任意图支配集精确算法回顾

路 纲[1] 周明天[2] 唐 勇[2] 吴振强[1] 裘国永[1] 袁 柳[1]

[1](陕西师范大学计算机科学学院 西安 710062)

[2](电子科技大学计算机科学与工程学院 成都 610054)

摘 要 该文综述了任意图支配集精确算法分析和设计的新进展.支配集问题是经典 NP 完全问题,很多问题都能与它相联系.我们针对最小支配集、最大独立集、最小独立支配集、最小连通支配集、最小加权支配集问题提供了详尽算法描述和实例说明,以使文章自包含方便阅读.文中还讨论了诸如分支简化策略、复杂度分析、测度分析、记忆等技术.自 Claude Berge 首次准确阐述现代图支配概念后,经过很长一段时期的沉寂,关于指数时间精确算法设计的研究热情在过去五年中显著增涨.除回顾这些最新成果之外,作者还盼望国内研究团体能更加重视这个快速发展的研究领域.

关键词 支配集;精确算法;计算复杂性;图;测度分析技术

中图法分类号 TP393 **DOI 号**: 10.3724/SP.J.1016.2010.01073

A Survey on Exact Algorithms for Dominating Set Related Problems in Arbitrary Graphs

LU Gang[1] ZHOU Ming-Tian[2] TANG Yong[2] WU Zhen-Qiang[1] QIU Guo-Yong[1] YUAN Liu[1]

[1](*School of Computer Science, Shaanxi Normal University, Xi'an* 710062)

[2](*College of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu* 610054)

Abstract This paper provides a review on recent advances in the analysis and design of exact algorithms for domination in arbitrary graphs. The dominating set problem is one of the classical NP-complete problems and fits into broader class of domination problems. The authors provide the detailed algorithms, which have been very recently obtained, and illustrations for the domination problems in arbitrary graphs including minimum dominating set, maximum independent set, minimum independent dominating set, minimum connected dominating set and minimum weighted dominating set for the purpose of making the paper be self-contained and easy reading. Several techniques such as Branch & Reduce, Complexity analysis, Measure & Conquer, Memorization and so on are also discussed. After a long period of silence since Claude Berge formulated for the first time of modern conception of domination, the interest in design of exact exponential time algorithms has been growing significantly within the last five years. In addition to present a view of the latest results, the authors also hope more domestic research communities would pay attention to this rapidly developed field of research.

Keywords dominating set; exact algorithm; computational complexity; graph; measure and conquer analysis

---

收稿日期:2008-01-15;最终修改稿收到日期:2010-02-01.本课题得到国家自然科学基金(60633020)、国家"八六三"高技术研究发展计划项目基金(2007AA01Z438200)和陕西师范大学科研项目基金(999414)资助.**路 纲**,男,1972 年生,博士,主要研究方向为计算几何、图论及其应用.E-mail: goforlg@126.com.**周明天**,男,1939 年生,教授,博士生导师,主要研究领域为网络计算、信息安全.**唐 勇**,男,1973 年生,博士,副教授,主要研究方向为网络计算、网络管理和无线网络.**吴振强**,男,1968 年生,博士,副教授,主要研究方向为网络与信息安全、移动计算和可信计算.**裘国永**,男,1966 年生,博士,副教授,研究方向为组合优化理论.**袁 柳**,女,1979 年生,博士,主要研究方向为语义网络、信息检索.

1 引 言

自 Claude Berge 在 20 世纪 60 年代初首次描述现代图的支配概念至今(2010),整整 50 年过去了,支配集及其各种变形问题已成为图论、组合数学、计算机科学等领域被广泛研究和重视的问题之一,发表的相关论文数以千记,文献[1]综述了 20 世纪 90 年代前约 1200 多篇支配集相关文章的内容.不仅限于理论研究,支配集问题同样有着重要的实用价值:通信基础设施的布局、城市交通路线规划、测量及地理信息系统、系统控制和机构优化管理、编码理论、图像处理、数值分析、VLSI 设计、信息检索、超级计算机的 CPU 阵列布局和通信机制、生物学、材料学、物理学、军事学等众多领域的一些重要问题都可归结为某种形式的支配集问题.

人们在 20 世纪 70 年代就认识到任意图最小支配集问题及其变形如最小独立支配集、最大独立集等问题的 NP-难性质,在 $P=NP$ 这个目前被广泛认为不大可能的结果出现之前,要获得问题的准确解只能求之于某个指数 $O(c^n)$ 时间的算法,其中 c 是大于 1 的常数,n 是图顶点数,本文简称这类指数时间的准确解算法为精确算法.

精确算法设计有着悠久的历史.非平凡精确算法的历史可以追溯到 1962 年 Held 和 Karp 发表的 TSP 问题动态规划算法[2]——迄今仍是最快算法之一($O(2^n n^2)$);Tarjan 等[3]在 1977 年提出 $O(2^{n/3})$ 最大独立集算法,规则描述多达十几页.令人困惑的是,就最小支配集问题来说,直到 2004 年前后才由一些独立研究者如 Fomin、Grandoni 等人突破 2^n 这个平凡界限.2004 年以来精确算法研究逐渐形成一个热潮——本文主要工作就是介绍这期间的重要成果.

经过 20 世纪 90 年代的沉寂后,精确算法近期再次引起重视大致有以下原因:

(1)受计算复杂性理论制约,指数时间算法是目前获得精确解的唯一可行选择.

(2)近似算法不一定都能满足需要,并且许多 NP 难问题没有或很难获得近似算法.例如,著名的 3-SAT 问题,要么为真要么为假,采用近似算法毫无意义.

(3)设计近似算法时,如果能事先获得一些小规模网络的精确解感性认识,将对设计大有助益.

(4)如果能使精确算法的时间底数降低,例如 c 降至 $\sqrt{c}$,可解决问题规模将翻倍.当问题规模不是很大且 c 接近 1 时,精确算法甚至比某些高次多项算法表现更好.

(5)计算机硬件技术的进步、计算资源的普及促进了精确算法的研究和使用.

(6)增进人们对计算复杂性理论的理解,激发一些组合优化理论和算法设计方面的新成果诞生.

除了算法本身外,另一个同样重要的方面是复杂度分析技术.最新成果表明似乎没有必要设计过多规则,过去那种多达几十上百条规则的设计风格不但违背人思维习惯,也没有给复杂度带来更理想结果.这方面的重要进展是 Measure & Conquer 技术,它归功于 Fomin 和 Eppstein 等人的工作.本文中,我们将看到分支搜索技术(Branch & Reduce)、记忆(Memorization,实际是一种数据库技术)、动态规划等技术在精确算法设计中的应用,文末处还提供了这些精确算法的实测效果.我们尝试通过提供详细算法描述和示例来记述近几年的重要研究成果.

本文第 2 节介绍术语和一些理论成果;第 3 节介绍最小支配集、最大独立集、最小独立支配集、最小连通支配集和最小加权支配集问题的精确算法、设计思想、复杂度等;第 4 节列表总结重要算法,介绍一些技术发展趋势,并提供了算法实测效果;第 5 节结束全文.为保持文章主体层次清晰,我们将部分内容置于文尾附录中.

2 准备工作

2.1 符号及术语

$G=(V,E)$:G 代表图,V,E 分别表示该图的点和边集;$|S|$ 为集合 S 的势(cardinality),即 S 中非空元素个数,例如 $|V|=n$,$|E|=m$;$N(u)$ 为点 u 的开邻集,即所有与 u 之间存在一条边的点集合;$N[u]$ 为 u 的闭邻集,即 $N(u)\cup\{u\}$;$\Delta(G)$、$\delta(G)$ 分别代表图 G 的最大度和最小度,表示为 $\max_{u\in V}|N(u)|$ 和 $\min_{u\in V}|N(u)|$;$G[S]$ 为 G 中只保留集合 S 中的点及两端点都属于 S 的边,得到的诱导子图;$G-\{S\}$ 为 G 中去掉点集 S 及与 S 中任一点连接的边而得到的导出子图.通常用大写字母表示集合,用小写字母表示某一元素.

支配集 DS(Dominating Set)表示图 G 中点集 $DS\subseteq V$,满足对 $\forall u\in V$,要么 $u\in DS$ 要么 u 是 DS 中某点的邻节点,则称 DS 为支配集.$\gamma(G)$ 表示图 G

的$|DS|$.最小支配集 MDS(Minimum DS)表示图 G 所有DS中$|DS|$最小者.连通支配集 CDS(Connected DS)表示$G[DS]$导出图连通.最小连通支配集 $MCDS$(Minimum CDS)表示 CDS 中$|CDS|$最小者.独立集 IS(Indepdent Set)表示点集 $IS\subseteq V$,且 IS 中任一对点间不存在边.最大独立集 MIS(Maximum IS)表示 IS 中$|IS|$最大者.独立支配集 IDS(Independent DS)表示某点集 IDS 既是 IS 又是 DS.最小独立支配集 $MIDS$(Minimum IDS)表示 IDS 中$|IDS|$最小者.最小加权支配集 $MWDS$(Minimum Weighted DS)表示顶点具有权值时的 MDS;顶点(或边)覆盖是指给定一个顶点(或边)集合,使图中任一边(或点)都与该集合的某点(或边)相接,则称该集合为图 G 的一个顶点(或边)覆盖.O,O^* 为复杂度记号,后者省略了多项式因子,$O^*(c^n)=O(poly(n)\cdot c^n)$.

2.2　理论基础

DS 相关概念较早见于 19 世纪,那时人们研究棋盘上最少需要多少个王后才能攻击所有的方格,可要求王后之间互相不能攻击,以及符合条件的放置方法最大数量,用现代数学观点来看分别是集合覆盖、独立支配集和独立支配集最大数目的问题.

现代 DS 概念最早由 Claude Berge 在文献[4]中提出,Ore 在文献[5]中首次使用了支配集(dominating set)这个术语.Cockayne 在一篇综述文章[6]中首次使用 $\gamma(G)$ 来表示支配集的势($=|DS|$),这个记号已被广泛使用.Johnson[7] 在 1979 年表明了 DS 问题的 NP-完全性质(确切说文献[7]证明的是顶点覆盖问题,可转换为 DS 问题).

所谓 DS 问题即,给定一个图 G 和正整数 k,问题(P):G 是否存在一个 DS,使得$|DS|\leq k$.这个问题的性质涉及支配集算法设计中最重要的理论基础,我们有必要简单回顾一下.

定理 1.　Dominating Set 问题是 NP 完全问题[1,7].

证明.

(1)给定任意点集 $S\subseteq V$,$|S|\leq k$,验证 S 是否为 G 的一个支配集仅需多项时间,即 $P_{DS}\in NP$.

(2)已知 3-SAT 问题为 NP 完全问题(由 Karp 于 1972 年证明),图 1 将 3-SAT 变换为支配集问题(P_{DS}).

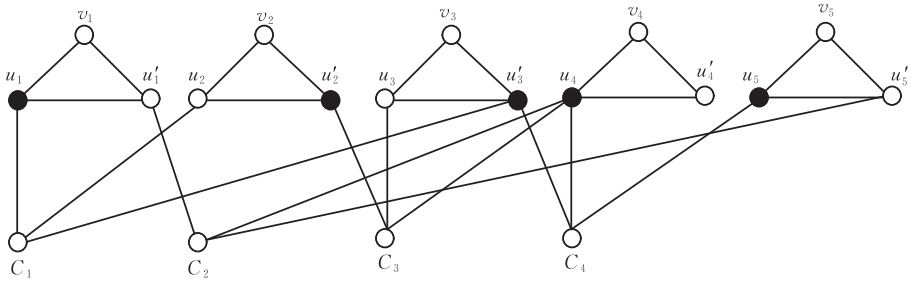


图 1　3-SAT 归约到支配集

给定变量集合 $U=\{u_1,u_2,\cdots,u_n\}$,u_i取 0 或 1;子句集合 $C=\{C_1,C_2,\cdots,C_m\}$,三元子句 $C_j=\{u_p,u_i',u_l\}$,其中 u_i'表示 u_i 取反,$p,l\in[1,n]$.对 U 中每个变量 u_i 建立一个三角形$\triangle u_iu_i'v_i$,如图 1 所示,如果 u_i 为真用黑色点表示,为假则将 u_i'用黑色点表示.对每个 C_j,如果含某 u_i 则在图中以边连接 u_iC_j(此图$k=n$).

(3)首先需说明,如果存在 C 为真的一种安排$\Rightarrow G$ 含支配集$|S|(=n)$,按如下规则构造集合 S:

如 u_i 为真则加入 S,反之将 u_i'加入 S.我们断言 S 一定是 G 的支配集,因为:① 每个三角形恰有一个点属于 S,三角形点被支配;② 根据假设 C 为真知:每个 C_j 必与某黑色点连接(支配).

(4)反过来我们需表明,如果 G 存在支配集$|S|\Rightarrow C$ 为真(即每个 C_j 为真),则

① 每个三角形恰含一个 v_i,且 v_i 已经被支配,因此支配集 S 的大小至少为 n,换句话说每个三角形内有一点属于 S,$|S|$至少为 n;② 按(3)步的方法构造 S,然后使与 C_j 相交的三条边中至少有一条连接黑色点,这种安排的结果必然使 C 为真.

(5)最后,图的构造方法表明,转换到 DS 算例只需建立 3-SAT 算例时间的常数倍.　　证毕.

Chang[8] 等人证明即使在偶图中,MDS 问题仍为 NP-完全问题.近似算法方面,对任意图 MDS、MCDS 问题来说,有任意常数 $\varepsilon>0$,使得除非 $NP\subseteq DTIME(n^{O(\log\log n)})$,否则在多项时间内不存在一个近似比(即近似精确解大小之比)小于$(1-\varepsilon)\ln n$ 的近似算法[9-10],对于 MIDS 问题,文献[11]表明除非 $P=NP$ 否则不存在近似比在 $n^{1-\varepsilon}$ 之内的算法.文献[12]表明,多项时间内 MIS 问题不存在近似比

低于 $N^{1-O(1)}$ 的近似算法.

下面列举了关于 $\gamma(G)$ 的主要定理.它们的证明方式几乎如出一辙:采用某种策略从 G 中选点加入集合 S,当 S 成为 DS 时计数其势的上界,以此估算 $\gamma(G)$,因而基本上是存在性定理,通常不易用来设计算法.凡事都有例外,第 3 节中第一个突破 2^n 界限的 MDS 算法就利用了定理 2 的性质.

定理 2[13]. 在含 n 个点、$\delta(G)\geq 3$ 的图中,存在一个 DS,其 $\gamma(G)\leq 3n/8$.

定理 3(Arnautov,1974). $\delta(G)=k$ 的图中,存在一个 DS,其 $\gamma(G)\leq\frac{1+\ln(k+1)}{k+1}\cdot n$.

定理 4[5]. 一个图如果没有孤立点则 $\gamma(G)\leq N/2$.

定理 5[13]. $\frac{n}{1+\Delta(G)}\leq\gamma(G)\leq n-\Delta(G)$.

定理 6[14]. 对于任意图,$\gamma(G)\leq\left(n+1-(\delta(G)-1)\frac{\Delta(G)}{\delta(G)}\right)/2$.

定理 7. 对任意 $n\geq 3$ 的连通图 G,$\gamma_c(G)\leq n-2$[15];$\frac{n}{\Delta(G)+1}\leq\gamma_c(G)\leq 2m-n$[16];当 $\delta(G)\geq 3$ 时 $\gamma_c(G)\leq 3n/4-2$,当 $\delta(G)\geq 4$ 时,$\gamma_c(G)\leq 3n/5-2$[17].

定理 8. 对任意连通图有 $\gamma_c(G)\leq n-\Delta(G)$[4];$\gamma_c(G)\leq 3\gamma(G)-2$[18].

2.3 关于 DS 历史

如 2.2 节记述,Berge 和 Ore 是目前学界公认引入现代 DS 概念的先驱.在文献[1]等众多文献中,西方学界认为更早的 DS 相关概念来自 19 世纪对国际象棋(Chess)里某些问题的讨论.按照这个逻辑,中国古代产生 DS 思想不比西方晚.诞生于约 2500 年前的围棋(Go),追求以最少的子支配尽可能多的面积,和最小支配集思想一致.围棋中任给一个图,判断某点是否为黑或白方支配,最坏情况需至终局才能肯定,尽管此时搜索范围不超过一个常数(围棋博弈树复杂度为 10^{360},状态空间复杂度为 10^{170}),但此常数比宇宙中已知原子数(10^{77},不计暗物质)还多得多;反观国际象棋,王后支配范围为确定的横竖斜,任给一个图,我们容易在多项时间内判断某点是否被攻击(国际象棋博弈树复杂度为 10^{123},状态空间复杂度为 10^{48});围棋将是人与 AI 角逐的最后阵地之一.另外我国古代军事学说常提到兵家必争之地,现代观点可看做求一个战场图的最小加权支配集,也包含 DS 思想;等等.

3 支配集构造技术

对于任意图,我们总能够通过枚举比较所有可能的点组合来获得任一支配集问题的精确解,由此导致平凡的 $O\left(\sum_{i=1}^{n}\binom{n}{i}=2^n-1\right)=O(2^n)$ 时间复杂度.

3.1 求最小支配集的 FKW 搜索算法

FKW 分别是文献[19]中 3 位作者名的开头字母,该算法用于计算图的 MDS.算法思想:利用 2.2 节定理 2.为此,需将原始图 G 改造成 $\delta(G')\geq 3$ 的图 G'.对于零度点只需加入 DS 集合即可;对于 1 度点 u,必然有一个 MDS 含 u 的邻节点 v 而不含 u;对于 2 度点,例如 u 来说,有下面的结论.

结论 1[19]. 设 $N(u)=\{v,w\}$,如果 MDS 是任意点集 $X\subseteq V$ 的最小支配集,则下述三者之一成立:① $v\in MDS,u\notin MDS$;② $u\in MDS,\{v,w\}\notin MDS$;③ $v\notin MDS,u\notin MDS,w\in MDS$.

当 0 度、1 度和 2 度点都被消去后,剩下的子图满足定理 2 的要求:$\delta(G')\geq 3$.此时通过枚举所有点组合,给出 G' 的最小支配集,进而得到原图的 MDS,即深度图理论(deep graph theory).

算法. $fkw(G,S,X)$.

输入:图 G、$S=\varnothing$、X(表示待支配的点集合,初始可令 $X=V$)

输出:一个 MDS

1. 如 $\exists u\in G$ 且 $d(u)=0$;如果 $u\in X$,返回 $fkw(G-\{u\},S\cup u,X\backslash u)$;否则,返回 $fkw(G-\{u\},S,X)$;

2. 如果 $\exists u\in G$ 且 $d(u)=1$,令 $N(u)=v$;
如果 $u\in X$,返回 $fkw(G-\{u,v\},S\cup v,X\backslash N[v])$;
否则,返回 $fkw(G-\{u\},S,X)$;

3. 如果 $\exists u\in G$ 且 $d(u)=2$,令 $N(u)=\{v,w\}$.如果 $u\notin X$ 转步 4,否则,转步 5;

4. 形成 3 个分支,返回:(1) $fkw(G-\{u,v\},S\cup v,X-N[v])$;(2) $fkw(G-\{u,v,w\},S\cup u,X-v-w)$;(3) $fkw(G-\{u\},S,X)$;

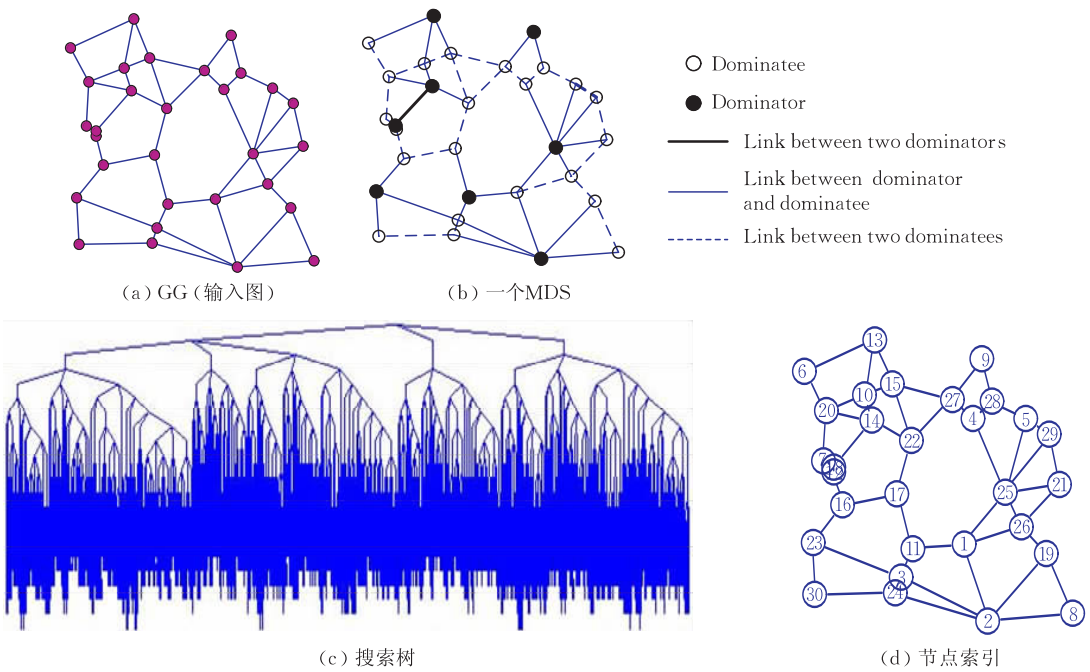
5. 形成 3 个分支,返回:(1) $fkw(G-\{u,v\},S\cup v,X-N[v])$;(2) $fkw(G-\{u,v,w\},S\cup u,X-u-v-w)$;(3) $fkw(G-\{u,w\},S\cup w,X-N[w])$;

6. 如果以上条件都不满足,则 $\delta(G)\geq 3$,枚举法求出并返回当前子图的最小支配集;

7. 比较所有分支结果,选择一个最小支配集 S 做最后的输出.

复杂性主要由 6 步决定，定理 2 表明最多只需枚举 $\binom{n'}{1}+\cdots+\binom{n'}{3N'/8}$ 次，利用 stirling 近似公式 $x!=\left(\frac{x}{e}\right)^x\sqrt{2\pi\cdot x}$，有 $O^*\left(\binom{n'}{3N'/8}\approx 1.93782^n\approx 2^{0.955n'}\right)$. n' 为子图点数，不超过 n，算法突破了 $O(2^n)$ 界限.

图 2 是该算法示例，其中，图 2(a)为初始输入图，图 2(b)显示一个 *MDS*. 人们形象地称此类算法为搜索树(Search Tree)或分支简化(Branch & Reduce，以下简记 B&R)算法，见图 2(c). 当子问题只含一种情况时，称为简化规则(reduce rules)，例如 *fkw* 算法的步 1、步 2，搜索树不产生分支；如果子问题含多种情况，称此时采取的策略为分支规则(branch rules)，例如算法的步 4、步 5，搜索树产生至少两个分支. 图 2(c)中我们让程序描绘算法每一个简化点和分支点，得到一个搜索过程树，树中含 13493 个叶结点. 算法在每个叶子所代表的子图里执行枚举(步 6). 输入节点编号见图 2(d).



(a) GG(输入图)　(b) 一个MDS　(c) 搜索树　(d) 节点索引

图 2　FKW 算法示例

B&R 是 Davis 和 Putnam[20] 在 40 多年前开发的技术，是解决 NP 难题最常用的方法. 其核心思想是应用一系列简化规则将问题分解为小规模的子问题，然后递归地在子问题上应用同样规则直到问题分解为基问题(base case). 尽管最早突破 $O(2^n)$ 界限，*fkw* 仍难令人满意，如果图中无低于 2 度的点，算法又回到平凡的枚举. 下面的 Grandoni 算法是目前最好的 MDS 算法之一.

3.2　求最小支配集的 Grandoni 算法

原算法[21] 给出的是 $|MDS|$ 而非具体 *MDS*，本文稍做改动，给出具体 *MDS*. 算法的复杂度[22] 为多项空间 $O^*(2^{0.61n})$，指数空间为 $O^*(2^{0.598N})$. 用自然语言描述算法如下.

算法.　*Grandoni*(S,S^*).

输入：初始化 $S=\{N[v]\mid v\in V\}$，$S^*=\varnothing$

输出：S^*(图 G 的 *MDS*)

1. $|S|=0$，则返回 S^*；

2. 当前 S 中寻找子集 $S_i\subseteq S_j$，返回 *Grandoni*$(S\backslash S_i, S^*)$；

3. 寻找 $u\in S_i$，要求 u 的出现频率为 1. 返回 *Grandoni*$(S-(S_i), S^*\cup S_i)$；

4. 寻找 S 中具有最大势的 S_i，判断：如果 $|S_i|\leq 2$，调用最大匹配算法直接给出 S^*；//多项时间

否则，返回 $\min\{$ *Grandoni*$(S\backslash S_i, S^*)$, *Grandoni*$(S-(S_i), S^*\cup S_i)\}$；

说明，$S-(*)$表示从每个子集 S_i 中都去除元素 $\{*\}$，$S\backslash S_i$ 表示仅从 S 中删除子集 S_i. 步 4 当 $|S_i|\leq 2$ 时，需要构造一个图 G'，这个图的点集为 $U=\{u\in S_i\mid S_i\in S\}$，如果某 $S_i=(u,v)$，则 uv 为 G' 的边. 求出该图的最大匹配 M 后，每个 M 中的边对应一个 S_i，再将孤立的点所对应子集加入 S^*. 任意图最大匹配问题多项时间算法[23] 最早由 Edmond(1964)提出，当时复杂度为 $O(n^4)$，后被改进到 $O(n^3)$.

Grandoni 算法获得优势原因有三:(1)将 MDS 问题转化为集合覆盖问题(初始化).(2)特定条件下覆盖问题转换为最大匹配在多项时间内解决.(3)应用 Measure and Conquer(M&C)复杂度分析技术.

B&R 技术广泛应用后的相当长一段时间内(20 世纪 70、80 年代),人们倾向于设计更多更复杂的规则,其数量动辄几十上百,然而用来度量规则所产生子问题规模的测度却非常简单,例如当前图点数,我们称此类测度为常规测度."M&C"思想是用更复杂的非常规测度来衡量每一个分支规则的影响.做个比喻:假设身高体重是健康的常规测度,1.75m 体重 85kg 算健康,然而这个测度只说明他大致健康,健康到何种程度,是优良中还是及格?还需做更细致的检测分析,包括测量肌肉重量和脂肪重量、腰部重量和臀部重量、肢体重量、器官重量,等等,它们之间的比率要适当才算健康,一个人如果细胳膊细腿却粗身子,就算 85kg 也枉然.在得到这些更细致的"重量"后,把它们作为一个整体系统中的子系统而分别赋予一个"恰当"的权值,求和后就得到一个衡量健康的更精确参数,使最终结论更接近(而非到达!)真实健康值.设计更多规则不一定导致算法更快的道理容易理解,例如用不规则开口桶来盛水,水的高度不会超过桶的最低处,同样 $O(3^n,2^n)=O(3^n)$,算法复杂度取决于最慢规则的复杂度.过多(分支)规则有时反而增加复杂性,例如导致巨大的隐藏系数.

搜索树复杂度常规分析技术使用递归方程(recurrence equations)的特征多项式.以本节算法为例,用 $L(k)$ 表示算法分支产生的叶子节点数,U 表示子集中待覆盖的点,常规测度 $k=|S|+|U|$.算法只在步 4 当 $|S_i|>2$ 时产生两个分支,第 1 个分支 S^* 不含 S_i,则 $S-S_i$,此时 $k'=k-1$;第 2 个分支 $|S_i|$ 至少为 3,$k''\leq|S|-1+|U|-3$;所以有 $L(k)\leq L(k-1)+L(k-4)$,取方程 $x^4-x^3-1=0$ 的最大正根后有 $2^{0.465(|S|+|U|)}=2^{0.93n}$,获得最坏情况下的一个上界.如果有多个这样的方程,取所有解中的最大正实根.

当去掉的 $|S_i|$ 较大时算例规模下降较快,反之下降较慢,常规测度(权值都为 1)没有刻画这些更细致的特征.M&C 针对 $|S_i|$ 的不同,权重化测度为 $k=\sum_{i\geq1}\alpha_i|S|+\sum_{j\geq1}\beta_j|U|$,$\alpha_i,\beta_j\in[0,1]$.还可进一步对 α_i,β_j 做一些假设以简化分析过程(例如限制 $i,j\geq6$ 时 $\alpha_i,\beta_j=1$ 等),最终得到形如 $x^k=x^{k-\Delta i}+x^{k-\Delta j}$ 的方程,其中 $\Delta i\Delta j$ 分别为含 α_i,β_j 的函数.这是一个多元递归方程,这种方程可以同时有很多个,求解它们可利用 Eppstein、Amenta 等人关于 quasi-convex programming 的工作[24],伪凸规划是广义的线性规划,多元变量被看做是一个关于 $|S|$ 等参数线性组合的单变量,求得解与真实值只差一个多项系数.M&C 在选定一个初始向量 w 及允许误差后不断微调其值(类似多重梯度下降),然后利用伪凸规划求解方程,最后得到原算法时间的一个(非紧致)上界,更细致的描述请参考原文献.利用 M&C 重新计算 Grandoni 算法复杂度为多项空间内 $O(2^{0.305(|S|+|U|)}=2^{0.61n})$,与前段常规分析结果 $O(2^{0.93n})$ 相比进步显著,而算法本身没有任何变动,完全得益于分析技术的进步!

Grandoni 算法还可进一步改进.Robson 介绍了 Memorization 技术[25]用于精确算法设计,核心思想是以空间换时间:某子问题已被解决,将结果 $(G',algo.(G'))$ 存入库,其中 $algo.()$ 代表算法计算结果;此后算法在处理任何子图问题前,先检查库中是否已包含该算例 G',如果有则直接取出结果.原始 G 的子图最多有 2^n 个,排序后查找某子图只需对数时间,故在线性 $O(n)$ 时间内我们可找到某子图(或者知道库中不含此子图).

让我们大致观察一下这种做法对复杂度的影响,以前面例子 $L(n)\leq L(n-1)+L(n-4)$ 示意说明:$L_h(n)$ 表示当前子问题的数量,并且这些子问题的 h 点导出图已在前面算法过程中被解决.有 $L_h(n)\leq2^{0.465(n-h)}$,G 的 h 导出图共 $\binom{n}{h}$ 个,由于任何子图最多被算法计算一次,有 $L_h(n)\leq\min\left\{2^{0.465(n-h)},\binom{n}{h}\right\}$;对 $\binom{n}{h}$ 应用 stirling 近似公式,并注意当 n 较大时 $\sqrt[n]{cn^{-1/2}}\to1$,c 为某正常数.$2^{0.465(1-\alpha)}\approx\frac{1}{\alpha^\alpha(1-\alpha)^{1-\alpha}}$,$h=\alpha n$,求此方程零点,可得 $\alpha=0.0865$,所以最终复杂度为 $O\times(2^{2\times0.465(1-\alpha)n}=2^{0.85n})$,小于前面常规方法得到的 $O(2^{0.93n})$.更细致的分析[22]表明 Grandoni 算法应用 Memorization 后复杂度为 $O\times(2^{0.598n})$.

3.3 求全部支配集的 LMDS(Listing Minimal Dominating Sets)算法

很多时候人们希望知道一个图的所有 *MDS*,文

献[26]之前未见除穷举外其他算法. LMDS 计算所有的最小支配集，同前面 3.2 节算法一样也是先将问题转换为集合覆盖. 设计算法规则时遵循：如果确定集合 S_i 对覆盖的贡献可由其它集合替代，那么不让 S_i 出现在解集合中；换个方式表达即，如果集合 S 是图中所有点的最小覆盖且子集 $S_i \in S$，则 S_i 至少含一元素 u 不属于 S 中任何其它子集. 算法把符合这样规则的所有集合都搜索出来. 文献[26]中没有显式给出算法，我们将其整理如下.

说明：算法 *lmds* 步 10 依据的是如果 $S_1 S_2 \in S^*$，且 $S_1 \backslash S_2 = u$，$S_2 \backslash S_1 = v$，则其余 $S_i \in S^*$ 既不能含 u 也不能含 v. 假如含 u，则可从 S^* 中去除 S_1，违背了前面提到的设计规则，步 9 依据同理. 算法复杂度为 $O^*(2^{0.8235n})$. 附录中的附表 1 根据算法 *lmds* 列出了图 2(a)的所有最小支配集，共 56 个.

算法 1.　*lmds*(U,S,S^*).

输入：初始化 $U=V$，$S=N[v](v\in V)$，$S^*=\varnothing$

输出：S^*(图 G 的所有 *MDS*)

0.(暂时为空)

1. U 或 S 为空则返回 S^*；

2. 寻找 $u\in U$，要求 u 在集合 S 中出现的频率为 1. 设 $u\in S_i$，返回 *lmds*$(U-\{S_i\},S-(S_i),S^*\cup S_i)$；

3. 寻找 $u\in U$，要求 $\{u\}=S_1=S_2=\cdots=S_r$，$r\geq 2$，且 u 仅属于这些集合. 算法在这里产生 r 个分支：

令 $S=S\backslash(S_i)$，其中 i 遍取 $1,2,\cdots,r$；然后依次返回 *lmds*$(U-\{u\},S,S^*\cup S_i)$；

4. 寻找 $|S_i|\geq 5$. 设 $S_i=\{u_1,u_2,\cdots,u_r\}$，产生两个分支，返回：

(1) *lmds*$(U,S\backslash S_i,S^*)$；(2) *lmds*$(U-\{S_i\},S-(S_i),S^*\cup S_i)$；

5. 寻找 $|S_i|=4$. 处理方式类同步 4；

6. 寻找 $u\in U$，要求 u 在集合 S 中出现的频率为 2. 设 S_i,S_j 包含 u，且 $|S_j|\geq|S_i|$，判断：

如果 $|S_i|=1$，返回：(1) *lmds*$(U-\{S_i\},(S\backslash S_i)\backslash S_j,S^*\cup S_i)$；(2) *lmds*$(U-\{S_j\},(S\backslash S_i)-(S_j),S^*\cup S_j)$；

如果 $|S_i|\geq 2$，接着判断，如果 $S_i\subseteq S_j$，返回两分支，

(1) *lmds*$(U-\{S_i\},(S\backslash S_j)-(S_i),S^*\cup S_i)$；(2) *lmds*$(U-\{S_j\},(S\backslash S_i)-(S_j),S^*\cup S_j)$；

否则，产生 3 个分支，返回：(1) *lmds*$(U-\{S_i\},(S\backslash S_j)-(S_i),S^*\cup S_i)$；(2) *lmds*$(U-\{S_j\},(S\backslash S_i)-(S_j),S^*\cup S_j)$；(3) *lmds*$(U-\{S_iS_j\},S-(S_iS_j),S^*\cup S_i\cup S_j)$；

7. 寻找 $|S_i|=3$. 处理方式类似步 4；

8. 寻找 $S_i\subseteq S_j$. 返回两分支：(1) *lmds*$(U,S\backslash S_j,S^*)$；(2) *lmds*$(U-\{S_j\},S-(S_j),S^*\cup S_j)$；

9. 寻找 $u\in U$，要求 u 在集合 S 中出现的频率为 3. 设 S_i 包含 $\{u,u_i\}$，$i=1,2,3$，产生下面 7 个分支：

(1～3) *lmds*$(U-\{S_i\},S-(S_i),S^*\cup S_i)$；//共 3 个分支

(4～6) *lmds*$(U-\{$两 $S_i\}$、$S\backslash$所有含两 u_i 之一子集，$S^*\cup($两 $S_i))$；//共 3 个分支…

//两 S_i 表示下标 i 任取两个数组合. 从 S 中除去含任一 u_i 的子集.

(7) *lmds*$(U-\{S_i,i=1,2,3\},S-(S_i,i=1,2,3),S^*\cup(S_i,i=1,2,3))$；//一个分支

10. 寻找 $|S_i|=2$. 设 $S_i=\{u,v\}$，用 S_uS_v 表示除 S_i 外其它分别包含 u,v 的集合. 产生 3 个分支：

(1) *lmds*$(U,S\backslash S_i,S^*)$；(2) *lmds*$(U-\{S_i\},S\backslash S_i\backslash S_u,S^*\cup S_i)$；(3) *lmds*$(U-\{S_i\},S\backslash S_i\backslash S_v,S^*\cup S_i)$；

11. 算法停后将具有最小势的集合覆盖 S^* 转化为最小支配集表.

在基本算法基础上，本文进一步优化该算法，思路很简单，将算法 *lmds* 的步 0 变为：如果待搜索分支的 $|S^*|\geq|$当前最小覆盖$|$，标记该分支搜索结束，$|$当前最小覆盖$|$初始化为无穷大，并随搜索过程动态更新. 这样操作对复杂度没有影响，因为最坏情况下可能首先搜索到势最大的解集，或者所有解集同势. 然而通常情况下，这样做将为算法节省指数级的时间，因为我们总将子集加入覆盖集合(递增)，一旦搜索出某覆盖集合的势为 k，则从势 $k+1\sim n$ 的分支不必继续搜索. 本文对图 2 算例进行实测，添加步 0 使运行时间减少近一个数量级. 前节 Grandoni 算法、本文中介绍的其它算法适当处理后都可添加此步(以下不再重复). 敏锐的阅者立即意识到，这里使用的是 α-β 剪枝技术(Knuth and Moore, 1975).

算法 *lmds* 规则复杂，该算法进化版本 *ls* 在文献[27]中提出，见附录 2. *ls* 去掉 *lmds* 中的一些规则，并且每次选择具有最大势的子集来产生分支，复杂度降至 $O^*(2^{0.78n})$，针对图 2 的具体算例，实测结果比 *lmds* 的快一倍多.

3.4　最大独立集 MIS 精确算法

有了支配集基本概念后，人们尝试给 *DS* 加上一些属性限制，于是当属性为 $G[DS]$——不含边、连通、完全图、包含汉密尔顿圈、权重最小时，对应 IDS、CDS、Dominating Clique、Hamilton Cycle DS、MWDS，另外还包括 DN(Domatic Number，即图 G 是否可分为 k 个顶点两两互不相交的极小支配集——其真子集不是支配集)等，我们统称这些为条件支配集. 类似地，属性也可以用来限制点集$\{V-DS\}$，例如要求每个 dominatee 至少被支配 k 次以满足容错需求. 多个属性可以组合，关于这方面的更多内容，我们推荐参考文献[1]. 本节介绍的 *MIS* 相当于给 *DS* 赋予属性：$|DS|$ 最大且 $G[DS]$ 不含边.

我们说 MIS 一定为 DS 是由于：如果点 u 未被支配，则 u 与 MIS 中点形成 IS，$MIS\cup u$ 是一个比 MIS 更大的 IS，这与 MIS 是最大独立集矛盾.

MIS 算法研究历史悠久：Tarjan 等在 1977 年提出 $O^*(2^{0.3333n})$ MIS 精确算法[3]；Tang Jian[28] 在 1986 年将算法改进到 $O^*(2^{0.3039n})$；同年 Robson[25] 发表了两个算法分别使用多项、指数空间，对应时间复杂度为 $O^*(2^{0.292n})$ 和 $O^*(2^{0.276n})$；上世纪最快算法由 Beigel 获得[29]，为 $O^*(2^{0.29n})$ 及多项空间. 本节要介绍的文献[30]中算法复杂度为 $O^*(1.221^n=2^{0.288n})$ 是目前多项空间内最简单、最快的——其优势获得原因有三：一是应用了两个重要规则“镜像”和“折叠”；二是得益于 M&C 分析技术；三是应用了 M&C 设计思想.

镜像(mirror). 设某点 v 有一个 2 跳邻节点 u，令点集 $S=N(v)-N(u)$，如果 $G[S]$ 为完全图或空，则称 u 为 v 的镜像. u 可能不只一个，此时用 $M(v)$ 表示 v 的镜像点集合.

镜像性质. 如果某 G 的所有 MIS 不含 v，则 MIS 也不含 $M(v)$. 原因如下：设所有 MIS 不含 v，则 $d(v)\geq 2$(否则 v 为零度或 1 度点，此时必有一个 MIS 含 v)，且 $N(v)$ 中至少两个邻点属于 MIS(否则假设 $N(v)$ 中 $z\in MIS$，v 替换 z 后 $|MIS|$ 不变)；由于 $G[S]$ 为完全图，最多只能含一个 MIS 中点，所以定有一点 $w\in S'=N(v)\cap N(u)$ 属于 MIS，于是我们肯定 $u\notin MIS$. 同理 $M(v)\notin MIS$.

折叠(folding). 果 $N(v)$ 中不含势为 3 的独立集，则称 v 是可折叠的.

设 $u_iu_j\cdots$ 构成 $N(v)$，折叠操作由 4 个步骤组成：① 如果任意一对 (u_iu_j) 之间不存在边，产生一个新点 u_{ij}；② 每对 u_{ij} 之间添加一条边(只有一个 u_{ij} 则不添边)；③ 在每个 $\{(N(u_i)\cup N(u_j))\backslash v\}$ 点与 u_{ij} 之间添加一条边；④ $G-N[v]$.

下面给出的 mis 算法是文献[30]作者修改后的版本，删除了文献[30]中一些复杂的判断条件. mis 复杂度为 $O^*(2^{0.287N})$，我们进而做稍许改动使其给出一个具体 MIS(原算法 mis 计算的是 $|MIS|$).

算法 2. $mis(G)$.

输入：图 $G=(V,E)$

输出：一个最大独立集 MIS

1. 如果 G 中点数 $|V|\leq 1$，返回 $\{V\}$；

2. 如果 G 中有一个独立连通组元 C，返回 $mis(C)\cup mis(G-C)$；

3. 如果 $\exists v,w$ 满足 $N[w]\subseteq N(v)$，返回 $mis(G-\{v\})$；

4. 如 $\exists d(v)=2$；如果 $u_{ij}\notin mis(\hat{G})$，返回 $mis(\hat{G})\cup v$；否则返回 $mis(\hat{G})\backslash u_{ij}\cup N(v)$；

5. 选择 G 中具有最大度的点 v，返回 $\max\{mis(G-v-M(v)),mis(G-N[v])\}$.

算法经过前 3 步后在步 4 折叠时，v 只有一种形式，见图 3. 不难看出折叠后的图 $\hat{G}$ 恰比原图 G 的 $|MIS|$ 小 1.

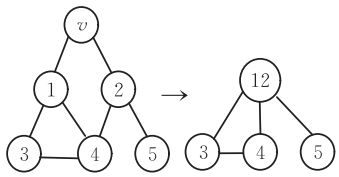


图 3 折叠 v

M&C 原理表明，使子图规模下降快的规则对降低精确算法(更确切说是 B&R 算法)复杂度，从长远看，是有利的. 因此 mis 算法步 5 从当前图中选取最大度点产生两个分支：其一含不含 v 点，利用镜像性质知，如果解集不含 v 则也不含 $M(v)$，故 $mis(G-v-M(v))$；其二包含 v 点，根据独立集的特性，如果解集含 v 则一定不含 $N(v)$，故 $mis(G-N[v])$. 这种设计思路在 Grandoni 算法和 ls 算法中广泛使用，取得很好效果，我们认为可以作为今后算法设计的规则而确定下来.

mis 算法简洁优雅，与 20 世纪 70、80 年代十几页近百规则的算法相比，几乎完胜. 计算示例在图 4 中.

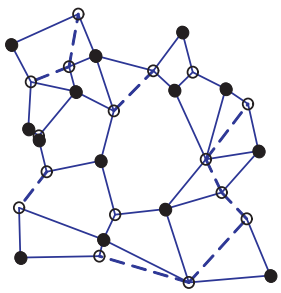


图 4 一个最大独立集

3.5 最小连通支配集 MCDS 算法

文献[7]表明 MCDS 问题的 NP 难性质. CDS 问题等价于寻找图 G 的一个生成树，要求该树的叶子节点数达到最大. 问题的难点在于连通的 $G[S]$ 需要全局知识，类似这样需全局知识的有著名的 TSP 问题(最小代价遍历完全图所有节点)和 Steiner 最小树问题(连接 $\{V\text{-}Steiner\}$ 点的最小代价树). 以前提到的算法规则无法直接应用到 CDS 问题中；在文献[31]之前没有算法突破 $O(2^n)$ 平凡界限.

文献[31]中 MCDS 算法的设计思想：猜测某条边可能属于 MCDS，这样的猜测最多有 $O(n^2)$ 个，因

此算法主体必须要运行同样次数. 假设计算某次猜测需指数级别复杂度 $O^*(\cdot)$，重复多项次其复杂度仍将是 $O^*(\cdot)$级别，至此，只要在实现猜测过程中应用某些规则来替换穷举，就能突破 $O(2^n)$界限，“规则”以始终保持导出图 $G[S]$连通为准来设计.

文献[31]介绍的算法较复杂，列于附录 3 中，复杂度为 $O^*(2^{0.9566N})$，计算示例见图 5.

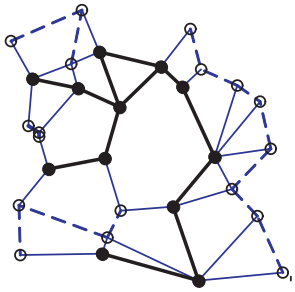


图 5　一个最小连通支配集

尽管突破 $O(2^n)$界限且是目前最快的，该 MCDS 算法并不实用(参看本文第 4 节的数据)，其中原因值得深入思考. 无线移动网络中，连通支配集在广播、数据融合、安全等方面有着特殊重要的应用价值. 构造 CDS 所采用近似算法大都是先建立一个独立支配集或极大独立集，而后再连接成 CDS.

3.6　最小独立支配集 MIDS

Karp 在 1972 年证明独立集问题是 NP-完全性质的. 文献[32]所介绍的 MIDS 算法(参考附录 4)思想新颖之处在于利用了一种称做标记图的性质.

标记图(marked graph). 图的节点被分为 F、M 两类，分别代表自由、标记节点. 最初所有节点都是自由节点，当算法产生分支时，例如某 2 度点 u 的两个邻节点为 v_1，v_2，产生三个分支，解集 S 依次含 u，v_1，v_2，那么在第 3 个分支时，我们可以认为 S 中不含 v_1，因为在第 2 个分支的解集 S 中已经考虑了含 v_1的情况(第 2 分支解集仍可能含 v_2)，因而在第 3 分支中，标记 v_1点，将其加入 M 集合. 标记某点相当于前面已经考虑了可能含该点的情况，这种“记忆”随即被传递给后面分支.

图的 F 点集导出子图 $G[F]$可能含多个连通组元 C，一些连通组元的特殊性质可以加以利用：当 C 为团(clique)时，显然任何 MIDS 恰含 1 个 C 中元素；当 C 构成完全二分图($C=X+Y$，且点集 X 的每个点都连接点集 Y 的所有点)时，算法产生两个分支分别含 X，Y(最终返回分支势小者). 当前面条件都不满足时，算法选择一个具有最小 F 度的点 u 产生分支.

尽管规则略多，却都比较简单，容易望文知义，最后得到 $|MIDS|=8$. 只需稍加改造就可直接输出一个具体 $MIDS$，见图 6. 利用 M&C 技术分析算法复杂度为 $O^*(2^{0.441n})$.

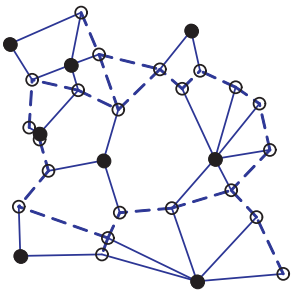


图 6　一个最小独立支配集

3.7　最小加权支配集 MWDS 算法

某地区需要布设信号发射设备，问怎样布局才能以最低代价让信号覆盖整个区域？由于在不同地点架设成本不一样，此时问题归结为寻找一个地区图的 $MWDS$.

前面 3.3 节算法 $lmds$(及 ls)可改造用于计算 $MWDS$，回忆在算法 $lmds$ 中，所有能成为 DS 且每个 S_i都不可能被其它子集替代的解集被一个不落的搜索出来，这样，只要去掉步 0，或将其判断条件改为当前最小权重，就能给出图 G 的所有 $MWDS$，复杂度不变. 假设图 2(a)中点权重为 $V\to R^+$：$weight=2^{d(v)}$(2 的该点度次方)，计算结果见图 7($weight=64$). 如果只需要一个 $MWDS$，则算法复杂度可进一步降低至 $O^*(2^{0.658n})$，这是因为当所有 $|S_i|\le 2$ 时，由这些 S_i可构造一个新图 G'(方法见 3.2 节算法第 4 步的说明)，G'是二分图，将 G'转换为完全二分图后(用匈牙利算法)返回其最小加权完美匹配，这只要多项时间即可.

目前最快 MWDS 算法来自文献[33]，用一句话概括算法思想：结合 B&R 技术与动态规划技术的优势. 原因：当前图中点度数较大时利用 B&R 产生分支将迅速降低子问题规模，一旦子问题规模降到较小阶段(图的树宽较小时)，继续分支规则将使问题复杂化，此时动态规划技术表现良好.

限于篇幅这里仅概述算法流程：类似 $lmds$ 将最小支配集问题转换为集合的最小覆盖问题，建立相同的数据结构⇒如果 $|S|=0$ 则返回空⇒某 $u\in S_i$ 频率为 1，将 S_i加入解集；⇒所有 $|S_i|\le 3$，由剩余子集 S_i及其中元素构造一个新图 G_S，执行路径分解后利用动态规划技术计算并返回该二分图的所有最小加权支配集；⇒选择一个最大的 $|S_i|$，解集中

要么含 S_i 要么不含，产生两个分支，返回二者中权值最小的. 最终算法给出所有的 *MWDS*，复杂度为 $O^*(2^{0.6355n})$，计算结果与 *lmds* 的一致，见图 7.

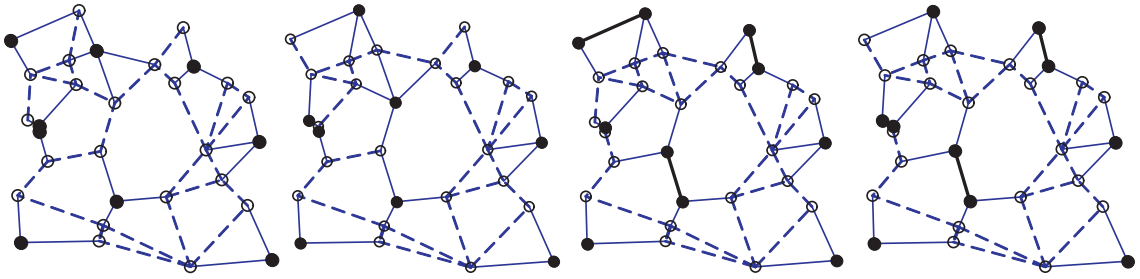


图 7 图 2(a)的所有最小加权支配集

4 小结与展望

表 1 列出常被引用及最新(截至 2010 年)的 DS 相关算法. 其中，文献[34]于 2009 年提供了目前最快的 MDS 算法，比本文介绍的 Grandoni 算法复杂度降低一个几乎可以忽略的值，但其规则极为复杂. 表 1 中最后一列是编程复杂度，分简单、中等、复杂三个等级，由于没有公认衡量标准，我们以算法规则数量来评价，仅供参考.

目前寻求包括 MDS 等在内的 NP 完全或 NP 难题精确解方法以 B&R 为主，此外常见的还包括动态规划(dynamic programming)、预处理技术(preprocessing the data)、局部搜索技术(local search，从某个可行解出发，过渡到较近的另一个可行解，最终遍历整个可行域空间，这种邻近通常会随具体问题而不同，例如 Hamming 距离)；另外还有一种策略即随机算法(randomization)，它的思路是随机选择问题的某些参数或随机选择节点加入的顺序. 这样无疑会导致某些时候走在错误的方向上，因此同样的算法需重复多次以使错误概率减小(重复足够次时，成功的概率接近 1)，确切地说，假设完成每次选择需时间 t，成功的概率为 p，为使算法最终出错的概率不超过 e^{-c}($\geq(1-p)^{c/p}$)，需要重复 $c\cdot t\cdot p^{-1}$ 次. 这方面有大量文章是关于 k-可满足性等问题的，虽然我们介绍的是支配集问题，但文章开篇表明我们可以将支配集转换为 3-SAT. 怎样进行随机操作可使错误的概率降低也是一个很令人感兴趣的研究课题，这方面的综述可参考文献[40-41]等. 再有就是我们现在常见到的那些著名智能优化计算方法，如遗传算法、人工神经网络等等，这些方法在求解 TSP 等 NP 难问题方面都有令人信服的表现. B&R 方向上，最近有人尝试用计算机来自动地、机械化地发现规则，这样规则数目将达万记，如何使用这些规则、编程等都可作为进一步的研究方向. 最后值得一提的是随着多核心计算能力的普及，并行算法的重要性日益突出，B&R 算法树状结构天然适合多线程多处理器的执行方式，随着核心数量的增多，计算时间线性降低.

本文介绍的是 DS 算法，稍加变化就可用于其它 NP 完全或 NP 难问题. 例如在任意图中寻找最大团(clique)也是 NP 完全问题，3.4 节 MIS 算法不需任何改变就可用来求图的最大团，只需将图 G 转换为补图 $\overline{G}$，然后求 $\overline{G}$ 的最大独立集；这个问题还等价于补图的最小顶点覆盖. 细心观察，这样的例子将非常多.

精确算法研究可为近似算法设计提供理论指导，至少可以避免浪费精力去追求一些理论上已经证明不可能达到的近似度(除非在 $P=NP$ 这个关键问题上有突破). 关于 DS 问题近似算法有两点情况值得重视：一是贪心策略实际效果良好，所得某类支配集 $|XDS|$ 值接近精确解的值；二是近似算法所得结果与精确结果相比的实测平均近似度远好于理论近似度，这种差异产生的原因尚未明了.

最后，我们对文中介绍的算法进行了简单测试，考察精确算法解决问题规模，即图顶点数. 实验在一台 Dell 家用笔记本电脑上进行，配置为：CPU，Intel Yonah 1.6GHz；内存，2.5GB DDR2 667；操作系统为 Windows XP. 实验结果见表 2.

表 2 中，我们限制算法运行不超过一天的工作时间，输入为无线网络应用领域所常用的任意无向图 UDG(应用广泛)、GG、DTG、MST 等[42]，为节约空间只列出最终统计结果.

表 1　一些重要的最小支配集及相关问题精确算法列表

Problem	Reference	Complexity		Complexity analysis technique	Complexity for programming
		time	space		
MDS	Fomin, 2004, [19]	$O^*(2^{0.955n})$	polynomial	Stirling's approx.	simple
MDS(all)	Fomin, 2005, [26]	$O^*(2^{0.8235n})$	polynomial	M&C	moderate
MDS	Grandoni,2006, [21] Fomin, 2005, [22]	$O^*(2^{0.61n})$ $O^*(2^{0.598n})$	polynomial exponential	M&C	simple
MDS	Johan,2009, [34]	$O^*(2^{0.59n})$	polynomial	M&C	complicated
MIS	Moon, 1965, [35]	$O^*(2^{0.5283n})$	polynomial	Recurrence equat.	moderate
MIS	Tarjan, 1977, [3]	$O^*(2^{0.334n})$	polynomial	Recurrence equation	complicated
MIS	Tang Jian. 1986, [28]	$O^*(2^{0.304n})$	polynomial	Recurrence equat.	moderate
MIS	Robson, 1986, [25]	$O^*(2^{0.296n})$ $O^*(2^{0.276n})$	polynomial exponential	Recurrence equations	complicated
MIS	Beigel, 1999, [29]	$O^*(2^{0.29n})$	polynomial	Recurrence equat.	moderate
MIS	Fomin, 2006, [30]	$O^*(2^{0.288n})$	polynomial	M&C	simple
MIS	Furer, 2006, [36]	$O^*(2^{0.3063m-n})$	polynomial	Measure m-n	simple
MCDS	Fomin, 2006, [31]	$O^*(2^{0.9566n})$	polynomial	M&C	moderate
MIDS	Gaspers, 2006, [32]	$O^*(2^{0.441n})$ $\Omega^*(2^{0.4057n})$	polynomial	M&C	moderate
MWDS	Fomin, 2005, [26]	$O^*(2^{0.658n})$	polynomial	M&C	complicated
MWDS(all)	Fomin, 2007, [33]	$O^*(2^{0.6355n})$	polynomial	M&C	complicated
the number of maximum weighted IS	Dahllöf, 2002, [37]	$O^*(2^{0.4057n})$	polynomial	Recurrence equation	complicated
Min Dominating Clique	Kratsch, 2006, [38]	$O^*(2^{0.4212n})$ $\Omega^*(2^{0.3334n})$	polynomial	M&C	moderate
Exis. Dominating Clique		$O^*(2^{0.35n})$	polynomial	M&C	simple
Max Dominating Clique	Bourgeois, 2009, [39]	$O^*(2^{0.4n})$, tight	polynomial	M&C	simple
Min Dominat. Clique		$O^*(2^{0.41n})$	polynomial	M&C	complicated

表 2　可解决问题规模

问题种类及算法	8 小时内可解决问题规模(约)	平凡枚举法
MDS[21]	200 点	根据问题性质不同，计算 30 点图约需 1～3 天时间
LS[27]	100 点	
MIS[30]	300 多点	
MCDS[31]	不到 50 点	
MIDS[32]	不到 200 点	
MWDS[27]	100 点	

关于表 2 做两点说明：(1) 问题的性质依赖输入图的类型. 对于某些问题，当输入是 MST(最小生成树)时，问题性质可能退化为一个多项时间内就能解决的问题，例如 MDS 算法只产生简化点，不产生分支点，表 2 最终统计结果中排除了此类现象；(2) 限定时间内可解决问题规模严重依赖于图的平均邻节点度，过大(接近完全图)或过小(接近树)实际上都导致可解决问题规模迅速增加，远超表 2 中值，此类结果被排除.

参考表 2，同为 NP 难题，为什么有的问题可以获得较快的算法，而有的却不能？值得思考.

在可预见时期内，我们难以指望摩尔定律对解决问题提供多大帮助，追寻更快的算法、更好的复杂度分析技术、组合优化理论将是未来一段时期内的主要研究方向.

5　后　记

文中提供的算法经过了校验和测试. 就作者所知，迄今(至 2010 年)，文中介绍的算法仍是同类问题算法中最快的之一[43]. 附录 5 提供例图(图 2(a))点坐标及生成方法，读者可自行检验.

限于水平，我们的理解难免存在谬误，欢迎指出，我们将及时改正.

致　谢　挪威 University of Bergen 的 Fedor V. Fomin 教授和 Alexey A. Stepanov 博士的耐心帮助使本文工作得以进行，谨致敬意并衷心感谢！

参　考　文　献

[1] Haynes T W, Hedetniemi S T, Slater P J. Fundamentals of Domination in Graphs. New York: Marcel Dekker Inc., 1998

[2] Held M, Karp R M. A dynamic programming approach to sequencing problems. Journal of SIAM, 1962, 10(1): 196-210

[3] Tarjan R, Trojanowski A. Finding a maximum independent

set. SIAM Journal on Computing, 1977, 6(3): 537-546
[4] Claude Berge. The Theory of Graphs and Its Applications. New York: Methuen & Co.: John Wiley & Sons, 1962
[5] Ore O. Theory of Graphs. Providence: American Mathematical Society, 1962
[6] Cockayne E J, Hedetniemi S T. Towards a theory of domination in graphs. Networks, 1977, 7(3): 247-261
[7] Garey M R, Johnson D S. Computers and Intractability: A Guide to the Theory of NP-Completeness. New York: Freeman, 1979
[8] Chang G J, Nemhauser G L. The k-domination and k-stability problems in sun-free chordal graphs. SIAM Journal on Algebraic and Discrete Methods, 1984, 5(3): 332-345
[9] Miroslav Chlebik, Janka Chlebikova. Approximation hardness of dominating set problems//Albers S, Radzik T eds. ESA 2004. LNCS 3221. Berlin: Springer, 2004: 192-203
[10] Guha S, Khuller S. Approximation algorithms for connected dominating sets. Algorithmica, 1998, 20(4): 374-387
[11] Halldoresson M M. Approximating the minimum maximal independence number. Information Processing Letters. 1993, 46(4): 169-172
[12] Hastad J. Clique is hard to approximate within $n^{1-\varepsilon}$. Acta Mathematica, 1999, 182(1): 105-142
[13] Reed B. Paths, stars and the number three. Combinatorics, Probability and Computing, 1996, 5(3): 277-295
[14] Flach P, Volkmann L. Estimations for the domination number of a graph. Discrete Mathematics, 1990, 80(2): 145-151
[15] Hedetniemi S T, Laskar R C. Connected domination in graphs//Bollobas B ed. Graph Theory and Combinatorics. London: Academic Press, 1984: 209-218
[16] Sampathkumar E, Walikar H B. The connected domination number of a graph. Journal of Mathematical Physical. Sciences, 1979, 13: 607-613
[17] Kleitman D J, West D B. Spanning trees with many leaves. SIAM Journal on Discrete Mathematics, 1991, 4(1): 99-106
[18] Dahlhaus E, Kratochvil J, Manual P, Miller M. Transversal partitioning in balanced hypergraphs. Discrete Applied Mathematics, 1997, 79(3): 75-89
[19] Fomin F V, Kratsch D, Woeginger G J. Exact (exponential) algorithms for the dominating set problem//LNCS 3353. Berlin: Springer, 2004: 245-256
[20] Davis M, Putnam H. A computing procedure for quantification theory. Journal of the Association for Computing Machinery, 1960, 7(3): 201-215
[21] Grandoni F. A note on the complexity of minimum dominating set. Journal of Discrete Algorithms, 2006, 4(2): 209-214
[22] Fomin F V, Grandoni F, Kratsch D. Measure and conquer: Domination-a case study//LNCS 3580. Berlin: Springer, 2005: 191-203
[23] Edmonds J, Johnson E L. Matching: A well-solved class of integer linear programs//Jünger M et al. eds. Combinatorial Optimization (Edmonds Festschrift). LNCS 2570. Berlin: Springer, 2003: 27-30
[24] David Eppstein. Quasiconvex analysis of multivariate recurrence equations for backtracking algorithms. ACM Transactions on Algorithms, 2006, 2(4): 492-509
[25] Robson J M. Algorithms for maximum independent sets. Journal of Algorithms, 1986, 7(3): 425-440
[26] Fomin F V, Grandoni F, Pyatkin A V, Stepanov A A. Bounding the number of minimal dominating sets: A measure and conquer approach//LNCS 3827. Berlin: Springer, 2005: 573-582
[27] Fomin F V, Grandoni F, Pyatkin A V et al. Combinatorial bounds via measure and conquer: Bounding minimal dominating sets and applications. ACM Transactions on Algorithms, 2008, 5(1): 9:1-9:17
[28] Tang Jian. An $O(2^{0.304N})$ algorithm for solving maximum independent set problem. IEEE Transactions on Computers, 1986, 35(9): 847-851
[29] Beigel R. Finding maximum independent sets in sparse and general graphs//Proceedings of the 10th ACM-SIAM Symposium on Discrete Algorithms. Baltimore, 1999: 856-857
[30] Fomin F V, Grandoni F, Kratsch D. Measure and conquer: A simple $O(2^{0.288n})$ independent set algorithm//Proceedings of the seventeenth Annual ACM-SIAM Symposium on Discrete Algorithm. Miami, 2006: 18-25
[31] Fomin F V, Grandoni F, Kratsch D. Solving connected dominating set faster than 2^n//Arun-Kumar S, Garg N eds. FSTTCS 2006. LNCS 4337. Berlin: Springer, 2006: 152-163
[32] Serge Gaspers, Mathieu Liedlof. A branch-and-reduce algorithm for finding a minimum independent dominating set in graphs//Fomin F V ed. WG 2006. Berlin: Springer, 2006: 78-89
[33] Fomin F V, Stepanov A A. Counting minimum weighted dominating sets//Lin G ed. LNCS 4598. Berlin: Springer, 2007: 165-175
[34] Johan M M van Rooij, Jesper Nederlof, Thomas C van Dijk. Inclusion/Exclusion meets measure and conquer: Exact algorithms for counting dominating sets//LNCS 5757, 2009: 554-565
[35] Moon J W, Moser L. On cliques in graphs. Israel Journal of Mathematics, 1965, 3(1): 23-28
[36] Martin Furer. A faster algorithm for finding maximum independent sets in sparse graphs//Correa J R, Hevia A, Kiwi M eds. LATIN 2006. LNCS 3887. Berlin: Springer, 2006: 491-501
[37] Vilhelm Dahllöf, Peter Jonsson. An algorithm for counting maximum weighted independent sets and its applications//Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms. San Francisco, 2002: 292-298
[38] Dieter Kratsch, Mathieu Liedloff. An exact algorithm for the minimum dominating clique problem//Bodlaender H L, Langston M A eds. IWPEC 2006. LNCS 4169. Berlin: Springer, 2006: 130-141

[39] Bourgeois N, Della Crocl F, Escoffier B et al. Exact algorithms for dominating clique problems//LNCS 5878. 2009: 4-13

[40] SchÄoning U. Algorithmics in exponential time//LNCS 3404. Berlin: Springer, 2005: 36-43

[41] Woeginger G. Exact algorithms for NP-hard problems: A survey//Combinatorial Optimization — Eureka, you shrink! LNCS 2570. Berlin: Springer, 2003: 185-207

[42] Lu Gang, Zhou Ming-Tian, Niu Xin-Zheng et al. A survey of proximity graphs in wireless networks. Journal of Software, 2008, 19(4): 888-911(in Chinese)
(路纲，周明天，牛新征等. 无线网络邻近图综述. 软件学报，2008，19(4)：888-911)

[43] Fomin F V, Grandoni F, Kratsch D. A measure and conquer approach for the analysis of the exact algorithms. Journal of the ACM, 2009, 56(5): 25.1-25.32

附录 1.　由 lmds 算法给出的图 2(a)的所有 MDS.

附表 1　图 2(a)的所有最小支配集

序号	支配集中节点的编号	序号	支配集中节点的编号	序号	支配集中节点的编号	序号	支配集中节点的编号
第 1 个	2 6 9 10 17 18 23 25	第 15 个	2 7 9 13 17 18 25 30	第 29 个	2 9 12 15 17 20 25 30	第 43 个	2 12 15 17 20 21 28 30
第 2 个	2 6 9 10 17 18 25 30	第 16 个	2 9 10 12 17 20 23 25	第 30 个	2 9 13 14 17 18 23 25	第 44 个	2 12 15 17 20 23 25 28
第 3 个	2 6 9 11 15 18 23 25	第 17 个	2 9 10 12 17 20 25 30	第 31 个	2 9 13 14 17 18 25 30	第 45 个	2 12 15 17 20 25 28 30
第 4 个	2 6 9 13 17 18 23 25	第 18 个	2 9 10 13 17 18 23 25	第 32 个	2 9 13 17 18 20 23 25	第 46 个	2 15 17 18 20 21 23 28
第 5 个	2 6 9 13 17 18 25 30	第 19 个	2 9 10 13 17 18 25 30	第 33 个	2 9 13 17 18 20 25 30	第 47 个	2 15 17 18 20 21 28 30
第 6 个	2 6 9 15 17 18 23 25	第 20 个	2 9 10 17 18 20 23 25	第 34 个	2 9 15 17 18 20 23 25	第 48 个	2 15 17 18 20 23 25 28
第 7 个	2 6 9 15 17 18 25 30	第 21 个	2 9 10 17 18 20 25 30	第 35 个	2 9 15 17 18 20 25 30	第 49 个	2 15 17 18 20 25 28 30
第 8 个	2 6 11 15 18 21 23 28	第 22 个	2 9 11 12 15 20 23 25	第 36 个	2 11 12 15 20 21 23 28	第 50 个	8 9 11 12 15 20 25 30
第 9 个	2 6 11 15 18 23 25 28	第 23 个	2 9 11 12 15 20 25 30	第 37 个	2 11 12 15 20 21 28 30	第 51 个	8 11 12 15 20 21 28 30
第 10 个	2 6 15 17 18 21 23 28	第 24 个	2 9 11 13 14 18 23 25	第 38 个	2 11 12 15 20 23 25 28	第 52 个	8 11 12 15 20 25 28 30
第 11 个	2 6 15 17 18 21 28 30	第 25 个	2 9 11 15 18 20 23 25	第 39 个	2 11 12 15 20 25 28 30	第 53 个	9 11 12 15 19 20 25 30
第 12 个	2 6 15 17 18 23 25 28	第 26 个	2 9 12 13 17 20 23 25	第 40 个	2 11 15 18 20 21 23 28	第 54 个	11 12 15 19 20 21 28 30
第 13 个	2 6 15 17 18 25 28 30	第 27 个	2 9 12 13 17 20 25 30	第 41 个	2 11 15 18 20 23 25 28	第 55 个	11 12 15 19 20 25 28 30
第 14 个	2 7 9 13 17 18 23 25	第 28 个	2 9 12 15 17 20 23 25	第 42 个	2 12 15 17 20 21 23 28	第 56 个	11 12 15 19 20 28 29 30

附录 2.　算法 *lmds* 的进化版本 *ls*.

1～3. 同 *lmds* 步 1～步 3(也可以添加步 0，方法同)

4. S_i为最大势子集，如果$\exists u\in S_i$频率为 2，则寻找所有的 $S_j(\neq S_i)$，要求这些 S_j与 S_i共享一个频率为 2 的元素(不单 u). 返回 $ls(U-\{S_i\}, S-(S_i), S^*\cup S_i)$；返回 $ls(U-\{S_j\}, S-(S_iS_j), S^*\cup S_j)$；

5. 寻找 S_i具最大势，且$|S_i|\geq 3$. 处理方法同 *lmds* 第 4 步；

6. 寻找 S_j具最大势及$S_i\subseteq S_j$，处理同 *lmds* 第 8 步；

7. 寻找 u 具有最大出现频率，设 $S_i=\{u, v\}$，其余同 *lmds* 第 10 步.

附录 3.　最小连通支配集算法 MCDS.

算法使用符号的含义：连通点集 S 代表 CDS；弃点集 D 表示已抛弃节点；可用点集 $A=V-\{S\}-\{D\}$；侯选点集 candidate 表示 A 中与 S 一跳邻接的点；算例不可行表示 $G[V-D]$不构成 CDS；期望点 promise 表示 A 中的某点 v，如果将 v 加入 D 后该例不可行；自由点集 $F=V-N[S]$(不失一般性假设图 G 是无向连通图).

1. 建立图 G 的边集 E；

2. 取出第一条边 e 两端点构成初始 S，然后 $E\backslash e$(G 中该边保留，仅在集合 E 中删)；计算初始 D, A, F 集合；如 E 空，算例停，比较对应每条边的$|S|$值，选取最小的作为最终输出；

3. 如果当前 S 还不是 CDS，但$|S|$大于算法已求出的最小$|CDS|$；或者算例不可行；或者 S 已经是 CDS，记录相应的 S 集合，停止该算例，转步 2；否则向下执行；

//以下步 4～6 是简化规则，不产生任何分支；步 7～9 是分支规则，每个产生至少两个分支；

4. 在 candidate 中寻找一点 v，要求其满足 promise 条件. 令 $S=S\cup v, A=A-v, F=F-N[v]$；转步 3；

5. 在侯选点集 candidate 中寻找两点 v, w，要求满足点集$\{N(v)\cap F\}\subseteq$点集$\{N(w)\cap F\}$，如果存在这样的 v, w，则 $D=D\cup v, A=A-v$；转步 3；

6. 在可用点集 A 中寻找一点 v，使满足 $N(v)\cap F=\varnothing$，即没有 v 支配任何自由节点. 如果存在这样的 v，则 $D=D\cup v, A=A-v$；转步 3；

7. 在 candidate 中寻找某点 v，使满足下面条件二者之一即可，产生两个分支：

条件 1. $|N(v)\cap F|\geq 3$，即 v 支配了三个或以上的自由节点；

条件 2. $N(v)\cap A=w$，w 是一个点集，如果存在某 w_i 使 $(N(v)\cup N(w_i))\cap F=\varnothing$；

(1) $S=S\cup v$，$A=A-v$，$F=F-N[v]$；转步 3；

(2) $D=D\cup v$，$A=A-v$；转步 3；

8. 在 candidate 中寻找某点 v，满足 $|N(v)\cap F|=|w|=1$. 令 $U=(N(w)\cap A)-N[v]$，再产生 3 个递归调用：

(1) $D=D\cup v$，$A=A-v$；转步 3；

(2) $S=S\cup v\cup w$，$A=A-v-w$，$F=F-N[v]-F[w]$；转步 3；

(3) $S=S\cup v$，$D=D\cup w\cup\{U\}$，$A=A-v-w-U$，$F=F-N[v]$；转步 3；

9. 在 candidate 中寻找某点 v，$|N(v)\cap F|=|w|=2$. 如果存在这样的 v 进入本步骤，否则转步 3.

假设 $w=\{w_1,w_2\}$，然后按规则确定 w 中的两元素下标号：

(1) 如果 w_1，w_2 同属于 A 或同时不属于 A，当前下标合法；否则应用下一个规则；

(2) 如果 $w_1\subseteq A$，当前下标合法；否则，交换 $w_1\leftrightarrow w_2$.

对 $i=1,2$ 计算，$U_i=(N[w_i]\cap A)-N[v]$；判断是否满足下面的 3 个条件.

条件 1. 如果 w_1，w_2 是邻节点，且 $w_1\subseteq A$，$w_2\subseteq D$，产生下面 3 个分支：

(1) $D=D\cup v$，$A=A-v$；转步 3；

(2) $S=S\cup v\cup w_1$，$A=A-v-w_1$，$F=F-N[v]-F[w_1]$；转步 3；

(3) $S=S\cup v$，$D=D\cup w_1\cup\{U_1\}$，$A=A-v-w_1-U_1$，$F=F-N[v]$；转步 3；

条件 2. 如果 w_1，w_2 是邻节点，且 $w_1\subseteq A$，$w_2\subseteq A$，产生下面 4 个分支：

(1) $D=D\cup v$，$A=A-v$；转步 3；

(2) $S=S\cup v\cup w_1$，$A=A-v-w_1$，$F=F-N[v]-F[w_1]$；转步 3；

(3) $S=S\cup v\cup w_2$，$D=D\cup w_1$，$A=A-v-w_1-w_2$，$F=F-N[v]-N[w_2]$；转步 3；

(4) $S=S\cup v$，$D=D\cup w_1\cup w_2\cup\{U_1\}\cup\{U_2\}$，$A=A-v-w_1-w_2-U_1-U_2$，$F=F-N[v]$；转步 3；

条件 3. 如果不满足上述条件 1、2，则产生下面 5 个分支：

(1) $D=D\cup v$，$A=A-v$；转步 3；

(2) $S=S\cup v\cup w_1$，$A=A-v-w_1$，$F=F-N[v]-F[w_1]$；转步 3；

(3) $S=S\cup v\cup w_2$，$D=D\cup w_1$，$A=A-v-w_1-w_2$，$F=F-N[v]-N[w_2]$；转步 3；

(4) $S=S\cup v$，$D=D\cup w_1\cup w_2\cup\{U_1\}$，$A=A-v-w_1-w_2-U_1$，$F=F-N[v]$；转步 3；

(5) $S=S\cup v$，$D=D\cup w_1\cup w_2\cup\{U_2\}$，$A=A-v-w_1-w_2-U_2$，$F=F-N[v]$；转步 3.

附录 4. 最小独立支配集算法 MIDS.

符号含义，$N_F(u)$：点 u 的 F 邻节点，即 $F\cap N(u)$；$d_F(u)$：点 u 的 F 邻节点度，即 $|N_F(u)|$；其余类同.

算法. $|S|=ids(G,F,M,E)$

输入：图 G、自由节点集 F(初始化为所有节点)、标记点集 M(初始化为空$\varnothing$)、E(G 的边集合)

输出：图 G 的最小独立支配集大小 $|S|$(略做改动，算法可直接输出一个具体的 $MIDS$)

1. 如果 $F=\varnothing$ 且 $M=\varnothing$，返回 $|S|=0$；

2. 如果 $\exists u\in M$ 且 $d_F(u)=0$，返回 $|S|=\infty$；

3. 如果 $\exists u\in M$ 且 $N_F(u)=v$，返回 $|S|=1+ids(G,F-N[v],M-N(v),E)$；

4. 如果某连通组元 $C\subseteq F$ 且 C 为团、且 $N_F(C)=\varnothing$，返回 $|S|=1+ids(RedClique(G,F,M,E))$；

5. 如果某连通组元 $B\subseteq F$ 且 B 为完全二分图(X,Y 两分部)、且 $N_F(B)=\varnothing$，返回 $|S|=\min\{|X|+ids(G,F-N[X],M-N(X),E);|Y|+ids(G,F-N[Y],M-N(Y),E)\}$；

6. 如果某连通组元 $C\subseteq F$ 且 C 为团、$|C|\geq 3$ 且 $\exists$ 唯一 $v\in C$ 使得 $d_F(v)\geq|C|$，返回 $|S|=\min\{1+ids(G,F-N[v],M-N(v),E);ids(G,F-v,M\cup v,E)\}$；

7. 如果上述条件都不满足则选择一个 $u\in F$，要求 u 具有最小 F 邻节点度，进入以下 3 种情况：

(1) 如果 $d_F(u)=1$，返回 $S=\min\{1+ids(G,F-N[u],M-N(u),E);ids(G,F-N[N_F(u)],M-N(N_F(u)],E)\}$；

(2) 如果 $d_F(u)=2$，令 $N_F(u)=\{v_1,v_2\}$，返回，$|S|=\min\{ids(G,F-N[u],M-N(u),E);ids(G,F-N[v_1],M-N(v_1),E);ids(G,F-N[v_2]-v_1,M+v_1-N(v_2),E)\}$；

(3) 否则($d_F(u)>2$)，从 u 的 F 邻节点中选取具有最大 F 邻节点度的某点 v，返回 $|S|=\min\{1+ids(G,F-N[v],M-N(v),E);ids(G,F-v,M+v,E);\}$；

函数：$RedClique(G,F,M,E,C)$

输入：图 G、自由点集 F、标记点集 M、边集 E、$G[F]$中连通组元 C(且 C 为团)

输出：标记图 G'、F'、M'、E'

1. 如果 $|C|=1$，返回 $G'=G$、$F'=F-C$、$M'=M=N(C)$、$E'=E$；

2. 如果 $\exists v\in C$，且 $N_M(v)=\varnothing$，令 $F=F-v$、$C=C-v$；返回 $RedClique(G,F,M,E,C)$；

3. 如果前面条件都不满足，令 $N(C)=\{h_1,h_2,\cdots,h_k\}$，$H=\varnothing$. 然后，

(1) 针对所有点对组合 (h_i,h_j) 判断，如果 $N_C(h_i)\cap N_C(h_j)=\varnothing$，则生成新点 h_{ij}，$H=H\cup h_{ij}$；

(2) $F'=F-C$；$M=M-N(C)$；$M'=M\cup H$；

(3) 针对每个点 h_{ij} 判断，如果 $\exists v\in N_F(N[C])$，且边 $(v,h_i)\in E$ 或边 $(v,h_j)\in E$，则 $E'=E\cup(v,h_{ij})$.

附录 5.　例图点坐标及生成方法.

附表 2　图 2(a)的点坐标

index	*x*	*y*	*index*	*x*	*y*	*index*	*x*	*y*
1	58.6440	30.3661	11	40.3491	28.5947	21	93.0041	50.3888
2	67.5112	4.6192	12	12.2021	54.3663	22	39.9020	64.6810
3	36.1022	19.5477	13	26.8439	98.4776	23	4.7401	30.7746
4	62.0278	72.0166	14	25.7846	71.5678	24	34.2374	13.8725
5	81.1151	72.1753	15	33.1665	83.8970	25	73.5966	47.5573
6	1.9257	87.7799	16	15.2234	43.3261	26	79.4682	36.2459
7	8.3874	58.2433	17	34.8008	47.0625	27	54.4906	78.8113
8	97.4802	7.0684	18	12.1658	56.0713	28	68.6223	78.0296
9	65.135	92.2745	19	88.4153	26.9092	29	89.3633	66.8512
10	23.1238	80.0372	20	9.4278	74.9018	30	5.4792	13.3504

生成图 2(a)中 GG 图的方法参考文献[42].

LU Gang, born in 1972, Ph. D.. His research interests include computational geometry, graph theory and their applications.

ZHOU Ming-Tian, born in 1939, professor, Ph. D. supervisor. His research interests include network computing and information security.

TANG Yong, born in 1973, Ph. D., associate professor. His research interests include network computing, network management, and wireless networking.

WU Zhen-Qiang, born in 1968, Ph. D., associate professor. His research interests include information and Network security, mobile computing and trusted computing.

QIU Guo-Yong, born in 1966, Ph. D., associate professor. His research interest focuses on combinatorial optimization.

YUAN Liu, born in 1979, Ph. D.. Her research interests include semantic Web and information retrieval.

Background

This work was supported by the National High Technology Research and Development Program under grant No. 2007AA01Z438200, the National Natural Science Foundation of China under grant No. 60633020 and the Research Project of Shaanxi Normal University under grant No. 999414.

As remarked by Haynes in Ref. [1], Dominating Set (and related problems) is "an area of graph theory that is rich in history, applications, interesting questions and results, and unsolved research questions." Within the last fifty years, concurrent with the growth of computer science and engineering, the study of domination has seen explosive growth both in algorithm design and application. To date, thousands of papers have been written on this topic.

Recently, especially after 2004, the interest of design and analysis of the exact exponential time algorithm has been growing significantly. This interest has several motivations:

(1) Before the very unexpected thing in complexity theory occurs, the best we can hope for when we are dealing with an NP hard problem are the exponential algorithms.

(2) Approximation algorithms are not always satisfactory, and many problems are hard to approximate.

(3) When we are starting to design an approximation algorithm, the results of exact solution for moderate input size are usually helpful.

(4) A reduction of the base of the exponential running time, together with the increased speed of the modern computers, can effectively improve the size of the instances solvable. This may actually lead to practical algorithms.

(5) The design and analysis of exact algorithms leads to a better understanding of NP-hard problems and will initiates new combinatorial and algorithmic challenges.

One of the major techniques for constructing fast exponential time algorithms is the Branch & Reduce paradigm which can be traced back to sixties in 20th century. Such algorithms (also called search tree algorithms) recursively solve NP hard problems by using reduction rules and branching rules. Though have been used for more than 40 years, the analytical tools for Branch & Reduce algorithms available are still far from producing tight worst-case running time bounds. An important progress——"Measure & Conquer" analysis technique occurred only few years ago. The M&C technique has shown that a good choice of measure for analyzing branching algorithms can lead to a significantly better analysis results. Other techniques such as Local Search, Randomization, Memorization, Dynamic Programming and so on, which are interesting in their own, have also been used in the design and analysis of exact algorithm.