无向图中子集反馈顶点集问题的精确算法

周晓清[1),2)] 肖鸣宇[1)]

[1)](电子科技大学计算机科学与工程学院 成都 611731)

[2)](成都大学信息科学与工程学院 成都 610106)

摘 要 子集反馈顶点集问题是一个经典的 NP 难问题,该问题是指在一个无向图中删除最少的顶点使得图中某些给定的顶点不在任何圈中.子集反馈顶点集问题包含了经典的最小反馈顶点集、多路割等重要特例问题,并且可应用于电路测试、操作系统解死锁等领域.子集反馈顶点集问题也是精确算法中的一个重要问题,该问题存在一个运行时间为 $O^*(2^n)$ 的简单暴力搜索算法,其中 n 为图中顶点数.直到 2011 年 Fomin 等人给出一个运行时间为 $O^*(1.8638^n)$ 的算法,这个运行时间界才被打破.文中将该运行时间上界进一步改进到 $O^*(1.7743^n)$.文中的算法是一个分支搜索算法,为了改进该问题的运行时间界,文中对问题的结构性质进行了深入的分析,挖掘出若干有效的规约和分支规则,再采用测量治之方法对算法的运行时间进行分析,最终将运行时间上界给予改进.

关键词 NP 难问题;精确算法;测量治之;子集反馈顶点集问题

中图法分类号 TP301 **DOI 号** 10.11897/SP.J.1016.2018.00493

Exact Algorithms for the Subset Feedback Vertex Set Problem in Undirected Graphs

ZHOU Xiao-Qing[1),2)] XIAO Ming-Yu[1)]

[1)](*School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu* 611731)

[2)](*College of Information Science and Engineering, Chengdu University, Chengdu* 610106)

Abstract Since 1971 when Cook proposed NP and NP-complete concepts, NP theory has been widely studied. NP-completeness has become one of the most important concepts in computational theory. When P≠NP, there is no polynomial time algorithm for any NP-complete problem. Exact algorithms aim to develop nontrivial fast exponential time algorithms for NP-complete and NP-hard problems. Some topics in this area, such as whether there is an algorithm faster than $O^*(2^n)$ for TSP, whether the maximum independent set problem can be solved in $O^*(1.0001^n)$ and so on, are well known hard tasks in exact algorithms and computational complexity. During the last 20 years, many elegant techniques have been developed to design exact algorithms and this area becomes a hot area in theoretical computer science. The measure-and-conquer method, introduced by Fomin et al., is one of the most important methods to investigate exact algorithms and parameterized algorithms. This method is based on a kind of amortization analysis, in which we set different weights to each vertex and use the sum of all vertex weights as the measure to evaluate the problem size. Compared to traditional measures, the weighted measure may catch more structural information of the graph and lead to a further improvement without modifying the algorithms. Nowadays, for many important NP-hard problems, the best exact algorithms are designed and analyzed by this new method. The subset feedback vertex set problem is a classical

收稿日期:2016-05-15;在线出版日期:2017-04-19.本课题得到国家自然科学基金(61370071)资助.**周晓清**,女,1982 年生,博士研究生,主要研究方向为算法分析与设计、精确算法和智能计算.E-mail:cduzxq@163.com.**肖鸣宇**(通信作者),男,1979 年生,博士,教授,中国计算机学会(CCF)会员,主要研究领域为精确算法、参数算法、组合最优化、图论及图算法.E-mail:myxiao@uestc.edu.cn.

NP-hard problem that has been extensively studied in exact algorithms. Given an undirected graph $G=(V,E)$ and a set $S\subseteq V$, the subset feedback vertex set problem asks us to delete a minimum number of vertices from G such that each vertex in S is not in any cycle in the remaining graph. This problem contains the feedback vertex set problem and the multiway cut problem as special cases. When $S\subseteq V$ includes all vertex of the graph, it is equivalent to the classical NP-hard feedback vertex set problem. When $S\subseteq V$ only contains one vertex and it does not allow to be delete, it can be transformed to the multiway cut problem. The subset feedback vertex set problem finds applications in circuit testing, network communication, deadlock recovery of operating system, synchronization system, artificial intelligence, biological computing and so on. The subset feedback vertex set problem is also an important problem in exact algorithms. A brute-force algorithm that tests each vertex subset of the graph gives a trivial running time bound of $O^*(2^n)$, where n is the number of vertices in the graph. This trivial bound was not broken until recently Fomin et al. gave an $O^*(1.8638^n)$-time algorithm in 2011. This paper will further improve the running time bound to $O^*(1.7743^n)$. The new algorithm is a branch-and-search algorithm that is analyzed by using the measure-and-conquer method. In order to get the improvement, this paper deeply analyzes the problem structural properties and explores a number of effective reduction and branching rules. With the help of the new reduction and branching rules, the measure-and-conquer method can reduce the problem instance effectively and then leads to the claimed running time bound.

Keywords NP-hard problem; exact algorithms; measure-and-conquer; the subset feedback vertex set problem

1 引 言

自从 1971 年 Cook[1] 提出 NP 和 NP 完全的概念以来，NP 理论得到了广泛研究，成为计算机领域中被引用最多的专业术语之一，NP 完全也成为计算理论中最为重要的概念之一. 当 P≠NP 时，NP 完全问题不存在多项式时间算法. 精确算法是通过研究快速的指数时间算法来解决 NP 完全问题和 NP 难问题. 在过去的几十年，精确算法得到了广泛研究，虽然不如智能算法、近似算法等领域火热，但是同样研究出了许多有趣的结果和方法，并且在计算理论中产生了重大的影响，见 Woeginger 的论文[2] 和 Fomin 与 Kratsch 的专著[3]. 事实上，很多 NP 完全问题都存在着一种非常简单的算法，即指数级的暴力搜索算法. 但是，到目前为止对于某些 NP 完全问题其最佳的算法仍然还是这种简单的暴力搜索算法，即便使用各种巧妙的算法设计技巧，它们的运行时间仍然未能得到改进. 例如 TSP 问题，在 20 世纪 60 年代就提出了运行时间为 $O^*(2^n)$ 的动态规划算法(其中 n 为图中的顶点数)，但是在过去 50 多年的大量研究中一直未能打破 2^n 这个上界，而该问题是否存在运行时间为 $O^*(1.99^n)$ 的算法已经成为计算理论中一个著名的公开难题. 另一个重要的问题是最大独立集问题(the maximum independent set problem)，目前该问题的运行时间界被改进到 $O^*(1.1996^n)$[4]，而将这个上界改进到 1.2^n 以下用了 30 多年. 这个上界是否能无限的改进下去呢？是否存在运行时间为 $O^*(1.0001^n)$ 的指数时间算法呢？这些都是精确算法中常提及的问题，其研究意义不仅仅在于解决该问题本身，而且有利于更好地认识 NP 完全问题的计算复杂性.

在过去几十年，有许多基础的 NP 完全问题其精确算法的运行时间不断被改进，同时也产生了许多新的算法设计技巧和方法. 例如，3-着色问题(the 3-coloring problem)可以在 $O^*(1.3289^n)$ 时间内解决[5]，最小顶点支配集问题(the dominating set problem)可以在 $O^*(1.4969^n)$ 时间内解决[6]，最小边支配集问题(the edge dominating set problem)可以在 $O^*(1.3160^n)$ 时间内解决[7]，最小反馈顶点集问题(the feedback vertex set problem)可以在 $O^*(1.7266^n)$ 时间内解决[8] 等等. 本文研究的是子

集反馈顶点集问题(the subset feedback vertex set problem)的精确算法.

图的一个反馈顶点集是一个顶点子集使得删除它以后剩下的图不存在任何圈.最小反馈顶点集问题就是在图中找出一个顶点数最少的反馈顶点集,该问题是一个经典的 NP 难问题[9].子集反馈顶点集问题是最小反馈顶点集问题的一般化,给出图 G 和图 G 一个顶点子集 S,要求找出图中一个最小的顶点集 C 使得删除 C 以后 S 中的所有顶点都不在任何圈中,这个顶点集 C 称为图的一个子集反馈顶点集.当 S 指定为图中所有顶点时,子集反馈顶点集问题退化为最小反馈顶点集问题.当 S 中仅包含一个顶点而此顶点不允许被删除时,子集反馈顶点集问题等价于另一个经典问题——多路割问题(the multiway cut problem)[10],这两个问题之间的相互转换见参考文献[11].

子集反馈顶点集问题在电路测试、网络通信、操作系统解死锁、分析工艺流程、同步系统、人工智能和生物计算等领域都有重要的应用[11-13].例如,在操作系统中,如果资源分配图含有圈则说明构成了死锁.为了解除死锁,需要停止部分进程,即删除资源分配图中某些点,使得图中不再含有圈,等价于在对应的图中寻找顶点数最少的反馈顶点集.在某些应用中,为了快速恢复操作系统,只需保证部分重要的进程不存在死锁,这等价于在对应的图中寻找顶点数最少的子集反馈顶点集.另一个例子是在遗传连锁(Genetic linkage)领域中的应用.遗传连锁是不同基因的等位基因呈现和基因偶联的现象.遗传连锁分析的目的是通过重组率来量化遗传连锁,重组率是指在特定染色体的重要区域发生交叉的概率.贝叶斯网络可以应用到遗传连锁分析中[14],在这种模型下,估计重组率问题可转化为贝叶斯网络中用谱系表示的更新问题,其本质是计算条件概率.但是通常这样的计算无论是时间上还是空间上都是低效的.Pearl 提出了用于解决更新问题的算法[15].Bar-Yehuda 等人将具有特定拓扑结构的贝叶斯网络的更新问题与无向图中的反馈集之间建立了联系,他们证明了 Pearl 提出的算法的运行时间成指数级取决于计算从贝叶斯网络中导出的无向图的反馈顶点集[16].由于任意分配一组基因,Pearl 提出的算法可以处理顶点产生的诱导圈,所以不用执行穷举搜索,估计重组率的复杂性由子集反馈顶点集所决定.

2　相关工作

自从 1979 年 Garey 等人[9]证明了子集反馈顶点集问题是 NP 完全问题以来,该问题得到了系统且广泛的研究.在近似算法领域里,Even 等人[12]最先开始对子集反馈顶点集问题进行研究,他们得到了一个 8 倍近似率的算法,这也是该问题第一个常数因子的近似算法.在参数算法领域里,以解集大小 k 为参数的子集反馈顶点集问题是否是固定参数可解的(Fixed Parameter Tractable,FPT)是知名的公开难题[17].无向图中子集反馈顶点集问题的固定参数可解性最后由 Kawarabayashi 等人[18]和 Cygan 等人[11]分别证明.随后 Wahlström[19]给出了第一个单指数运行时间的参数算法,其运行时间为 $4^k n^{O(1)}$.Hols 等人[20]则给出了该问题的第一个多项式大小的随机问题核.Chitnis 等人[21]证明了有向图上子集反馈顶点集问题的固定参数可解性,给了一个运行时间为 $2^{O(k^3)}n^{O(1)}$ 的参数算法.

在精确算法领域里,如果将图中所有的顶点子集都作为候选解考虑一次,可以得到运行时间为 $O^*(2^n)$ 的暴力搜索算法来解决子集反馈顶点集问题.这个运行时间上界很长时间都没有被打破.为了解决这个问题,一些学者研究在某些特殊图上的子集反馈顶点集问题.例如,Golovach 等人[22-23]证明了可以在 $O^*(1.5848^n)$ 时间内找到无向弦图上一个最小的子集反馈顶点集,同时证明了可以在 $O^*(1.6708^n)$ 时间内枚举(带权重的)无向弦图中所有的最小子集反馈顶点集.对于一般的无向图,$O^*(2^n)$ 这个运行时间上界直到 2011 年才被 Fomin 等人打破[24](全文版参见文献[13]).他们给出一个运行时间为 $O^*(1.8638^n)$ 的算法,该算法不仅可以找到一个最小的子集反馈顶点集,而且可以枚举出所有最小的子集反馈顶点集.Chitnis 等人[25]则研究了有向图上子集反馈顶点集问题,提出了一个运行时间为 $O^*(1.9993^n)$ 的精确算法.以上都是目前为止该问题已知的最好的精确算法.

另外,子集反馈顶点集问题的两个特例:最小反馈顶点集问题和多路割问题,在精确算法里也进行了深入研究.对于无向图中最小反馈顶点集问题,Razgon[26]第一次打破了 $O^*(2^n)$ 这个上界,提出了运行时间为 $O^*(1.8899^n)$ 的算法,随后这个结果不断被改进[8,27-28],Fomin 等人将结果改进到 $O^*(1.7347^n)$[27],Xiao 和 Nagamochi 将结果改进到

$O^*(1.7266^n)$[8]. 对于多路割问题,Chitnis 等人证明此问题可以在 $O^*(1.4766^n)$时间内解决[25].

3 本文贡献

本文将研究一般无向图中子集反馈顶点集问题的精确算法,改进该问题的精确算法运行时间上界. 正如上所述,一般无向图上的子集反馈顶点集问题的存在一个简单运行时间上界 $O^*(2^n)$,能否改进这个简单上界是一个多年的公开问题,直到 2011 年 Fomin 等人[13,24]将上界改进到 $O^*(1.8638^n)$,这也是目前已知最好的结果. 本文将这个运行时间上界进一步改进到 $O^*(1.7743^n)$.

本文算法是一个分支搜索算法. 通过对图性质的深入分析得到一系列有效的约简和分支规则,算法则是这些规则的一种合理组合. 传统的算法分析方法只能将该算法分析到 $O^*(2^n)$这个原始的运行时间上界. 为了得到改进的运行时间上界,文中采用测量治之方法(the measure-and-conquer method)[29]来对算法进行分析. 测量治之方法是一种相对较新的精确算法和参数算法分析技巧,其核心是分摊分析的思想,将图中的顶点根据其在计算复杂度上的贡献大小加以区分对待,赋予不同的权值,然后以图中所有顶点的权值和作为衡量问题大小的度量. 目前很多重要的 NP 难问题的最佳精确算法都是基于测量治之方法分析出来的. Fomin 等人[13]之所以能将该问题的原始运行时间上界打破也是因为采用了测量治之这个新的算法分析工具.

对比 Fomin 等人的算法[13]和本文中的算法,主要有如下一些异同点. 首先两个算法都是基于测量治之方法的分支搜索算法. 两个算法中最基本的分支规则都是不断选择图中一个顶点进行分情况讨论:要么将该点放入解集中,要么将该点留在最终的图(称为带约束的导出森林)中. 然而在每一步中选择不同点进行分支对问题的计算复杂度的下降是不一样的,两个算法都试图在自己的度量规则下寻找最有利于降低问题计算复杂度的顶点进行分支. 为了计算被分支的顶点对计算复杂度的贡献,两个算法均定义了"激活块"、"广义邻居"等概念. 然而本文在更深入研究了该问题的图结构性质后,定义的"广义邻居"等概念大大简化了文献[13]中的概念,使得算法分析更加简洁而且更加容易观察到问题的一些其它性质. 这样本文也得到了一些新的或精炼的约简和分支规则,所有这些都是文中算法的基础. 本文算法中最为关键的一个步骤是定理 6 以及由它产生的分支规则,其目的是为了避免在某些简单分支规则下的瓶颈情况而设计的一个复杂的分支规则. 总体来说,本文算法基本沿用了文献[13]中算法分析的方法和思路,但是本文更加细致挖掘了问题的图结构性质,在此基础上设计了更有效的简约和分支规则,最终实现算法的运行时间上界的改进.

本文第 4 节介绍基本概念和定义;第 5 节介绍分支搜索算法和测量治之方法的相关背景;第 6 节证明问题的一些性质并设计算法;第 7 节采用测量治之方法分析算法的运行时间;第 8 节进一步改进算法,得到最终的运行时间上界;第 9 节进行总结.

4 符号和问题定义

无特殊说明,本文提及到的所有图均指无向图,图中允许出现平行边和自环. 简单图是指没有平行边和自环的图. $G=(V,E)$表示一个顶点集为 V 和边集为 E 的图,其中 $|V|=n$ 和 $|E|=m$. 如果一个顶点子集仅包含一个元素 $\{v\}$,可以简单记为 v. 对于一个顶点子集 $X\subseteq V$,用 $G[X]$表示由 X 导出的子图,$G[V\setminus X]$也可以简写为 $G\setminus X$. 对图 G 一个顶点子集 $X\subseteq V$ 的压缩是指在 G 中删除 X 以及与 X 相连的边,然后添加一个新顶点 v^*,在 v^* 和 $v\in V\setminus X$ 之间添加 x_v 条平行边,其中 x_v 表示 G 中与集合 X 相连的边的条数. 压缩 G 中一个顶点子集 X 之后的图用 G/X 表示,压缩一个顶点子集后可能会产生平行边. 而压缩一条边就是压缩这条边的两个端点. 对于任意一个顶点 v,用 $N(v)$表示图中顶点 v 的邻居集合;$N_2(v)$表示距离顶点 v 为 2 的顶点集合;对于一个顶点子集 X,令 $N(X)=\bigcup_{v\in X}N(v)\setminus X$. 顶点 v 的度 $d(v)$是指图中一个端点为 v 的边的条数,其中 $d(v)\geq|N(v)|$. 在简单图中,$d(v)=|N(v)|$始终成立. 算法运行时间的表示中,用 O^* 符号省略多项式部分,如对 $f(n)$和 $g(n)$两个函数,用 $f(n)=O^*(g(n))$表示 $f(n)=g(n)\cdot n^{O(1)}$.

给定图 G 的一个顶点子集 $S\subseteq V$,称为终端. 如果图中的一个圈包含 S 中的顶点,则称该圈为 S-圈. 若图中存在一个顶点子集 $C\subseteq V$ 使得删除 C 以后,在剩余图 $G[V\setminus C]$中 $S\setminus C$ 中的任意顶点都不在圈中(即不存在 S-圈),则称顶点子集 C 为图 G 的一个 S-反馈顶点集. 对于图 G 的一个导出子图 G_1,如果 G_1 中没有 S-圈,则称 G_1 为 G 的一个 S-导出森林. 给定一个图 G 和终端集 S,最小子集反馈点集问

题就是求一个包含顶点数最少的 S-反馈顶点集；最大子集导出森林问题就是求一个包含顶点数最多的 S-导出森林. 可以看出这两个问题具有等价性，即如果一个顶点集 C 是最小子集反馈顶点集问题的解，当且仅当其导出子图 $G[V\backslash C]$ 是最大子集导出森林问题的解. 为了描述方便，文中将以求解最大子集导出森林问题的形式来阐述. 本文考虑的是如下一个带约束的最大子集导出森林问题：

带约束的最大子集导出森林问题
输入：一个无向图 $G=(V,E)$ 和 G 的两个顶点子集 $S,F\subseteq V$，其中导出子图 $G[F]$ 不存在任意的 S-圈.
问题：找到图 G 的一个最大顶点子集 $Y\subseteq V$ 满足如下性质：$F\subseteq Y$ 且导出子图 $G[Y]$ 中任何圈都不包含 $S\cap Y$ 中的顶点，即 $G[Y]$ 是一个 S-导出森林.

可以看出，当 F 为空集时，带约束的最大子集导出森林问题就是最大子集导出森林问题. 本文研究这种带约束条件的问题，其主要原因是在分支搜索算法中某些分支会要求将一些顶点保留在最后的森林中，那么可以将这些顶点直接加入 F 中. 对于带约束的最大子集导出森林问题，任意一个不含 S-圈并且包含 F 中所有顶点的子图称为该问题的一个可行解. 当一个可行解的大小达到最大时，就称该可行解为最优解，在本文中将一个最优解简称为一个解. 本文用 U 来表示图中不在 F 中的顶点集，即 $U=V\backslash F$.

5　分支搜索算法和测量治之方法

本文采用的算法是分支搜索算法，这种算法包含两种操作：约简和分支. 约简就是在多项式时间内将原问题实例转换成一个更小的等价实例. 例如，在求解最大子集导出森林问题中，图中所有度为 1 的顶点都可以被删除，这是因为度为 1 的顶点不会在任何圈中，所以可以将这些顶点直接加入最后的导出森林中. 注意，这种约简操作并不会指数级地增加算法的运行时间. 当不能对图进一步约简时，再将问题实例分成几种情况进行搜索求解，即分支操作. 分支就是将原问题实例分成几个较小的问题实例，通过求解所有这些小问题实例的解而得到原问题实例的解. 算法中的所有分支操作将会产生一棵“搜索树”，而搜索树的大小则对应于问题的指数运行时间部分.

为了对算法进行分析，需要选择一个度量来分析搜索树的大小. 例如，图中的顶点数是最常见的度量之一. 假设在某个分支操作中，一个包含 n 个顶点的图 G 被分成 l 个子实例 $G_1,G_2,\cdots,G_l$，其中 G_i 只有 $n-n_i$ 个顶点（$i=1,2,\cdots,l$），n_i 表示图 G 分解成子实例 G_i 后减少的顶点数. 用 $C(n)$ 表示在任意 n 个顶点的图上算法所产生搜索树大小的上界. 那么以上的分支操作将产生如下的一个递归关系式：

$$C(n)\leq\sum_{i=1}^{l}C(n-n_i).$$

函数 $f(x)=1-\sum_{i=1}^{l}x^{-n_i}$ 的最大实根 $\tau(n_1,n_2,\cdots,n_l)$ 称为上述递归关系式的分支因子，而分支因子也是衡量分支操作好坏的一个重要指标. 如果一个算法中所有分支操作对应的分支因子都不大于 τ，那么算法对应搜索树的大小为 $C(n)=O(\tau^n)$. 有关分支搜索算法的分析和递归方程求解的内容可以参考文献[3].

测量治之方法是 Fomin 等人近年来新提出的一种算法分析方法[29]，该方法在精确算法和参数算法中有着非常大的应用空间，在该方法提出后的五年内，基于测量治之方法的算法改进了许多 NP 难问题原有的最佳精确算法. 测量治之方法的核心是采用一种分摊分析的思想，对于图上的某个问题，测量治之方法将图中的顶点进行区分对待，并根据它们对问题整体求解的难易程度分别赋予不同权重，然后计算图中所有顶点的权值之和作为问题的度量，最后对分支操作设计相应的递归关系式. 按照这种方式设置的度量能更多地考虑图的结构信息，甚至可以在不改变原有算法的前提下改进算法的运行时间界. 例如，在测量治之方法中一种最为常见的权值设置方案就是根据图中顶点的度对每个顶点进行权重赋值[29]，度越大的顶点其权重值越大，在分支操作中，当删除一个顶点 v 时，问题度量的减少不仅包括 v 这个顶点的权重（删除 v），可能还包括 v 所有邻居权重的减少量（删除 v 后，其所有邻居的度减少 1，其权重也会减少）. 所以相对于以图中顶点数为度量的传统方式来说，这种方法可以从图结构中得到一些信息从而可以更多地减少问题度量的规模. 因此，测量治之方法中最为重要的一步是设置顶点的权重值，当列出所有递归关系式后可以通过求解一个拟凸规划得到一个最好的赋值方案，使得在该赋值方案下算法的最大分支因子达到最小. 本文将采用测量治之方法对子集反馈顶点集问题进行算法的设计和分析.

6　算法设计

6.1　初步思想

解决带约束的最大子集导出森林问题的一种最

简单的分支搜索算法就是任意拿出 U 中一个顶点 v 分两种情况进行讨论：要么 v 在 Y（最后的导出森林）中；要么 v 不在 Y 中. 对于前一种情况，可以将 v 加入 F 中；对于后一种情况，则可以直接删除 v. 如此不断地进行分支搜索，直到 U 为空集，这样就可以解决该问题. 然而在理论分析上，该方法只能得到 $O^*(2^n)$ 的运行时间界，很长时间以来，没有更好的方法突破这个简单的运行时间界，直到 Fomin 等人用测量治之方法来解决该问题[13,24]. 事实上，Fomin 等人的算法主要还是不断选出一个顶点 v 分两种情况进行讨论：要么将 v 留在最后的导出森林中，要么删除. 只是他们的算法不是随机地选择顶点进行分支搜索，而是机智地选取可能很快降低问题计算复杂度的顶点进行分支搜索，并采用测量治之方法对算法进行设计和分析. 本文同样采用测量治之方法对算法进行分析和设计，但是本文将更深入研究问题的结构性质，与此同时采用一种新的更有效的顶点选择方案进行分支搜索，从而将算法的运行时间界进一步改进.

首先，在一种特殊图 $G^*=(V^*,E^*)$ 中考虑带约束的最大子集导出森林问题. 这种特殊图只包含一个终端 $s\in S$，并且图中所有顶点到 s 的距离不大于 2，即 $V^*=\{s\}\cup N(s)\cup N_2(s)$ 且 $\{s\}\cup N(s)\subseteq F$，如图 1 所示.

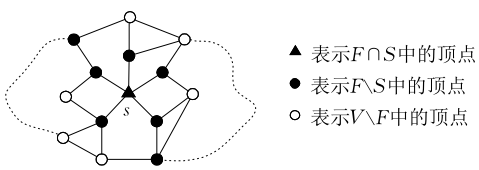

图 1 图中所有顶点距离 s 不大于 2 的特殊图

为了解决该问题，只需考虑 $N_2(s)$ 中的顶点是加入森林 Y 还是被删除. 由于该图的特殊性可以得到如下一个性质：如果图中存在 S-圈，则一定存在一个 S-圈 Q_0 至多只包含 $N_2(s)$ 中的两个顶点. 如果 Q_0 只包含 $N_2(s)$ 中的一个顶点 v，那么可以不用分支直接删除 v；如果 Q_0 正好包含 $N_2(s)$ 中的两个顶点 v 和 u，那么需要对 v 进行分支处理. 要么将 v 包含到 Y 中，要么将 v 删除，而将 v 包含到 Y 中必须删除 u，这样该分支可以同时减少两个顶点. 所以，在最差的分支情况下可以得到一个分支减少一个顶点，另一个分支减少两个顶点，从而该问题可以在 $O^*(1.6182^n)$ 的时间内解决，打破了原有 $O^*(2^n)$ 的时间界. 因此，文中算法的思想就是尝试将一般图一步步转化成这种特殊图，从而降低运行时间复杂度.

事实上，文中算法所处理的特殊图 G^* 原比上面描述的更为复杂，其主要原因是终端集 S 可能不止包含一个顶点，而且 F 可能包含 $\{S\}\cup N(S)$ 以外的顶点，这会导致问题更为复杂. 为了处理更为复杂的结构，文中将定义“广义邻居”和“激活块”等重要概念.

接下来先分析问题的图结构性质，给出一些约简和分支规则，然后再介绍整体算法流程.

6.2 定理和性质

一个问题实例 I 等价于另一个问题实例 I' 是指在多项式时间内可以通过实例 I 的一个解 Y 构造出实例 I' 的一个解 Y'，反之亦然. 文中将证明一些问题实例的等价性质，在这些等价性质的基础上定义一些规约操作，通过这些规约操作简化原问题实例.

定理 1. 对于一个带约束的最大子集导出森林问题实例 $I=(G,S,F)$，若 I 存在一个解不包含某个顶点 v，则删除 v 以后的实例 $I'=(G\setminus v,S\setminus v,F)$ 的任意一个解是原问题实例 I 的一个解；若 I 存在一个解包含某个顶点 u，则将 u 加入 F 以后的实例 $I''=(G,S,F\cup\{u\})$ 的任意一个解是原问题实例 I 的一个解.

证明. 设 $G'=G\setminus v$，Y' 为 I' 的任意一个最优解. 因为 G' 是 G 的一个子图，所以 Y' 也是 I 的一个可行解. 假设实例 I 的一个最优解 Y 不包含顶点 v，那么 Y 仍然是 I' 的一个可行解. 这种情况下有

$$|Y|\leq|Y'|.$$

此时 Y 是 I 的一个最优解.

令 Y'' 为 I'' 的任意一个最优解. 因为 Y'' 的导出子图中不包含 S-圈，因此 Y'' 是 I 的一个可行解. 假设实例 I 的一个最优解 Y 包含顶点 u，那么 Y 仍然是 I'' 的一个可行解. 这种情况下有

$$|Y|\leq|Y''|.$$

此时 Y'' 是 I 的一个最优解. 证毕.

定理 1 是文中很多分支操作的基础，当知道图中某个顶点在一个最优解中或者不在一个最优解中时，则可以利用定理将该顶点加入 F 中或者直接删除.

定理 2. 对于一个带约束的最大子集导出森林问题实例 $I=(G,S,F)$，用 $I'=(G',S',F')$ 表示压缩 $S\cap F$（或 $F\setminus S$）中两个相邻点 v 和 u 成一个新顶点 v^* 后的问题实例，那么 I 和 I' 这两个问题实例等价. 其中 $G'=G/\{v,u\}$，$F'=F\setminus\{v,u\}\cup\{v^*\}$，如果 $v,u\in S\cap F$ 则 $S'=S\setminus\{v,u\}\cup\{v^*\}$，否则 $S'=S$.

证明. 假设 Y' 是 I' 的任意一个解，以下证明 $Y=Y'\setminus\{v^*\}\cup\{v,u\}$ 是 I 的一个解. 如果 Y 不是 I

的一个解，则 $G[Y]$ 中存在一个 S-圈 Q'，这个圈必定通过 v 和 u 中一个或者两个顶点. 根据压缩操作的定义可知，在 G 中 v 和 u 除去它们本身以外的所有邻居在 G' 中都是 v^* 的邻居. 这样，在 Q' 中将 v、u 或者 $\{v,u\}$ 换成 v^* 后还能构造一个 S-圈，这与 $G[Y']$ 不包含 S-圈矛盾，所以 Y 是 I 的一个解.

下面假设 Y 是 I 的任意一个解，证明 $Y'=Y\backslash\{v,u\}\cup\{v^*\}$ 是 I' 的一个解. 如果 Y' 不是 I' 的一个解，则 $G[Y']$ 中存在一个 S-圈 Q 通过 v^*. 根据压缩操作的定义可知，对于 G' 中 v^* 的任意两个邻居 w_1 和 w_2，在 G 中 w_1 和 w_2 之间都有一条仅包含 $\{v,u\}$ 中顶点的路径. 那么，在 Q 中将 v^* 换成这条路径则可得到一个 S-圈，而该 S-圈在 $G[Y]$ 中，这与 Y 是 I 的一个解矛盾，所以 Y' 是 I' 的一个解.

由此可知 I 和 I' 等价.　　证毕.

根据定理 2 可知，压缩 $S\cap F$ 中的两个相邻点或 $F\backslash S$ 中的两个相邻点不会改变问题的最优性. 当定理 2 中蕴含的规则不能再压缩这个图的时候，$S\cap F$ 中的顶点构成一个独立集，$F\backslash S$ 中的顶点也构成一个独立集，这时实例具有如下性质：

性质 1.　假设定理 2 中的压缩规则不能再使用，并且如果 U 中一个顶点 v 与 $G[F]$ 中同一连通分量的两个不同顶点相邻，那么 v 一定在一个 S-圈中，且该圈只包含了 U 中的一个顶点 v.

定理 3.　对于实例 $I=(G,S,F)$，如果存在一个顶点 $v\in V$ 不在任何 S-圈中，则删除 v 后的实例 $I'=(G\backslash v,S\backslash v,F\backslash v)$ 与 I 等价.

证明.　假设 Y' 是 I' 的任意一个解，以下证明 $Y=Y'\cup v$ 是 I 的一个解. 如果 Y 不是 I 的一个解，那么 $G[Y]$ 中存在一个 S-圈 Q'，而且这个圈必定通过 v. 这与已知 v 在 G 中不属于任何 S-圈矛盾，所以 Y 是 I 的一个解.

下面假设 Y 是 I 的任意一个解，证明 $Y'=Y\backslash v$ 是 I' 的一个解. 令 $G'=G\backslash v$，如果 Y' 不是 I' 的一个解，那么 $G'[Y']$ 中存在一个 S-圈 Q. 由于 v 不在任何的 S-圈中，所以 S-圈 Q 必定不经过 v. 而 $Y=Y'\cup v$，这意味着 $G[Y]$ 中存在一个不包含 v 的 S-圈，这与 Y 是 I 的一个解矛盾，所以 Y' 是 I' 的一个解.

由此可知 I 和 I' 等价.　　证毕.

根据定理 3 可知，当知道某个顶点 v 不在任何 S-圈中时，删除 v 后的实例与原问题等价. 但是在求解带约束的最大子集导出森林问题时，根据定理 3 删除的所有顶点需要包含到原问题的最终解中.

定理 4.　如果存在一个 S-圈只包含 U 中的一个顶点，那么删除这个顶点后的实例与原实例等价.

证明.　假设图中一个 S-圈只包含 U 中的一个顶点 v. 用 I 表示原问题实例，I' 表示删除 v 以后的问题实例. 显然，v 一定不会在 I 的任何一个解 Y 中. 由定理 1 可知这两个问题的等价性.　　证毕.

根据定理 4 可知，在求解最大子集导出森林问题时，如果存在一个 S-圈只包含 U 中的一个顶点 v，那么可以将 v 直接从图中删除.

当 $F\cap S$ 不为空时，算法将选择 $G[F]$ 中一个包含有 S 中顶点的连通分量作为激活块，用 A_t 表示激活块. 激活块是一个重要的概念，基于这个概念本文才能将算法的运行时间给予改进. 最开始激活块是随机选择的，但是一旦某个 $G[F]$ 的连通分量选定为激活块后，即使该激活块压缩了（根据定理 2 两个相邻顶点进行压缩了）或者扩展了（与激活块相邻的 U 中顶点加入到 F 增大了该连通分量），该连通分量仍然是激活块. 仅仅当激活块不与图中任何 U 中的顶点相邻时，算法将改选 $G[F]$ 中另一个包含有 S 中顶点的连通分量作为激活块. 以下先认为激活块是 $G[F]$ 中任意一个包含有 S 中顶点的连通分量. $G[F]$ 中除激活块外其余连通分量称为非激活块.

如果一个顶点不在激活块中并且至少与激活块中一个顶点相邻时，称该顶点为激活块的附属点. 如果一个顶点不在非激活块中并且至少与非激活块中一个顶点相邻时，称该顶点为非激活块的附属点. 显然激活块的附属点和非激活块的附属点都属于 U，而且 U 的顶点可能既是激活块的附属点又是非激活块的附属点. 用 $N(A_t)$ 表示激活块所有附属点的集合. 对于一个激活块的附属点 v，定义它的广义邻居：U 中的顶点 u 称为 v 的广义邻居，当且仅当 u 不是激活块的附属点，并且 u 是 v 的邻居或者 u 和 v 都是同一个非激活块的附属点（即 u 和 v 之间存在一条路径，在该路径上除去 u 和 v 以外的顶点都属于非激活块）. 激活块附属点 v 广义邻居的集合用 $GN(v)$ 表示，广义邻居的个数用 $gd(v)$ 表示，即 $gd(v)=|GN(v)|$. 广义邻居是文中另一个重要的概念. 在图 2 的例子中，v 是激活块 A_t 的附属点，根据广义邻居的定义可知，$\{u_2,u_4\}$ 是 v 的广义邻居，

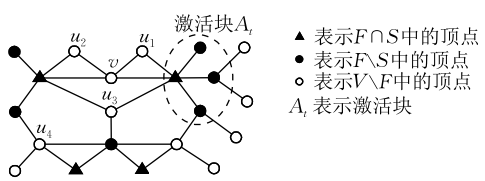

图 2　激活块及其附属顶点 v 的广义邻居

而$\{u_1,u_3\}$不是v的广义邻居,因为$\{u_1,u_3\}$是激活块的附属点.

根据广义邻居的定义可以得到如下性质.

性质 2. 如果一个顶点v的广义邻居u不是激活块的附属点,将v加入F后,则u变成新激活块的附属点.

下一个与广义邻居相关的定理在算法中被多次使用.

定理 5. 假设$S\cap F$中的顶点构成一个独立集且$F\setminus S$中的顶点也构成一个独立集,如果通过顶点v的任意S-圈至少包含U中的3个顶点且$gd(v)\leq 1$,那么该问题实例存在一个最优解包含v.

证明. 首先考虑$gd(v)=0$的情况.反设存在一个最优解Y'不包含v,以下推出矛盾.在反设条件下,生成子图$G[Y'\cup\{v\}]$中存在一个包含顶点v的S-圈C.令顶点u_1和u_2为v在C中的两个邻居,下面根据u_1和u_2的性质分为不同情况讨论.

情况 1. 假设u_1和u_2都是激活块中的顶点:如果u_1和u_2是激活块中两个不同的顶点,那么根据性质 1 可知,存在一个S-圈只包含U中的一个顶点v,这与定理的条件相矛盾.假设u_1和u_2是激活块中的同一顶点,即S-圈C只包含了v和$u_1=u_2$这两个顶点,而该S-圈也只包含了U中的一个顶点,矛盾仍然存在.因此情况 1 不可能发生.

情况 2. 假设u_1和u_2中有一个不是激活块中的顶点,不妨设u_2不是激活块中的顶点.

情况 2.1. u_2既不是激活块中的顶点也不是F中的顶点.既然v和u_2相邻且u_2不是v的广义邻居(因为$gd(v)=0$),根据广义邻居的定义可知u_2只能是激活块的附属点,并且v和u_2只能与激活块中同一个顶点$z\in F\setminus S$相邻.否则,若v和u_2分别与激活块中两个不同的顶点w_1和w_2相邻,记w_1和w_2在激活块中的路径为P,那么路径P与w_1v,vu_2,u_2w_2这3条边构成了一个S-圈,该S-圈只包含U中的两个顶点v和u_2,从而与定理的条件相矛盾.同时还可以看出$z=u_1$,否则v与激活块中两个不同的顶点相邻,由性质 1 可知,这与定理的条件相矛盾.显然v,u_2和$u_1=z$这3个顶点构成一个圈,而且这3个顶点都不是S中的顶点,否则存在一个S-圈只包含U中的两个顶点v和u_2,这与定理的条件矛盾.所以生成子图$G[Y'\cup\{v\}]$中的S-圈C肯定包含了除v,u_2和u_1之外的其它点,因为这3个点中没有S中的点.这样可以将C中的边u_1v和vu_2换成u_1u_2得到另一个不包含顶点v的S-圈C',而C'存在于生成子图$G[Y']$中,这与Y'是一个解相矛盾.因此情况 2.1 不可能发生.

情况 2.2. u_2不是激活块中的顶点却是F中的顶点.那么u_2是非激活块中的一个顶点,用A^*表示这个非激活块.由于C至少包含U中的3个顶点,因此C中至少还有A^*的另一个附属点$w\neq v$.由于v和w同为一个非激活块的附属点且w不是v的广义邻居,那么w也只能是激活块的附属点.根据情况 2.1 的分析可知,v和w只能与激活块中同一个顶点z相邻,并且$z=u_1$.在圈C中从v到w经过u_2的路径P上不存在S中的点,而且u_1也不是S中的点,否则P加上边vu_1和wu_1构成一个S-圈,该圈只包含U中的两个顶点并且其中一个是v,与定理的条件矛盾.这样C不是由P加上边vu_1和wu_1构成的,所以C中包含了更多的顶点.因此可以在C中将路径P和vu_1换成边wu_1得到另一个不包含顶点v的S-圈C',而C'存在于生成子图$G[Y']$中,这与Y'是一个解相矛盾.因此情况 2.2 同样不能发生.

由于情况 1 和情况 2 覆盖了所有可能,而且这两种情况都不会发生,因此该定理在$gd(v)=0$时成立.

下面考虑$gd(v)=1$的情况.令$GN(v)=\{v_0\}$.如果存在一个最优解Y'不包含v,那么可以证明$Y=Y'\cup\{v\}\setminus\{v_0\}$是另一个包含$v$的最优解.显然$|Y|\leq|Y'|$.反设$Y$不是一个解,那么在生成子图$G[Y]$中必定存在一个$S$-圈$C$,这个圈不包含顶点$v_0$但是包含顶点$v$.重复以上$gd(v)=0$情况的分析,始终得到矛盾.因此总存在一个最优解$Y$包含顶点$v$. 证毕.

对于一个问题实例,如果$S\cap F$中的顶点构成一个独立集,$F\setminus S$中的顶点也构成一个独立集,而且任意的S-圈至少包含U中的3个顶点,则称该实例为约简实例,约简实例中对应的图称为约简图.基于定理 5 可以推导出如下定理,这个定理是文中主要分支的理论基础,并且只在约简图上使用.

定理 6. 给出一个约简实例$I=(G,S,F)$,任选$G[F]$中包含有S中顶点的连通分量作为激活块.对于激活块的任意附属点v,实例I将存在一个最优解Y使得如下成立:

(1) 如果$gd(v)\leq 1$,则$v\in Y$;

(2) 如果$gd(v)=2$,则要么$v\in Y$,要么$GN(v)\subseteq Y$;

(3) 如果$gd(v)=3$,则要么$v\in Y$,要么$u_1\in Y$,要么$\{u_2,u_3\}\subseteq Y$,其中$\{u_1,u_2,u_3\}=GN(v)$.

证明.　由于该实例为约简的，因此不存在 S-圈至多只包含 U 中的两个顶点.这样(1)满足定理 5 的条件，因此(1)直接由定理 5 给出.

(2)令 Y'为一个问题实例的最优解.假设 Y'不包含 $GN(v)$中至少一个点，否则定理直接成立.这样集合 $X=GN(v)\backslash Y'$不为空.令 I'表示在原问题实例中删除 X 以后的问题实例(G'为实例 I'中对应的图).可以证明对于 I'的任意一个最优解 Y^* 也是原问题实例的一个最优解.因为 X 不为空，因此在 G'中，顶点 v 满足 $gd(v)\leq 1$.根据(1)可知 I'存在一个最优解 Y^* 包含顶点 v.那么 $Y^*\cup X$ 是原问题实例中一个包含顶点 v 的最优解.

(3)令 $GN(v)=\{u_1,u_2,u_3\}$，Y'为一个问题实例的最优解.假设 Y'不包含 u_1，也不包含 u_2和 u_3中最少一个点，否则定理直接成立.这样集合 $X=GN(v)\backslash Y'$ 最少包含两个元素.令 I'表示在原问题实例中删除 X 中顶点以后的问题实例(G'为实例 I'中对应的图).可以证明对于 I'的任意一个最优解 Y^* 也是原问题实例的一个最优解.因为 X 至少包含 $GN(v)$中的两个顶点，因此在 G'中，顶点 v 满足 $gd(v)\leq 1$.根据(1)可知 I'存在一个最优解 Y^* 包含顶点 v.那么 $Y^*\cup X$ 是原问题实例中一个包含顶点 v 的最优解.

综上所述，该定理成立.　　证毕.

定理 6 是本文最为重要的一个定理.定理 1～5 主要是对图进行约简操作，而定理 6 是本文算法中分支操作的理论依据.当所有的约简操作都不能再进行的时候，算法通过分支搜索来寻找一个解.根据定理 6，在一个约简图中，对于激活块的任意附属点 v，如果满足条件(1)，算法将 v 加入解中；如果满足条件(2)，算法进行分支操作，要么将 v 加入解中，要么将 v 的所有广义邻居加入解中；如果满足条件(3)，算法进行分支操作，要么将 v 加入解中，要么将 v 的广义邻居 u_1加入解中，要么将 v 的广义邻居 u_2和 u_3 加入解中.当定理 6 中 3 个条件都不成立的时候，对于激活块的其它附属点 v，算法进行简单的分支操作，要么将 v 加入 F 中要么删除.

6.3　算法设计

基于以上的定理和性质，下面进行算法的详细设计，其主要步骤参见算法 1.

算法 1.　算法 $MIF(G,S,F)$.

输入：一个无向图 $G=(V,E)$和 G 的两个顶点子集 S，$F\subseteq V$，其中导出子图 $G[F]$不包含任何 S-圈

输出：图 G 中满足 $F\subseteq Y$ 的最大 S-导出森林 Y

1. 如果图中所有的顶点都在 F 中，则返回 F 作为解.

2. 否则如果图中存在两个同属于 $S\cap F$ 或者同属于 $F\backslash S$ 的相邻顶点，则将这两个顶点进行压缩，令 I'为压缩后的实例，则返回 $MIF(I')$.

3. 否则如果图中存在一个顶点 v 不在任何 S-圈中且 v 不在激活块中，则返回 $MIF(G\backslash v,S\backslash v,F\cup\{v\})$.

4. 否则如果图中存在一个 S-圈只包含 U 中的一个顶点 v，则返回 $MIF(G\backslash v,S\backslash v,F)$.

5. 否则如果图中存在一个 S-圈只包含 U 中的两个顶点 u_1和 u_2，则返回 $MIF(G\backslash u_1,S\backslash u_1,F)$和 $MIF(G\backslash u_2,S\backslash u_2,F\cup\{u_1\})$中较大的解.

6. 否则如果 F 为空集或者 F 中的所有顶点都不与 U 中的顶点相邻，那么在图中任选一个不属于 F 的顶点 v，返回 $MIF(G\backslash v,S\backslash v,F=\varnothing)$和 $MIF(G,S,F=\{v\})$中较大的解，在 $MIF(G,S,F=\{v\})$实例中，将 $G[F]$中包含 v 的连通分量选为激活块.

7. 否则如果当前图中没有激活块或者激活块不与任何 U 中的顶点相邻，且图中存在有 $G[F]$的连通分量与 U 中的顶点相邻，则任意选择一个这样的连通分量为激活块.

8. 否则如果当前图中存在一个激活块的附属点 v 满足 $gd(v)\leq 1$，则返回 $MIF(G,S,F\cup\{v\})$.

9. 否则如果当前图中存在一个激活块的附属点 v 满足 $gd(v)=2$，则返回 $MIF(G,S,F\cup\{v\})$和 $MIF(G\backslash v,S\backslash v,F\cup GN(v))$中较大的解.

10. 否则如果当前图中存在一个激活块的附属点 v 满足 $gd(v)=3$，令 $GN(v)=\{u_1,u_2,u_3\}$，则返回 $MIF(G,S,F\cup\{v\})$，$MIF(G\backslash v,S\backslash v,F\cup\{u_1\})$和 $MIF(G\backslash\{v,u_1\},S\backslash\{v,u_1\},F\cup\{u_2,u_3\})$中较大的解.

11. 否则当前图中存在一个激活块的附属点 v 满足 $gd(v)\geq 4$，则返回 $MIF(G,S,F\cup\{v\})$和 $MIF(G\backslash v,S\backslash v,F)$中较大的解.

文中用 $MIF(G,S,F)$表示算法，输入部分为一个带约束的最大子集导出森林问题的实例，包括一个无向图 $G=(V,E)$和 G 的两个顶点子集 $S,F\subseteq V$，其中导出子图 $G[F]$不包含任何 S-圈.算法开始时，设 F 为空集.算法输出的是一个包含 F 中所有顶点的最大 S-导出森林 Y.该算法主要包含 11 个步骤，用递归算法的形式进行描述，即每一个步骤都调用算法本身.第 1 步当 $F=V$，即图中所有顶点都在 F 中时，直接返回 $F=V$ 作为解，这一步显然正确.第 2 步主要是对图中属于 F 的一些顶点进行压缩，该步的正确性由定理 2 给出.第 3 步则是删除图中任何不在 S-圈中的顶点，该步的正确性由定理 3 给出.注意到在这一步中删除的顶点需要包含到原问题的最终解中.第 4 步同样是从图中删除一些顶点，但这些顶点不会包含到最终解中，该步的正确性由定理 4 给出.第 5 步则是考虑存在一个 S-圈只包含

U 中两个顶点 u_1 和 u_2 的情况. 算法分为两个分支，第 1 个分支将 u_1 从图中删除，第 2 个分支将 u_1 包含到 F 中，注意这时 u_2 成为一个 S-圈中唯一属于 U 的顶点，因此根据定理 4 可以直接删除 u_2. 当前 5 步都不被执行时，此时的图为一个约简图. 以下步骤中默认图都是一个约简图. 第 6 步和第 7 步是进行激活块的选定. 在第 6 步中，如果图中 F 为空集，那么任意选择一个点进行分支，要么删除该顶点要么将该顶点加入 F 中，在将顶点加入 F 的这个分支中，选择 $G[F]$ 为激活块. 第 7 步则是选择 $G[F]$ 中的一个连通分量作为激活块，选择的规则是要求激活块一定有附属点. 第 8～11 步则是选择激活块的一个附属点并根据其广义邻居的个数进行分支，其中第 8～10 步的正确性由定理 6 给出，第 11 步则是做简单的分支，要么删除这个附属点，要么将其放入 F 中. 可以看出，只要在图中 $U=V\setminus F$ 不为空集，6 至 11 步中至少有一步可以执行，而当 $V=F$ 时，第 1 步将会执行并且终止整个算法. 由以上的分析可以得到该算法的正确性，以下则是算法运行时间的分析.

7 基于测量治之的算法运行时间分析

根据以上设计的算法可知，算法的第 1，2，3，4，7 和 8 步只进行了约简操作，第 5，6，9，10 和 11 步进行了分支操作. 由于约简操作不会增加搜索树的大小，所以以下仅对分支操作进行分析.

基于测量治之的分析方法首先需要确定衡量问题大小的度量，然后以此度量为所有分支操作设计递归关系式. 对于一个问题实例 (G,S,F)，其度量定义如下：

$$\mu(G,S,F)=\alpha|N(A_t)|+|V\setminus(F\cup N(A_t))|,$$

其中 $0<\alpha<1$ 是一个实数，α 的具体值将在后面给出. 换句话说，就是将 F 中的顶点权重设为 0，激活块附属点的权重设为 α，U 中的其余顶点权重设为 1，然后将图中所有顶点的权重之和 $\mu(G,S,F)$ 作为度量. 由于 α 是一个小于 1 的正数，这说明算法认为激活块的附属点比 U 中的其它顶点在计算难易程度上的贡献要小，其主要原因是基于 6.1 节中的观察.

度量 $\mu(G,S,F)$ 具有如下一些性质：

(1) 任何时候度量 $\mu(G,S,F)$ 的值都不会大于图中顶点的个数. 特别地，令 $\mu=\mu(G,S,\varnothing)$ 为开始时问题实例的度量，则有 $\mu\leq n$.

(2) 当度量值为 0 时，问题实例可以很快得以解决(常数时间内).

(3) 当执行第 1，2，3，4，6 和 7 步中任何一步时，度量都不会增加.

这是因为在算法第 3 步中没有删除激活块中的顶点，而第 6 步仅在激活块没有附属点时才更换激活块，因此激活块不会在算法中消失并且其激活块的附属点的个数也不会减少. 而其余的第 1，2，4 和 7 步中的度量显然不会增加，由此可得上面第 3 条性质的正确性.

以上 3 条性质是算法分析的基础，下面分析在该度量下各分支操作产生的递归关系式. 文中用 $C(\mu)$ 表示算法在度量为 μ 的问题实例上产生的搜索树大小的上界.

第 5 步. 在这一步中第 1 个分支删除了 U 中的一个顶点 u_1，u_1 可能是激活块的附属点，因此在这个分支上度量至少减少 α. 第 2 个分支至少减少 U 中的两个顶点 u_1 和 u_2，从而度量至少减少 2α. 这一步可以得到如下递归关系式：

$$C(\mu)\leq C(\mu-\alpha)+C(\mu-2\alpha).$$

第 6 步. 这一步选择一个非激活块的附属点 v 进行分支. 注意到 v 至少在一个圈中且这个圈包含两个以上(U 中)的顶点，否则它将在第 3 步中被删除或者在第 5 步中被处理. 这样 v 至少有两个邻居在 U 中. 第 1 个分支删除 v 从而度量减少 1. 第 2 个分支将 v 加入 F 中同时选取 $G[F]$ 中包含 v 的连通分量为激活块，将 v 加入 F 中度量减少 1，v 的邻居变成激活块的附属点度量由 1 减少到 α，由于其邻居个数最少为 2 其度量至少减少 $2(1-\alpha)$，那么第 2 个分支的度量至少减少 $3-2\alpha$. 这一步可以得到如下递归关系式：

$$C(\mu)\leq C(\mu-1)+C(\mu-(3-2\alpha)).$$

以下根据激活块附属点 v 广义邻居的个数进行分析.

第 9 步. 当 $gd(v)=2$ 时，第 1 个分支将 v 放入 F 中，$GN(v)$ 中的所有顶点都变成激活块的附属点，因此度量至少减少 $\alpha+2(1-\alpha)=2-\alpha$. 第 2 个分支将 v 删除同时将 $GN(v)$ 中的两个顶点放入 F 中，这样度量减少 $\alpha+2$. 这一步可以得到如下递归关系式：

$$C(\mu)\leq C(\mu-(2-\alpha))+C(\mu-(2+\alpha)).$$

第 10 步. 当 $gd(v)=3$ 时，第 1 个分支将 v 加入 F 中，$GN(v)$ 中的 3 个顶点都变成激活块的附属点，因此度量减少 $\alpha+3(1-\alpha)=3-2\alpha$. 第 2 个分支将 v 删除同时将 u_1 加入 F 中，度量减少 $\alpha+1$. 第 3 个分支将 v 和 u_1 删除同时将 u_2 和 u_3 加入 F 中，度量减少 $\alpha+3$. 这一步可以得到如下递归关系式：

$$C(\mu)\leq C(\mu-(3-2\alpha))+C(\mu-(1+\alpha))+C(\mu-(3+\alpha)).$$

第 11 步. 当 $gd(v)\geq 4$ 时，对 v 进行简单分支，要么将 v 加入 F 中要么删除. 由于此时 $gd(v)\geq 4$，当 v 加入 F 后，至少其 4 个广义邻居变成激活块的附属点，其度量至少减少 $\alpha+4(1-\alpha)=4-3\alpha$. 这一步可以得到如下递归关系式：

$$C(\mu)\leq C(\mu-(\alpha+gd(v)(1-\alpha)))+C(\mu-\alpha)\leq C(\mu-(4-3\alpha))+C(\mu-\alpha).$$

将上述所有递归关系式作为约束条件组成一个拟凸规划的约束条件，目标是使以上递归关系式中最大分支因子最小，其中 α 为变量. 利用文献[30]中介绍的方法对这个拟凸规划进行求解得到，当 $\alpha=0.8$ 时，上述递归关系的最大分子因子为 1.8249，因此该算法运行时间上界为 $O^*(1.8249^\mu)$. 由于在初始图上 $\mu\leq n$，因此带约束的最大子集导出森林问题可以在 $O^*(1.8249^n)$ 时间内解决. 该结果改进了原有算法的运行时间界 $O^*(1.8638^n)$.

8　进一步改进

在以上算法第 5 步的递归关系式中产生了唯一的最大分支因子. 一种改进算法的思想是避免在算法中出现第 5 步的分支操作. 但是算法必须先处理第 5 步中的 S-圈（该 S-圈只包含 U 中两个顶点），才能保证定理 5 以及后面一些分支规则的正确性，所以在该小节中将更细致地分析这种 S-圈的性质，从而使用一些更为有效的分支操作来替换原有的分支操作，改进原有算法的运行时间界. 改进的算法是将原算法的第 5 步替换成如下系列操作.

为了叙述方便，将这种只包含 U 中两个顶点的 S-圈称为关键圈. 对于 U 中的一个顶点 v 来说，如果 U 中另一个顶点 u 使得存在一个只包含 v 和 u 的关键圈，则称 u 为 v 的关键伴侣. 显然，若 U 中的一个顶点在关键圈中当且仅当它的关键伴侣个数不为零. 文中用 $CN(v)$ 来表示顶点 v 关键伴侣的集合. 首先处理关键伴侣个数大于 1 的顶点.

情况 1. 如果存在一个顶点 v 其关键伴侣个数至少为 2，算法分为两个分支，第 1 个分支直接删除 v，第 2 个分支将 v 加入 F 中，并且所有 $CN(v)$ 中的顶点都将删除（根据定理 4）. 因此算法返回 $MIF(G\backslash v,S\backslash v,F)$ 和 $MIF(G\backslash CN(v),S\backslash CN(v),F\cup\{v\})$ 中较大的解.

由于 v 和 $CN(v)$ 中的顶点可能都是激活块的附属点，所以这些顶点的权值有可能都只是 α，因此得到如下递归关系式：

$$C(\mu)\leq C(\mu-\alpha)+C(\mu-(1+|CN(v)|)\alpha)\leq C(\mu-\alpha)+C(\mu-3\alpha).$$

情况 2. 如果图中存在一个关键圈且该圈包含一个不是激活块的附属点. 令 v 和 u 为这个圈中两个 U 中的顶点，其中 v 不是激活块的附属点，则算法在 v 上进行分支，要么删除 v，要么将 v 加入到 F 中，即返回 $MIF(G\backslash v,S\backslash v,F)$ 和 $MIF(G\backslash u,S\backslash u,F\cup\{v\})$ 中较大的解.

由于 v 不是激活块的附属点，所以 v 的权值为 1，因此可以得到如下递归关系式：

$$C(\mu)\leq C(\mu-1)+C(\mu-(1+\alpha)).$$

以下假设任何 U 中的顶点其关键伴侣个数都不大于 1，并且所有关键圈中属于 U 的顶点（即关键伴侣个数为 1 的顶点）都是激活块的附属点.

情况 3. 如果图中存在一个关键圈包含激活块的两个附属点，设为顶点 v 和 u. 算法选择 v 和 u 中广义邻居个数较多的顶点进行分支. 为了不失一般性，假设 $gd(v)\geq gd(u)$. 算法在 v 上进行分支，要么删除 v，要么将 v 加入 F 中，即返回 $MIF(G\backslash v,S\backslash v,F)$ 和 $MIF(G\backslash u,S\backslash u,F\cup\{v\})$ 中较大的解.

当 $gd(u)\geq 2$ 时，因为 $gd(v)\geq gd(u)$，所以在第 2 个分支 U 中至少有两个顶点变成激活块的附属点，其度量至少减少 $2\alpha+2(1-\alpha)=2$，得到如下递归关系式：

$$C(\mu)\leq C(\mu-\alpha)+C(\mu-2).$$

当 $gd(u)\leq 1$ 时，算法稍微有所不同. 在第 1 个分支操作删除 v 以后，u 所在的 S-圈至少包含 U 中 3 个顶点（u 的关键伴侣个数为 1），又因 $gd(u)\leq 1$，定理 5 成立，所以可以将 u 加入 F 中. 因此当 $gd(u)\leq 1$ 时，算法返回 $MIF(G\backslash v,S\backslash v,F\cup\{u\})$ 和 $MIF(G\backslash u,S\backslash u,F\cup\{v\})$ 中较大的解. 此时这两个分支都减少了激活块的两个附属点 v 和 u，度量至少减少 2α，得到如下递归关系式：

$$C(\mu)\leq C(\mu-2\alpha)+C(\mu-2\alpha).$$

将第 5 步进行上述改进以后，新解拟凸规划可得，当 $\alpha=0.6667$ 时，算法中递归关系式中最大的分子因子为 1.7743，所以得到如下定理.

定理 7. 子集反馈顶点集问题可以在 $O^*(1.7743^n)$ 时间内解决.

9　总　结

本文针对无向图中子集反馈顶点集问题设计了

一个运行时间为 $O^*(1.7743^n)$ 的算法，该算法改进了原有算法的最佳运行时间界 $O^*(1.8638^n)$. 文中算法的核心思想是将激活块的附属点和 U 中的其它顶点加以区分，并利用测量治之方法进行运行时间分析. 目前，基于测量治之的方法使得很多重要的 NP 难问题的精确算法得以改进，而该方法在更多 NP 难问题的精确算法中的应用正在进一步拓展中. 文中的算法只适合无向图中的子集反馈顶点集问题，对于有向图中子集反馈顶点集问题目前最好的精确算法的运行时间上界是 $O^*(1.9993^n)$[25]，非常接近 $O^*(2^n)$. 此运行时间的上界能否改进到 $O^*(1.8^n)$ 或更好，这是下一步值得研究的问题.

参 考 文 献

[1] Cook S A. The complexity of theorem proving procedures// Proceedings of the ACM Symposium on Theory of Computing. Shaker Heights, Ohio, USA, 1971: 151-158

[2] Woeginger G J. Exact algorithms for NP-hard problems: A survey. Lecture Notes in Computer Science, 2010, 2570: 185-208

[3] Fomin F V, Kratsch D. Exact Exponential Algorithms. Berlin, Germany: Springer, 2010

[4] Xiao M, Nagamochi H. Exact Algorithms for Maximum Independent Set. Berlin, Germany: Springer, 2013: 328-338

[5] Beigel R, Eppstein D. 3-coloring in time $O(1.3289^n)$. Journal of Algorithms, 2005, 54(2): 168-204

[6] Van Rooij J M M, Bodlaender H L. Exact algorithms for dominating set. Discrete Applied Mathematics, 2011, 159(17): 2147-2164

[7] Xiao M, Nagamochi H. A refined exact algorithm for edge dominating set. Theoretical Computer Science, 2014, 560: 207-216

[8] Xiao M, Nagamochi H. An improved exact algorithm for undirected feedback vertex sets. Journal of Combinatorial Optimization, 2015, 30(2): 214-241

[9] Garey M R, Johnson D S. Computers and Intractability: A guide to the theory of NP-completeness. New York, USA: W. H. Freeman and Company, 1979

[10] Calinescu G. Multiway Cut. In Encyclopedia of Algorithms. USA: Springer, 2008, 12: 1-99

[11] Cygan M, Pilipczuk M, Pilipczuk M, Wojtaszczyk J O. Subset feedback vertex set is fixed-parameter tractable. SIAM Journal on Discrete Mathematics, 2013, 27(1): 290-309

[12] Even G, Naor J, Zosin L. An 8-approximation algorithm for the subset feedback vertex set problem. SIAM Journal on Computing, 2000, 30(4): 1231-1252

[13] Fomin F V, Heggernes P, Kratsch D, et al. Enumerating minimal subset feedback vertex sets. Algorithmica, 2014, 69(1): 216-231

[14] Neuman R. Analysis of human genetic linkage, revised edition. Behavior Genetics, 1993, 23(3): 299-300

[15] Pearl J. Probabilistic reasoning in intelligent systems: Networks of plausible inference. Computer Science Artificial Intelligence, 1988, 70(2): 1022-1027

[16] Bar-Yehuda R, Geiger D. Approximation algorithms for the feedback vertex set problem with applications to constraint satisfaction and Bayesian inference. SIAM Journal on Computing, 1998, 27(4): 942-959

[17] Kakimura N, Kawarabayashi K, Kobayashi Y. Erdös-Pósa property and its algorithmic applications: Parity constraints, subset feedback set, and subset packing//Proceedings of the ACM-SIAM Symposium on Discrete Algorithms. San Francisco, USA, 2012: 1726-1736

[18] Kawarabayashi K I, Kobayashi Y. Fixed-parameter tractability for the subset feedback set problem and the S-cycle packing problem. Journal of Combinatorial Theory, 2012, 102(4): 1020-1034

[19] Wahlström M. Half-integrality, LP-branching and FPT algorithms. Arxiv Preprint Arxiv, 2013: 1762-1781

[20] Hols E M C, Kratsch S. A randomized polynomial kernel for subset feedback vertex set//Proceedings of the 33rd Symposium on Theoretical Aspects of Computer Science (STACS 2016). Orléans, France, 2016: 43:1-43:14

[21] Chitnis R, Cygan M, Hajiaghayi M, Marx D. Directed subset feedback vertex set is fixed-parameter tractable. ACM Transactions on Algorithms, 2010, 11(4): 1-28

[22] Golovach P A, Heggernes P, Kratsch D, Saei R. An exact algorithm for subset feedback vertex set on chordal graphs// Proceedings of the 7th International Conference on Parameterized and Exact Computation. Ljubljana, Slovenia, 2012: 7535: 85-96

[23] Golovach P A, Heggernes P, Kratsch D, Saei R. Subset feedback vertex sets in chordal graphs. Journal of Discrete Algorithms, 2014, 26: 7-15

[24] Fomin F V, Heggernes P, Kratsch D, et al. Enumerating minimal subset feedback vertex sets//Proceedings of the Algorithms and Data Structures: 12th International Symposium. New York, USA, 2011: 399-410

[25] Chitnis R, Fomin F V, Lokshtanov D, Misra P. Faster Exact Algorithms for Some Terminal Set Problems. Berlin, Germany: Springer, 2013: 150-162

[26] Razgon I. Exact computation of maximum induced forest// Proceedings of the 10th Scandinavian Conference on Algorithm Theory (SWAT 2006). Berlin, Germany: Springer, 2006: 160-171

[27] Fomin F V, Villanger Y. Finding induced subgraphs via minimal triangulations//Proceedings of the 27th International Symposium on Theoretical Aspects of Computer Science (STACS 2010). Nancy, France, 2009: 383-394

[28] Fomin F V, Gaspers S, Pyatkin A V, Razgon I. On the minimum feedback vertex set problem: Exact and enumeration algorithms. Algorithmica, 2008, 52(2): 293-307

[29] Fomin F V, Grandoni F, Kratsch D. A measure & conquer approach for the analysis of exact algorithms. Journal of the ACM, 2009, 56(5): 1-32

[30] Eppstein D. Quasiconvex analysis of multivariate recurrence equations for backtracking algorithms. ACM Transactions on Algorithms, 2006, 2(4): 492-509

ZHOU Xiao-Qing, born in 1982, Ph. D. candidate. Her research interests include design and analysis of algorithms, exact algorithms and intelligent computing.

XIAO Ming-Yu, born in 1979, Ph. D., professor. His research interests include exact algorithms, parameterized algorithms, combinatorial optimization, graph theory and graph algorithms.

Background

This paper focuses on exact algorithms, which develop fast nontrivial exponential algorithms for NP-hard problems. Some topics in this area, such as whether there is an algorithm faster than $O^*(2^n)$ for TSP, whether the maximum independent set problem can be solved in $O^*(1.0001^n)$ and so on, are well known hard tasks in algorithms and computational complexity. During the last 20 years, many elegant techniques have been developed to design exact algorithms and this area becomes a hot area in theoretical computer science. The measure-and-conquer method, introduced by Fomin et al. [Journal of the ACM, 2009], is one of the most important methods to investigate exact algorithms and parameterized algorithms. Nowadays, the running time bounds of exact algorithms for many NP-hard problems have been improved by using this new method. The subset feedback vertex set problem is one of the most extensively studied problems in exact algorithms, since it contains the feedback vertex set problem and the multiway cut problem as special cases. However, the trivial running time bound of $O^*(2^n)$ for the subset feedback vertex set problem was not broken until Fomin et al. gave an $O^*(1.8638^n)$-time algorithm in 2011. This paper gives a simple and practical algorithm for the subset feedback vertex set problem. By using the measure-and-conquer method, we can prove that the running time bound is $O^*(1.7743^n)$, better than the known bound of $O^*(1.8638^n)$. In China mainland, some universities, include Central South University, University of Electronic Science and Technology of China and so on, have research groups focusing on exact and parameterized algorithms. The problem studied in this paper is a task in the project, titled "The Measure-and-Conquer Method and Its Applications in Algorithm Design", supported by the National Natural Science Foundation of China under Grant No. 61370071. The result in this paper completely finishes this task.