的顶点，所以算法对 G 进行 5 次分解后(此时 G 还剩 4 个节点，无需分解，可以直接求解)，共产生 G 的 6 个子图，因为每个子图的顶点数都没有超过 m，所以最终很容易求得 G 的一个最大团的顶点集合 V 为$\{B, C, E, G\}$。虽然 G 的节点个数和节点的最大度都超过了 m，但是通过这种逐步“分解”计算的方法，最终快速地求出了正确解。

5　算法应用

最大团问题在实践中有非常广泛的应用，应用领域有市场分析、方案选择、信号传输、计算机视觉、故障诊断等。利用本文所提出求解最大团的算法，在实践中解决了一个旅游公司的具体问题。

某旅游公司组织旅行团，在 n 个游客中，有游客 i 不愿与游客 j 在同一旅行团的情况，公司记录了所有这些情况，且不会将不愿在同一旅行团的 2 人安排在同一旅行团。要求计算人数最多的旅行团的人数。如果用 n 个游客作为无向图 G 的顶点，顶点 i 和 j 有边当且仅当游客 i 愿意与游客 j 在同一旅行团，问题就转化为求 G 的最大团。

通过对该旅游公司近 2 年历史数据的分析，选择 100 个规模较大的图，图的顶点数 n 一般在 100~1 500 之间，顶点的度数一般在 20~300 之间，所以，G 是一个顶点特别多而顶点的平均度数并不高的图。如果直接用传统的确定性算法求解 G 的最大团，那么一般都将需要很长的时间。所以，应用本文算法，将 G 中的各个顶点依次分解进行求解就显得十分必要，虽然 G 的规模很大，但是如果将其某个顶点 v 及其邻接点从 G 中分离出来得到其子图 G_v，那么 G_v 的规模小很多，而求解 G_v 的最大团十分容易，这样通过“分解”的策略就很容易地解决该问题。

本文在 Window XP 操作系统下，用 Java 语言实现了求解最大团的算法，在测试程序时，令 m=70，求最大团的算法用文献[7]中描述的确定性算法。通过对选择的 100 个该旅游公司历史上的图进行测试，发现直接用传统的确定性算法对图求最大团的平均时间是用本文算法求最大团的 20 倍。其中，在选择的 100 个图中有 1 个图，直接对 G 求解需要 2 个多小时，而用本文算法只需 40 多秒就能正确求解。另外，即使对那些不完全满足本文约束条件的图，用本文算法求解所需要的时间同样比直接求解的时间少得多。

6　结束语

本文通过“分治”的策略提出了一种新的求解图的最大团算法，并应用算法解决了一个具体的问题，证明本算法对于解决顶点数特别多的低度图的最大团问题是十分有效的。应用本算法时，m 的选择十分重要，因为 m 越大，算法所适用的图越多，而且可以对满足条件的高度图进行求解。当然，选择 m 的大小与应用本算法的计算机的处理能力有关。总体来说，算法对于求解符合阈值 m 条件限制的图的最大团问题是行之有效的。

参考文献

[1] 仲　盛, 谢　立. 求解图的最大团的一种算法[J]. 软件学报, 1999, 10(3): 288-292.

[2] Garey M, Johnson D. Computers and Intractability: A Guide to the Theory of NP-completeness[M]. San Francisco, USA: [s. n.], 1979.

[3] 周　康, 同小军, 刘文斌, 等. 基于闭环DNA计算的最大独立集问题的算法[J]. 计算机工程, 2008, 34(4): 40-41.

[4] 周旭东, 王丽爱, 陈　峻. 启发式算法求解最大团问题研究[J]. 计算机工程与设计, 2007, 28(18): 4329-4332.

[5] 肖鸣宇, 陈建二, 韩旭里. 低度图的点覆盖和独立集问题下界的改进[J]. 计算机学报, 2005, 28(2): 153-160.

[6] Adleman L M. Molecular Computation of Solutions to Combinatorial Optimization[J]. Science, 1994, 226(11): 1021-1024.

[7] 王晓东. 算法设计与分析[M]. 北京: 清华大学出版社, 2003.

[8] 严蔚敏, 吴伟民. 数据结构[M]. 北京: 清华大学出版社, 2006.

编辑　张正兴

(上接第 28 页)

户机端 Notebook 为 CPU: P4 2.4 GB; RAM: 768 MB; HDD: 40 GB；Windows XP Home SP2，实验结果如表 2 所示。

表 2　文件传输速率

文件大小/MB	优化前/(MB·s^{-1})	优化后/(MB·s^{-1})	提升/(%)	文件大小/MB	优化前/(MB·s^{-1})	优化后/(MB·s^{-1})	提升/(%)
0.5	0.8	0.8	0.00	32	18.9	19.7	4.23
1	1.5	1.5	0.00	64	21.8	23.1	5.96
2	4.5	4.4	-2.22	128	23.5	25.0	6.38
4	13.5	13.6	0.74	256	23.5	25.2	7.23
8	15.0	15.5	3.33	512	23.8	25.3	6.30
16	15.7	15.8	0.64	1 024	23.6	25.3	7.20

从表 2 可以看出，对于尺寸比较大的文件，有效传输率的优化效果比较明显，而对小尺寸的文件优化效果不明显。究其原因，主要是小尺寸的文件传输过程中，连接的建立、文件定位等占据时间的百分比较高。只有在大尺寸文件传输中，才能将磁盘有效数据传输率提高的结果反映到传输速度上。

4　结束语

本文基于 Zoned-Disk 技术，设计的动态调整算法在统计访问数据的基础上实现了服务器文件的动态优化。理论分析和实验数据表明，对于尺寸较大的文件(>32 MB)，使用该算法可以充分利用 Zone-Disk 技术提供的最大数据传输率，从而提高系统的 I/O 性能。

参考文献

[1] Kim S S. Zoned-RAID[J]. ACM Trans. on Storage, 2007, 3(1): 1-9.

[2] Lim S. The Dynamic Sweep Scheme Using Slack Time in the Zoned Disk[C]//Proceedings of DASFAA'06. [S. l.]: IEEE Press, 2006.

[3] Aran R. Dynamic Data Reallocation in Disk Arrays[J]. ACM Trans. on Storage, 2007, 3(2): 10-26.

编辑　陈　文