

低度图的点覆盖和独立集问题下界改进

肖鸣宇¹⁾ 陈建二^{2),3)} 韩旭里¹⁾

¹⁾(中南大学数学科学与计算技术学院 长沙 410083)

²⁾(中南大学信息科学与工程学院 长沙 410083)

³⁾(德克萨斯 A&M 大学计算机科学学院 德克萨斯州 77843-3112 美国)

摘 要 给出了一种提高低度图点覆盖和独立集问题下界的精确算法. 通过分析如何有效地减少图中的顶点来打破原问题的 NP-Hard 结构建立起搜索递推关系; 得出 3 度图的最小点覆盖问题的解决时间为 $O(1.1033^n)$, 参数化的 3 度图点覆盖问题的解决时间为 $O(kn+1.2174^k)$; 将此算法应用到 3 度图的最大独立集问题上, 可以得到运行时间为 $O(1.1033^n)$ 的解. 以上 3 结果均打破原有最佳下界.

关键词 点覆盖; 独立集; 精确算法; 参数计算

中图法分类号 TP301

Improvement on Vertex Cover and Independent Set Problem for Low-Degree Graphs

XIAO Ming-Yu¹⁾ CHEN Jian-Er^{2),3)} HAN Xu-Li¹⁾

¹⁾(School of Mathematical Sciences and Computing Technology, Central South University, Changsha 410083)

²⁾(School of Computer Science and Technologies, Central South University, Changsha 410083)

³⁾(Department of Computer Science, Texas A&M University, College Station, Texas 77843-3112 USA)

Abstract An exact algorithm to improve the upper bound of vertex cover and independent set problem for low-degree graphs is presented. The algorithm constructs recurrence relations by concentrating on how many vertices be reduced then the NP-hard structure of the problem is broken. With this idea, running time $O(1.1033^n)$ for minimum vertex cover problem on degree-3 graphs is proved. For parameterized vertex cover problem on degree-3 graphs, it also can be solved with running time $O(kn+1.2174^k)$. Using the algorithm to the maximum independent set problem on degree-3 graphs, the authors can get the running time $O(1.1033^n)$. All of the above results improve the previous best results.

Keywords vertex cover; independent set; exact algorithm; parameterized computation

1 引 言

在计算机理论研究中公认的最难的问题是 NP 到底等不等于 P, 由于这个问题一直没有得到解决, 所以到目前为止所有 NP-Hard 问题都无法证明是

否能用多项式时间来解决^[1]. 而一般指数时间的算法往往使得许多稍微复杂的 NP-Hard 问题以现有计算机速度根本无法解. 怎样改进 NP-Hard 问题精确算法的复杂度使其达到实用的程度成了计算机理论研究中最为热门的问题之一^[2].

目前 NP-Hard 精确算法研究的主要路线是通过

对最大独立集问题(Maximum Independent Set)和参数化的点覆盖问题(parameterized Vertex Cover)的研究实现的^[2]. 就最大独立集问题而言,最早由 Tarjan 和 Trojanowski^[3]在 1986 年提出来的复杂度为 $O(1.259^n)$ 的算法,随后其下界不断被刷新^[4~7],目前最佳结果为 Robson^[6]给出的复杂度为 $O(1.211^n)$ 的算法和 Beigel^[7]给出的复杂度为 $O(1.083^e)$ 算法,其中 e 为图中边的条数. 参数化的点覆盖问题则由于最近在生物计算中得到重大应用而更受关注^[8]. 从 Buss 和 Goldsmith^[9]于 1993 年首次提出复杂度为 $O(kn + 2^k k^{2k+2})$ 的算法以来,其结果也不断被改进^[10~14],现在最佳结果为 Chen^[14]给出的复杂度为 $O(kn + 1.285^k)$ 的算法.

在解决独立集和点覆盖问题中最重要的一步就是对低度图的计算,以上提到的算法中有很多就是直接通过对低度图的分析实现对任意图的计算的^[6,13,14]. 因此低度图的独立集问题和点覆盖问题成了 NP-Hard 问题精确算法结果改进的瓶颈,为此也受到特别关注和重点研究^[7,14,15]. 对 3 度图的最大独立集问题(MIS-3),目前最好的结果为 Robson^[6]给出的 $O(1.1211^n)$;而 3 度图的参数化点覆盖问题(VC-3),最佳结果为 Chen^[15]给出的 $O(kn + 1.2365^k)$ (文献[15]中提到用 Iterative Branching 技巧可以将结果再改进到 $O(kn + 1.2192^k)$,但是这需要较多的预处理,不便于实际应用).

本文将从点覆盖问题着手,考虑如何有效地减去图中的顶点来打破原问题的 NP-Hard 结构. 一旦 NP-Hard 结构被打破,剩下的问题就可以用多项式时间来解决. 这样本文的中心就在于如何用尽可能少的递推次数建立去点的递推关系. 由此思想出发,本文给出了 3 度图最小点覆盖问题复杂度为 $O(1.1033^n)$ 的算法和 3 度图参数化的点覆盖问题复杂度为 $O(kn + 1.2174^k)$ 的算法,此结果优于原点覆盖问题最佳结果 $O(kn + 1.2365^k)$ ^[15]. 本算法可以很简单地应用到 3 度图的最大独立集问题上,得到其算法复杂度为 $O(1.1033^n)$,较大改进了原最佳结果 $O(1.1211^n)$ ^[6].

2 预备知识

设 G 为一个图, n 为 G 的顶点个数, I 为 G 的一个顶点集,如果 I 中任何两个顶点都不相邻,则称 I 为 G 的一个独立集,求 G 的独立集 I 并使得 I 中的元素个数最多的问题称之为最大独立集问题. C 为

G 的另一个顶点集,如果 G 中的每条边最少有一个顶点在 C 中,则称 C 为 G 的一个点覆盖,求 G 的点覆盖 C 并使得 C 中的元素个数最少的问题称之为最小点覆盖问题. 显然,对任何图, I 是 G 的一个独立集的充要条件为 $G-I$ 是 G 的一个点覆盖. 设 k 为一个整数,求 G 中是否存在点覆盖集 C 且 C 中的元素个数不多于 k 个的问题称之为参数化的点覆盖问题,简记为 (G, k) . 特别的,若图中所有点的度数都不大于 3 度,这样对应的问题称之为 3 度图的独立集或点覆盖问题. 另外,由于习惯问题国外常把最大独立集问题简称为独立集问题,而参数化的点覆盖问题简称为点覆盖问题.

从定义可以看出,最大独立集和最小点覆盖是一对对偶的问题,其中任何一个解决了另一个也可以很快解决. 在本文中,则是从点覆盖问题着手给出最小点覆盖问题和参数化点覆盖问题的算法.

很多有效的点覆盖问题算法都是基于两个方法建立起来的:减少问题的核心阶和建立搜索树^[14]. 减少问题的核心阶就是减少图中的顶点个数,如下引理 1 中用一个多项式时间的算法把原点覆盖问题转换为关于一个顶点个数更少的图的等价点覆盖问题;对于搜索树,最常见的一种就是对一个顶点分两个分支:或把这个点放入目标点覆盖集中,或不把这个点放入目标点覆盖集中,后面这种情况则需要把这个点所有的邻点放入目标点覆盖集中. 这种分支的操作也常称为“猜点”. 根据搜索树的结构就可以建立递归关系式,例如,对于一个给定的图,参数为 k 时设其算法复杂度为 $C(k)$,若在搜索树的某一步中把原问题分为两个子问题,分别在两个子问题中参数最少可减少 a, b ,则可建立递归关系 $C(k) \leq C(k-a) + C(k-b)$. 算法的重心就是考虑怎样设计适当的搜索树来增大 a, b 的值,从而降低算法的运行时间.

引理 1^[15,16]. 对任一个 3 度图参数化的点覆盖问题 (G, k) ,存在一个复杂度为 $O(\sqrt{k^3})$ 的算法,可将其转换为另一个 3 度图参数化的点覆盖问题 (G_1, k_1) ,且 G_1 最多有 $2k_1$ 个点, $k_1 \leq k$. 这两个点覆盖问题等价.

这样在下文中考虑的参数化点覆盖问题 (G, k) 都默认图 G 最多只有 $2k$ 个顶点. 另外,还假设问题中的原始图是连续的,而且顶点个数不少于 20 个.(对于每个顶点少于 20 的图或分支,普通算法都可以不花时间计算出来).

3 算法思想和基本操作

本文算法也将建立递归关系式

$$C(m) \leq C(m-a) + C(m-b),$$

其中参数 m 表示不确定点的个数,不确定点是指在本文算法中将要猜测是否属于目标点覆盖集的点.显然,2度图可以直接找出它的最小点覆盖来,对于3度图来说,如果处理掉所有的3度点以后,则可直接算出结果.因此这里将所有3度的点就看作不确定点.在每一次分支的过程尽量多减少一些3度顶点,将递归次数降下来以此优化算法,这就是本文的思想.这种算法思想是针对最小点覆盖问题的,算法中不涉及到参数化问题中的参数 k .对于参数化的点覆盖问题,则通过论证 m 与 k 的特殊关系将递归式化为关于 k 的形式从而得到解.

在给出算法前,先介绍一下本算法中将要用到的几个基本操作:折叠1度顶点,反复折叠1度顶点,折叠2度顶点,分支4度的圈.

设 v 是 G 中的一个1度顶点, u 是 v 的唯一邻点.折叠 v 就是直接将 u 放入目标点覆盖集中(每次将一些点放入目标点覆盖集中后,同时从图中去掉这些点以及所有和这些点的连线,然后再去掉所有孤立点).反复折叠1度顶点就是指若折叠一个1度顶点后又因此操作产生了另外的1度顶点,则继续折叠这些1度顶点,直到没有新的1度顶点产生为止.

设 v 是 G 中的一个2度顶点, u 和 w 是 v 的两个邻点.如果 u 和 w 是相邻的,则折叠 v 就是将 u 和 w 两点放入目标点覆盖集中.因为 v, u 和 w 三个点构成一个三角形,显然每个三角形最少要有两个顶点在目标点覆盖集中,由于 v 只和 u, w 相邻,故可以直接把 u 和 w 两点放入目标点覆盖集中.

2度顶点折叠的另一种情况是 u 和 w 不相邻.这时折叠 v 就是建立如下一个只有 $n-2$ 个顶点的图 G' :去掉 v, u 和 w 三个点,加入一个新的顶点 v_0 ,并且将 v_0 与 u, w 的所有邻点(除 v 外)相连接(如图1).

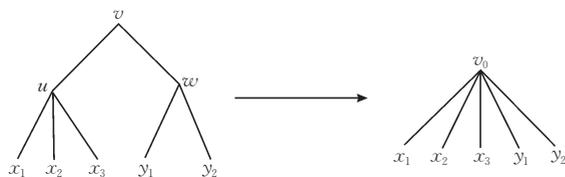


图1 2度顶点的折叠

引理2. 折叠1度顶点和2度顶点始终满足最小点覆盖.对于折叠2度顶点的第2种情况,原始

的参数化点覆盖问题 (G, k) 则等价于 $(G', k-1)$.

设 a, b, c, d 四个点组成一个4度的圈(如图2).要把这个圈中的4条边都覆盖掉,则要么 a 和 c 必须在目标点覆盖集中,要么 b 和 d 必须在目标点覆盖集中.分支4度的圈就是分别将4度圈中一对不相邻的顶点放入目标点覆盖集中如此将点覆盖问题分为两个分支的操作.

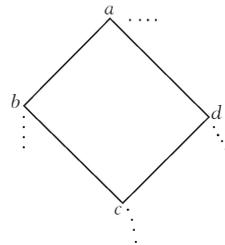


图2 4度的圈

引理3. 分支4度的圈始终满足最小点覆盖.下面分析这些基本操作的性质.

引理4. 如果图中只有一个1度顶点,则反复折叠这个1度顶点最少可以减少图中一个3度顶点.

证明. v 是1度顶点, u 是 v 的唯一邻点,若 u 是3度顶点,则折叠 v 最少可以去掉 u 这个3度顶点.如果 u 是2度顶点,则考虑 u 的另一个邻点 w (w 不同于 v).如果 w 是3度顶点,折叠 v 将会使得 w 变成2度顶点(减少一个3度顶点);否则 w 是一个2度顶点,折叠 v 将会使得 w 变成新的1度顶点,并且在这个过程中没有其它新的1度顶点产生.由于原始图中只有一个1度顶点且是有限图,所以反复折叠后一定会遇到图中的另一个奇度顶点即3度顶点.

根据上面的引理4及其证明,可以得到如下引理.

引理5. 如果图中只有两个1度顶点且存在3度顶点,则反复折叠这两个1度顶点最少可以减少图中两个3度顶点.

引理6. 折叠2度顶点和分支4度的圈不会增加图中3度点的个数.

在折叠2度顶点的第二种情况中,有可能产生4度或其它高度的顶点,在后面将会提到把高度的顶点看成是多个3度顶点的组合,按这种方法计算,这种操作仍然保证3度顶点的总个数不会增加.

4 算法及其复杂度分析

上节提到算法的中心思想是建立一个搜索关系,在每一个分支中我们减少图中的若干个点,直到

若干步以后打破原图的 NP-Hard 结构,即点覆盖问题可以用多项式时间解决.将图中所有 3 度点都去掉后,则可打破点覆盖问题的 NP-Hard 结构.本节将具体给出建立去除 3 度点的算法步骤及其复杂度分析.

为了叙述方便,这里把高度点(4 度和 4 度以上的点)看做是多个 3 度点的结合,比如一个 d 度点看成是 $(d-2)$ 个 3 度点的结合.所以当我们去掉一个 d 度点时看成是去掉了 $(d-2)$ 个 3 度点;去掉一个 3 度点或高度点的一条边时看成是去掉了 1 个 3 度点.

设 $N(v)$ 为与点 v 相邻的点集,称为 v 的邻集; $N^2(v)$ 为点 v 的二层邻集,即所有 $N(v)$ 中元素的邻居组成的点集(但不包括 $\{v\} \cup N(v)$).则对于 3 度图的最小点覆盖问题的算法步骤如图 3 给出.

1. 如果 G 中存在 1 度点或 2 度点,折叠它,否则进入步 2;
2. 如果 G 为零图(空图),直接进入步 4;否则选择图中度数最大的一个点 v ,执行 $Branch(v)$;
3. 如果 G 为零图,直接进入步 4;否则回到步 1;
4. 给出结果,退出.

图 3 算法步骤

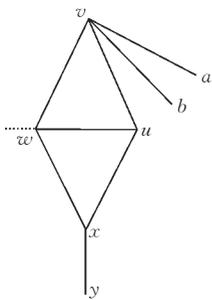
$Branch(v)$ 的描述如下:

当调用函数 $Branch(v)$ 时,图中没有任何 1 度和 2 度顶点.这也是后面算法分析中的一个重要背景条件.

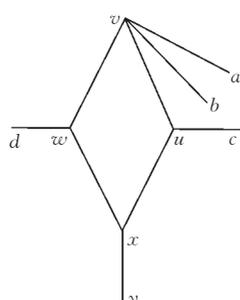
根据 v 的度数分成如下 3 种情形对 $Branch(v)$ 进行描述.

情形 1. v 是一个度数大于 4 的点.

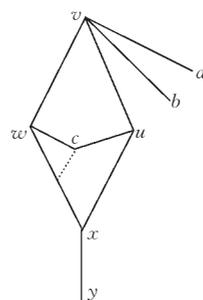
这时 $Branch(v)$ 做如下两个分支:把 v 放入目标



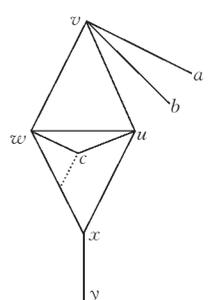
情形 2.1.1



情形 2.1.2



情形 2.1.3(a)



情形 2.1.3(b)

图 4 情形 2.1 的各种子情况

在情形 2.1.1 中, v, u, w, x 组成一个 4 度的圈,则 v, x 或 u, w 必须放入目标点覆盖集中.若把 v, x 放入目标点覆盖集中,注意到 u 和 w 相邻,故这两点中至少有一个也必须在目标点覆盖集中.考虑到把 v, x, u 三点放入目标点覆盖集不如把 $v, x,$

点覆盖集 C 中或把 $N(v)$ 放入目标点覆盖集 C 中.

设 v 的度数为 d ,则 $d \geq 5$.在第一个分支中,删掉点 v 相当于减少了 $(d-2)$ 个 3 度点,删掉 v 和 $N(v)$ 相邻的 d 条边则又减少图中 d 个 3 度点.这样这个分支总共最少可以减少 $(2d-2)$ 个 3 度点.在第二个分支中,同样最少要减少这 $(2d-2)$ 个 3 度点.

假设图中原有 m 个 3 度点,则这个分支操作始终有如下递归关系:

$$C(m) \leq 2C(m - (2d - 2)) \quad (1)$$

其中 $C(m)$ 为搜索路径的条数.

由于 $d \geq 5$,式(1)可由下式代替:

$$C(m) \leq 2C(m - 8) \quad (2)$$

定理 1. 当 v 是一个度数大于 4 的点,则 $Branch(v)$ 操作的分支递归最少满足如上关系式(2).

情形 2. v 是一个 4 度的点.

再把它分成如下两种情况.

情形 2.1 $\exists x \in N^2(v)$,且 x 有两个邻居 $u, w \in N(v)$.即 v, u, w, x 四点构成一个 4 度的圈.由于 x 和 v 没有相邻, x 最少还有一个邻点 y (y 不同于 v, u, w).再把它分为 3 种情况:

情形 2.1.1 u 或者 w 除了 v, u, w, x 四个点外没有其它的邻点,不失一般性,这里设为 u 点;

情形 2.1.2 u 有一个邻点 c, w 有一个邻点 d, c, d 不是相同点且 v, u, w, x 都不同;

情形 2.1.3 u 和 w 有除 v, x 外的另一个公共邻点 c (如图 4).

w 三点放入目标点覆盖集,这样在两个分支中都需要将 w 放入目标点覆盖集中,因此在情形 2.1.1 下, $Branch(v)$ 直接将 w 放入点覆盖集中,且最少减少 4 个 3 度顶点.

在情形 2.1.2 中, $Branch(v)$ 做两个分支,或把

v, x 放入目标点覆盖集中或把 u, w 放入目标点覆盖集中. 删去 v 则去掉了 6 个 3 度顶点, 删去 x 则最少还可以去掉 x, y 两个 3 度顶点(注意: y 可能与 a 或 b 是同一个点, 如果其度数也正好为 3, 这时删掉 v 和 x 后, y 变成 1 度的顶点, 且是整个图中唯一的 1 度顶点. 根据引理 4 在下一步折叠 1 度顶点时最少可以去掉一个 3 度顶点). 这样在第一个分支的操作中至少可以减去 8 个 3 度顶点, 它们是 $\{2v, a, b, u, w, x, y\}$ 或 $\{2v, a, b, u, w, x, y'\}$ ^①. 把 u, w 放入目标点覆盖集同样最少可以去掉 8 个 3 度顶点, 它们是 $\{2v, w, u, 2x, d, c\}$ (当 x 是 4 度点时), 或 $\{2v, w, u, x, d, c, y\}$ (x 是 3 度点, 这时把 u, w 放入目标点覆盖集会使得 x 变成 1 度点, y 则可以放入目标点覆盖集中, 当然 y 也可能与 c 或 d 是同一个点, 但不影响结果). 这样在情形 2.1.2 中, $Branch(v)$ 操作可得到与式(2)一样的递归式.

在情形 2.1.3(a) 中, 即 u 和 w 不相邻的情况. 如果 c 和 y 是同一个点, $Branch(v)$ 直接将 v, x, c 放入目标点覆盖集中; 否则 $Branch(v)$ 做两个分支, 或把 v, x 放入目标点覆盖集中或把 u, w 放入目标点覆盖集中. 每个分支都至少可以减少 8 个 3 度顶点, 它们是 $\{2v, a, b, u, w, x, y\}$ 和 $\{2v, u, w, 2c, 2x\}$ (c 和 x 有可能是 3 度点, 但操作会使得它们变成 1 度点, 由引理 4 和引理 5 可得, 它们分别可以再去掉一个 3 度顶点). 在情形 2.1.3(b) 中, 即 u 和 w 相邻的情况, $Branch(v)$ 同样做两个分支, 或把 v, x 放入目标点覆盖集中或把 u, w 放入目标点覆盖集中, 同情形 2.1.2 中的分析可得第一分支至少可减去 10 个 3 度顶点 $\{2v, a, b, 2w, 2u, x, y\}$ 或 $\{2v, a, b, 2w, 2u, x, y'\}$, 第二分支至少可以减去 10 个 3 度顶点 $\{2w, 2u, 2v, 2x, 2c\}$, 这样就得到如下递归式:

$$C(m) \leq 2C(m-10) \tag{3}$$

情形 2.2 $N(v)$ 中没有任何一对点在 $N^2(v)$ 中有公共邻居. 进一步将它分为两个小的情况:

情形 2.2.1 $\exists u \in N(v)$, 且 u 有两个邻居 $w, x \in N(v)$ (如图 5);

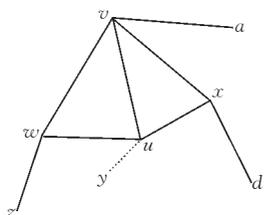


图 5 情形 2.1.1 的示意图

情形 2.2.2 对 $\forall u \in N(v)$, u 在 $N(v)$ 中不存

在两个邻居.

在情形 2.2.1 中, 如果 u 是一个 3 度顶点, 同情形 2.1.1 的分析得到 $Branch(v)$ 可以将 v 直接放入目标点覆盖集中, 此时至少可以减去 4 个 3 度顶点. 否则 u 存在一个邻居 y (y 不同于 v, x, w). 这时 $Branch(v)$ 做两个分支, 或把 v, u 放入目标点覆盖集中或把 w, x 放入目标点覆盖集中. 第一分支至少可减去 8 个 3 度顶点 $\{2v, 2u, w, x, y, a\}$ 或 $\{2v, 2u, w, x, y, a'\}$, 第二分支至少可以减 8 个 3 度顶点 $\{w, x, 2v, 2u, z, d\}$ 或 $\{w, x, 2v, 2u, z, d'\}$, 这样同样可得到递归式(2).

在情形 2.2.2 中, $Branch(v)$ 做两个分支, 或把 v 放入目标点覆盖集中或把 $N(v)$ 放入目标点覆盖集中. 显然在情形 2.2.2 中, $N^2(v)$ 最少有 4 个顶点, 这种情况如图 6 给出. 这样在第一个分支中, 至少可以减去 6 个 3 度顶点 $\{2v, a, b, c, d\}$, 在第二个分支中至少可以减去 10 个 3 度顶点 $\{2v, a, b, c, d, e, f, g, h\}$. 得到如下递归式:

$$C(m) \leq C(m-6) + C(m-10) \tag{4}$$

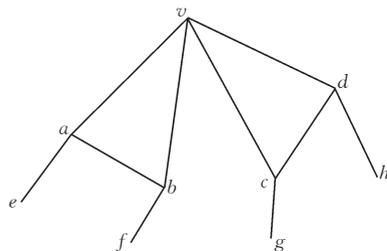


图 6 情形 2.2.2 中 $N^2(v)$ 正好只有 4 个点的情况

显然递归式(4)比式(2), (3)都要强, 这样可得到如下定理.

定理 2. 当 v 是一个度数为 4 的点, 则 $Branch(v)$ 操作的分支递归至少满足关系式(4).

情形 3. v 是一个 3 度的点.

这时 G 为一个 3 度的正则图, 同情形 2 再把它分成如下两种情况:

情形 3.1 $\exists x \in N^2(v)$, 且 x 有两个邻居 $u, w \in N(v)$. 即 v, u, w, x 这四点构成一个 4 度的圈. x 至少还有一个邻点 y (y 不同于 v, u, w). 再把它分为 3 种情况:

情形 3.1.1 u, w 相邻;

① 集合 $\{\}$ 中的点表示操作中将会去掉的 3 度顶点. $2v$ 表示从 v 这个点中去掉两个 3 度顶点; a 表示从 a 这个点中去掉一个 3 度顶点; y' 表示 y 是集合 $\{\}$ 中已经出现过的一个点, 但是操作还能从 y 中去掉一个 3 度顶点, 比如说操作将 y 变成一个 1 度点, 通过下一步对这个 1 度点的折叠操作可以去掉一个 3 度点等等.

情形 3.1.2 u, w 不相邻, 但是 u 和 w 有除 v, x 外的另一个公共邻点 c ;

情形 3.1.3 u, w 不相邻, 但 u 和 w 分别有一个邻点 b 和 c , 且 v, x, c, b 四点互不相同 (如图 7).

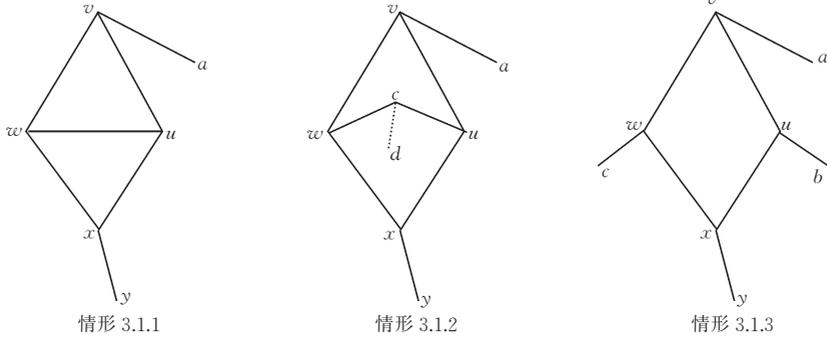


图 7 情形 3.1 的各种子情况

在情形 3.1.1 中, 如果 y 和 a 是同一个点, 则 $Branch(v)$ 将 u, w, y 直接放入目标点覆盖集中; 否则 $Branch(v)$ 做两个分支, 或把 v, x 放入目标点覆盖集中或把 $N(v) \cup N(x)$ 放入目标点覆盖集中. 这样两分支可分别去掉 6 个和 10 个 3 度顶点, 它们是 $\{v, a, u, w, x, y\}$ 和 $\{v, 3a, w, u, x, 3y\}$, 递归关系同式 (4).

在情形 3.1.2 中, 如果 y 和 c 是同一个点, 则 $Branch(v)$ 将 v, c, x 直接放入目标点覆盖集中; 否则设 d 为 c 的第三个邻点, v, x 或者 u, w 必须在目标点覆盖集中, 但是注意到若将 v, x 放入目标点覆盖集中后, 则可直接将 c 放入目标点覆盖集中, 若将 u, w 放入目标点覆盖集中后, 则可直接将 a, y, d 放入目标点覆盖集中, 这样, $Branch(v)$ 做两个分支, 或把 v, x, c 放入目标点覆盖集中或把 u, w, a, y, d 放入目标点覆盖集中. 每个分支都可以去掉 8 个以上的 3 度顶点.

在情形 3.1.3 中, 同样 v, x 或者 u, w 必须在目标点覆盖集中, 但是若将 v, x 放入目标点覆盖集中后, u, w 变成 1 度顶点, 可直接将 c, b 放入目标点覆盖集中, 若将 u, w 放入目标点覆盖集中后, v, x 变成 1 度顶点, 可直接将 a, y 放入目标点覆盖集中, 这样, $Branch(v)$ 做两个分支, 或把 v, x, c, b 放入目标点覆盖集中或把 u, w, a, y 放入目标点覆盖集中. 每个分支都可以去掉 10 个以上的 3 度顶点.

情形 3.2 $N(v)$ 中没有任何一对点在 $N^2(v)$ 中有公共邻居. 进一步将它分为 3 个小的情况:

情形 3.2.1 $\exists w \in N(v)$, 且 w 有两个邻居 $u, x \in N(v)$;

情形 3.2.2 在 $N(v)$ 中仅有一对顶点相邻, 不妨设为 u 和 w ;

情形 3.2.3 在 $N(v)$ 中没有任何顶点相邻 (如图 8).

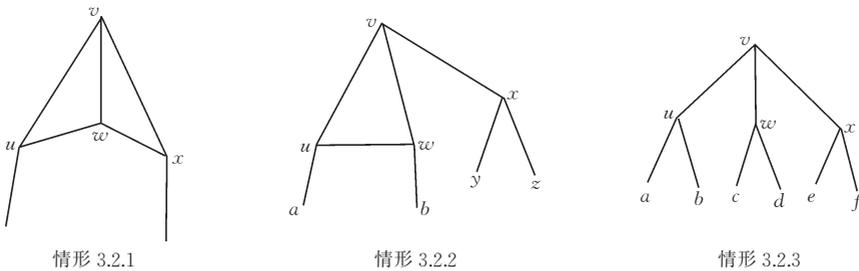


图 8 情形 3.2 的各种子情况

在情形 3.2.1 中, v, w 或者 u, x 必须在目标点覆盖集中, 注意到把 u, x 放入目标点覆盖集中的同时 v 和 w 中也要有一个点在目标点覆盖集中, 而把 u, w, x 放入目标点覆盖集中和把 v, u, x 放入目标点覆盖集中的效果是一样的, 这里就始终只取 v, u, x , 这样在两个分支中都要把 v 放入目标点覆盖集中, 则 $Branch(v)$ 直接将 v 放入目标点覆盖集中.

在情形 3.2.2 中, x 或者 $N(x)$ 必须在目标点覆盖集中, 但是若将 x 放入目标点覆盖集中后, 可直接将 u, w 放入目标点覆盖集中, 这样, $Branch(v)$ 做两个分支, 或把 x, u, w 放入目标点覆盖集中或把 v, y, z 放入目标点覆盖集中. 第一个分支最少可以去掉 $\{z, y, x, v, u, w, a, b\}$ 这 8 个 3 度顶点, 第二个分支最少可以去掉 $\{2z, 2y, x, v, u, w\}$ 这 8 个 3 度

顶点.

在情形 3.2.3 中, $Branch(v)$ 做两个分支, 或把 v 放入目标点覆盖集中或把 $N(v)$ 放入目标点覆盖集中. 第一个分支可以去掉 $\{v, u, w, x\}$ 这 4 个 3 度顶点, 第二个分支可以去掉 $\{v, u, w, x, a, b, c, d, e, f\}$ 这 10 个 3 度顶点. 递归关系式如下:

$$C(m) \leq C(m-4) + C(m-10) \quad (5)$$

在情形 3 中, 显然式(5)是最强的递归式, 这样可得到如下定理.

定理 3. 当 v 是一个度数为 3 的点, 则 $Branch(v)$ 操作的分支递归最少满足关系式(5).

在情形 3.2.3 的第一个分支中去掉 v 后, 使得 u, w, x 变成不相邻的 2 度顶点, 折叠它们将会使得 $Branch(v)$ 下一次对一个大于 4 度的点进行操作. 结合式(4), (5), 这样可以得到

$$\begin{aligned} C(m) &\leq C((m-4)-6) + C((m-4)-10) + C(m-10) \\ &= 2C(m-10) + C(m-14) \end{aligned} \quad (6)$$

显然式(6)是整个算法中所有递归式中最严格的一个.

定理 4. 对于任意一个 3 度图, $Branch(v)$ 操作的分支递归最少满足关系式(6).

5 结 果

定理 5. 本文算法解决 3 度图的最小点覆盖, 运行时间为 $O(1.1033^n)$.

证明. 根据定理 4, 若 G 为一个 3 度的图, 则 $Branch(v)$ 操作的分支递归始终最少满足(6), 不难证明 $C(m) = 1.1033^m$ 满足关系式(6) (其实也就是式(6)取等号时的解).

又显然, 图中 3 度点的个数少于图中所有点的个数, 即 $m \leq n$, 这样就有

$$C(m) = 1.1033^m \leq 1.1033^n \quad (7)$$

所以本算法运行时间为 $O(1.1033^n)$.

定理 6. 对于 3 度图参数化的点覆盖问题, 本文算法解决时间为 $O(kn+1.2174^k)$.

证明. 根据引理 1, 用 $O(\sqrt{k^3})$ 时间可以构造一个最多只有 $2k$ 个顶点和 $O(k)$ 条边的新图 G' , 将原点覆盖问题化到图 G' 上来. 这样就有

$$n \leq 2k \quad (8)$$

结合式(7)和式(8)可得

$$C(m) \leq 1.1033^n \leq 1.1033^{2k} = 1.2174^k \quad (9)$$

容易得出 $O(\sqrt{k^3}) = O(kn)$, 所以 3 度图参数化

的点覆盖问题可以在时间 $O(kn+1.2174^k)$ 下解决.

定理 7. 本文算法解决 3 度图的最大独立集问题, 运行时间为 $O(1.1033^n)$.

证明. 若 C 是图 G 的一个最小点覆盖集, 则 $V-C$ 是图 G 的一个最大独立集. 由定理 5 得, 此算法可以在运行时间 $O(1.1033^n)$ 下解决 3 度图的最小点覆盖问题, 则 3 度图的最大独立集问题也可在运行时间 $O(1.1033^n)$ 下解决.

6 结 束 语

本文考虑如何打破原问题 NP-Hard 结构为思想的出发点, 从另一个角度提出了构造递归结构的思路, 此思路不但有助于对 NP 结构的深入了解, 且已在低度图的点覆盖和独立集问题的解中得到应用得出了新的结果. 对于更高度的图问题, 也可应用此思想一步一步推算过去, 具体算法还在完善中.

本文算法中多次用到了一个小的操作技巧: 分支 4 度的圈. 应用此技巧可以大大简化以前一些相关论文^[14,15]中的算法及其复杂度分析, 特别是对于 3 度的图来说, 此操作可以得到关于参数 k 的递归关系式 $C(k) \leq 2C(k-4)$ (参看图 7 的情形 3.1.3, $\{v, x, c, b\}$ 或 $\{u, w, a, y\}$ 必须在目标点覆盖集中), 由此关系式甚至可以直接降低文献[15]中算法的结果.

另外, 在本文发稿之际, 我们利用文中思想和结果已将 3 度图参数化的点覆盖问题做到了 $O(1.190^k)$.

参 考 文 献

- Garey M., Johnson D.. Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman, San Francisco, 1979
- Downey R.G., Fellows M.R., Parameterized Complexity. New York: Springer, 1999
- Tarjan R.E., Trojanowski A.E.. Finding a maximum independent set. SIAM Journal on Computing, 1986, 7(4): 537~546
- Jian T.. An $O(2^{0.304n})$ algorithm for solving the maximum independent set problem. IEEE Transactions on Computers, 1986, 35(7): 847~851
- Shindo M., Tomita E.. A simple algorithm for finding a maximum clique and its worst-case time complexity. System and Computer in Japan, 1990, 21(1): 1~13
- Robson J.M.. Algorithms for maximum independent set. Journal of Algorithms, 1986, 7(4): 425~440

- 7 Beigel R. . Finding maximum independent sets in sparse and general graphs. In: Proceedings of the 10th CM-SIAM Symposium on Discrete Algorithms (SODA'99), 1999, 856~857
- 8 Kanj I. A. . Vertex cover: Exact and approximate algorithms and applications[Ph. D. dissertation]. Department Computer Science, Texas A&M University, College Station, Texas, 2001
- 9 Buss J. F. , Goldsmith J. . Nondeterminism within P . SIAM Journal on Computing, 1993, 22 (4): 560~572
- 10 Downey R. G. , Fellows M. R. . Parameterized computational feasibility. In: Clote P. , Rimmel J. , ed. . Feasible mathematics II, Boston, Birkhauser, 1995, 219~244
- 11 Balasubramanian R. , Fellows M. R. , Raman V. . An improved fixed parameter algorithm for vertex cover. Information Processing Letters, 1998, 65(3): 163~168
- 12 Stege U. , Fellows M. . An improved fixed-parameter-tractable algorithm for vertex cover. Department of Computer Science, ETH Zurich; Technical Report 318, 1999
- 13 Niedermeier R. , Rossmanith P. . Upper bounds for vertex cover further improved. Lecture Notes in Computer Science 1563, 1999, 561~570
- 14 Chen J. , Kanj I. A. , Jia W. . Vertex cover: Further observations and further improvements. Journal of Algorithms, 2001, 41(2): 280~301
- 15 Chen J. , Liu L. , Jia W. . Improvement on vertex cover for low-degree graphs. Networks, 2000, 35(4): 253~259
- 16 Bar-Yehuda R. , Even S. . A local-ratio theorem for approximating the weighted vertex cover problem. Annual Discrete Mathematics, 1985, 25(1): 27~46



XIAO Ming-Yu, born in 1979, master candidate. His research interests include computational complexity and optimization, computational geometry, computer aided geometric design.

CHEN Jian-Er, born in 1956, Ph. D. , professor. His research interests include computational complexity and optimization, graph theory and algorithm.

HAN Xu-Li, born in 1957, Ph. D. , professor. His research interests include approximation theory, computer aided geometric design.

Background

The Maximum Independent Set (MIS) problem and Vertex Cover (VC) problem, two of the six "basic" NP-complete problems, are the most important two lines of improving exact algorithms for solving NP-hard optimization problems. Both for the practical and theoretical research requirement, recently, new algorithms broking the upper bounds are frequently presented. For MIS problem, now the best algorithms are Robson's running time $O(1.211^n)$ algorithm and Beigel's running time $O(1.083^n)$ algorithm. For VC problem, the best algorithm is due to Chen *et al.* , with running time $O(kn+1.285^k)$.

One of the most important cases in the MIS and VC problem is the problem on low-degree graphs. And it is the bottleneck to improve exact algorithms of general graph. For MIS and VC problem on graphs of degree bounded by 3, now

the best results are running time $O(1.1033^n)$ and $O(kn+1.194^k)$ respectively.

In this paper, the authors concentrate on how to break the NP-hard structure of VC problem by reducing the vertices of the original graph step by step. If the NP-hard structure is broken, the problem can be solved in polynomial time. With this idea the authors build branching search trees from another aspect. And the recurrence relation functions are slightly different from the traditions. For Minimum Vertex Cover problem and Maximum Independent Set problem on graphs of degree bounded by 3, the above algorithm's running time is $O(1.1033^n)$; For parameterized Vertex Cover problem on graphs of degree bounded by 3, the running time is $O(kn+1.2174^k)$. All of the above results improve the previous best results.