文章编号:1007－6735(2012)04－0389－05

集合覆盖问题降阶算法

宁爱兵， 刘艳芳， 王英磊

（上海理工大学 管理学院,上海 200093）

摘要：集合覆盖问题是运筹学与计算机科学中的一个NP难题.首先将该问题转化为一个等价的二分图,给出该问题的上下界算法;接着给出该问题的数学性质,这些数学性质能降低问题的规模,加快算法的求解速度;然后将数学性质和上下界方法结合起来形成一个降阶算法,并给出了算法的时间复杂度分析.该算法不仅可以单独使用,还可以与其它算法结合起来使用达到更好的效果.最后通过多个示例进一步说明算法的原理及应用情况.

关键词：集合覆盖问题；降阶算法；上界；下界

中图分类号：O 223；N 94 **文献标志码**：A

Reduction Algorithm for Set Covering Problem

NING Ai-bing， LIU Yan-fang， WANG Ying-lei

(*School of Management, University of Shanghai for Science and Technology, Shanghai 200093, China*)

Abstract: The set covering problem (SCP) is a classical NP-hard problem in operation research and computer science. A SCP was transformed into an equivalent bipartite graph (BG) and an upper-lower bound algorithm for SCP or BG was proposed. Some mathematical properties of SCP or BG were presented which can decrease the size of the problem and can speed up the algorithm. A new reduction algorithm for SCP was proposed by combining the mathematical properties with the upper-lower bound algorithm. Then the worst-case time complexity of the new reduction algorithm was analysed. The reduction algorithm proposed can be used alone, or cooperating with other algorithms to get more effective results. Several instances were solved and analyzed to illustrate the principles and applications of the algorithm.

Key words: *set covering problem*; *reduction algorithm*; *upper bound*; *lower bound*

集合覆盖问题(set covering problem,SCP)是组合优化中一个典型NP(non-deterministic polynomial time,非确定多项式时间算法)难解问题,是计算机和运筹学中的一个经典问题,在人员调动、网络安全、资源分配、电路设计、运输车辆路径安排等领域有着广泛的应用[1－4].

对于集合覆盖问题,目前国内外主要有两种处理方法:一种是采用启发式算法[2－3];另外一种是采

收稿日期：2012－02－21

基金项目：国家自然科学基金资助项目(70871081);上海市重点学科建设资助项目(S30504)

作者简介：宁爱兵(1972－),男,副教授.研究方向:算法设计、系统工程.E-mail:nabnab@163.com

用精确算法或近似算法[1,4]. 各种算法的优缺点及其对比将在后面介绍.

本文将集合覆盖问题转换成二分图,再在二分图上给出降阶规则,给出上界和下界子算法,最后结合成一个新的降阶算法.

1 问题及算法介绍

1.1 问题简介

集合覆盖问题是一个优化问题,其原型是多资源选择问题. 集合覆盖问题可以看作是图的顶点覆盖问题的推广,它也是一个NP难问题.

集合覆盖问题的一个实例$\langle X, F\rangle$由一个有限集 X 及 X 的一个子集族 F 组成. 子集族 F 覆盖了有限集 X,也就是说 X 中每一元素至少属于 F 中的一个子集,即 $X = \bigcup_{S\in F} S$. 对于 F 中的一个子集 $C\subseteq F$,若 C 中的 X 的子集覆盖了 X,即 $X = \bigcup_{S\in C} S$,则称 C 覆盖了 X. 集合覆盖问题就是要找出 F 中覆盖 X 的最小子集 C^*,使得 $|C^*| = \min\{|C \parallel C\subseteq F$ 且 C 覆盖 $X\}$.

1.2 问题转换

在算法处理之前,将该问题转换成二分图来进行处理,转换方法如下:

将集合覆盖问题$\langle X, F\rangle$中 X 的每个元素作为二分图的一个顶点子集 X,把 F 中的每个元素作为二分图的另一个顶点子集 F,F 中的元素是集合. 若 F 中的元素 f_1 包含 X 中的元素 x_1,则在 f_1 与 x_1 之间连线,否则不连线.

这样处理后,得到一个图 $G = (V, E)$,其中,$V = X\cup F$,$E = \{(x, f) \mid x\in X, f\in F$ 且 $x\in f\}$. 很显然,图 G 是二分图. 而原问题的求解变为在顶点子集 F 中找出一个最小的子集 C^*,使得顶点子集 X 中的每一个顶点至少与 C^* 中的顶点有一条边相连.

1.3 数学符号

为了方便描述,除了问题简介和问题转换中定义的符号外,还定义如下符号:

n 表示二分图 G 中顶点集 X 中结点的个数;

m 表示二分图 G 中顶点集 F 中结点的个数;

$d(v_i)$表示结点 v_i 的度,其值为与 v_i 关联边的条数;

$N(v_i)$表示二分图中顶点 v_i 的邻点集,$N(v_i) = \{v_j \mid (v_i, v_j)\in E\}$;

Φ 表示空集;

b 表示最小覆盖子集 C^* 中元素个数的下界,即 $|C^*|\geqslant b$;

u 表示最小覆盖子集 C^* 中元素个数的上界,即 $|C^*|\leqslant u$.

1.4 下界子算法

对 X 中的元素 x_i,由于必须至少有1个集合覆盖它,也就是说 C^* 中至少有1个结点要与它关联,因此 $N(x_i)$至少有1个结点在 C^* 中,故有下面的下界子算法.

a. $i = 1$;

b. 集合 $temp\_c = \Phi$;

c. 若 $i > n$ 或者 $temp\_c$ 中元素的数量为 m,则整个算法结束并得到下界 b;

d. 对 X 中的元素 x_i,若 $N(x_i)\cap temp\_c = \Phi$,则 $i = i + 1$, $b = b + 1$, $temp_c = N(x_i)\cup temp_c$并跳至第 c 步,否则执行 $i = i + 1$ 并跳至第c步.

算法中的 $N(x_i)\cap temp\_c = \Phi$ 说明目前还没有集合覆盖 x_i.

1.5 上界子算法

采用如下贪心算法求解上界 u:

a. 令 G_2 为待处理的二分图,$X_2 = X$,$F_2 = F$;

b. $u = 0$,集合 $temp\_c = \Phi$;

c. 若 X_2 为空集,则算法结束,否则跳至第d步;

d. 在 F_2 中找出1个度最大的结点 f_1,其中 $f_1\in F_2$;

e. $u = u + 1$, $temp\_c = temp\_c\cup\{f_1\}$,在二分图 G_2 中删除结点 f_1 及其关联边,$F_2 = F_2 - \{f_1\}$,删除集合 $N(f_1)$中的结点及其关联边,$X_2 = X_2 - N(f_1)$,修改对应结点的 $d(v_i)$,跳到第c步.

该子算法对图的删除操作都是在临时变量图 G_2 中进行的,不影响原来的二分图.

1.6 降阶规则

规则1 对顶点集 X 中度为1的结点 x_i,其关联边对应的另一个结点 f_j 一定在最小子集 C^* 中.

证明 因为度为1的结点 x_i 必须被某个集合覆盖,而 x_i 仅仅只属于一个集合 f_j,故要覆盖 x_i,则 f_j 一定在最小子集 C^* 中.

规则2 对顶点集 F 中的两个元素 f_1 和 f_2,若 $N(f_1)\subseteq N(f_2)$,则可以将顶点 f_1 及其关联边从图中删除而不会影响最小的子集 C^* 求解.

证明 因为 $N(f_1)\subseteq N(f_2)$,所以 f_1 所起到

的覆盖作用完全可以由 f_2 来代替.代替后的覆盖功能只会更强,而不会更弱,而目标函数最小子集 C^* 中的数量不变.因此可以把顶点 f_1 及其关联边从图中删除而不会影响最小子集 C^* 求解.

在降阶的中间过程中,可能会出现度为 0 的结点,也可能在 F 中存在 1 个能覆盖 X 中所有结点的子集合 f_1,此时采用规则 3 和规则 4 来降阶.

规则 3　在降阶的任何过程中,可以将顶点集 F 中度为 0 的结点 f_1 直接从图中删除.

证明　由于顶点集 F 中度为 0 的结点 f_1 不能覆盖任何结点,因此可以直接删除.

规则 4　在降阶的任何过程中,若顶点集 F 中存在一个结点 f_1 覆盖 X 中的所有结点,则将 f_1 加入到最小子集 C^* 中.

证明　由于此时 X 中还有结点没有被覆盖,故至少需要在 F 中找出一个子集合来覆盖,而能在 F 中找到一个能覆盖 X 中的所有结点的子集合 f_1,则可以将 f_1 加入到最小子集 C^* 中.

其它的降阶方法在后面的算法中介绍.

1.7　降阶算法

对于已经转成二分图的问题,采用如下算法进行降阶:

a. 下界 $b=0$,$C^*=\Phi$;

b. 调用下界子算法求出下界 b;

c. 按照规则 1,对于 X 中度为 1 的结点 x_i,把其关联边对应的另一个结点 f_j 加到最小子集 C^* 中,$b=b-1$,并把结点 x_i 及其关联边从图中删除.将结点 f_j、结点集合 $N(f_j)$ 及其关联边从图中删除,并修改对应点的度 $d(v_i)$,修改变量 n 和 m.这样持续做下去,直到 X 中不存在度为 1 的结点为止;

d. 按照规则 2,对顶点集 F 中的两个元素 f_1 和 f_2,若 $N(f_1)\subseteq N(f_2)$,则可以把顶点 f_1 及其关联边从图中删除并修改对应点的度 $d(v_i)$,同时修改变量 m;

e. 若 d 中删除过结点,则跳到第 c 步,否则执行下一步;

f. 采用规则 3 和规则 4 来降阶,调用上界子算法求出上界 u,若此时 $u=b$,则令 C_1 为上界子算法中的 $temp\_c$,此时 $C^*=C^*\cup C_1$ 为最优解,整个算法结束;

g. $j=0$;

h. 若 $j>m$,则跳到第 o 步;

i. 对 F 中的元素 f_j,令 $N_2(f_j)$ 为 $N(f_j)$ 中度为 2 的结点集合,若 $N_2(f_j)$ 中结点数量大于等于 u,则执行下一步,否则跳到第 n 步;

j. 集合 $temp\_c=\Phi$;

k. 把 $N_2(f_j)$ 中每一个结点 v_i 的邻点集 $N(v_i)$ 加到集合 $temp\_c$ 中;

l. $temp\_c=temp\_c-f_j$(这里操作的含义是假设 f_j 不在最小子集 C^* 中);

m. 若 $temp\_c$ 的数量大于 u,则把结点 f_j 加到最小子集 C^* 中,把结点 f_j 和结点集合 $N(f_j)$ 从图中删除,同时执行 $m=m-1$,$n=n-|N(f_j)|$,$u=u-1$,并修改对应点的度;

n. 采用规则 3 和规则 4 来降阶;$j=j+1$,跳到第 h 步;

o. 若在第 h 至 n 步中删除过结点,则跳到第 c 步,否则整个算法结束.此时 C^* 中的元素是在最小集合覆盖中的元素,若此时图为无边的空图,则得到最小集合覆盖 C^*,否则说明只是降低了问题的规模和难度,但没有得到最终的最优解.

算法中的第 i 到第 l 步是利用反证法降阶,即假设 f_j 不在最小集合覆盖 C^* 中时,若能判断此时要加入到 C^* 的元素个数大于上界 u,则说明出现矛盾,因此 f_j 一定在最小集合覆盖 C^* 中.

1.8　算法的时间复杂度分析

下面的分析,都是指在最坏情况下的时间复杂度.

问题转换的时间复杂度为 $O(nm)$,下界子算法为 $O(nm^2)$,上界子算法为 $O(nm)$,降阶算法的第 c 步为 $O(n)$,第 d 步为 $O(n^2m^2)$,第 h 步到第 n 步为 $O(n^2m)$,因此整个算法的时间复杂度为 $O(n^2m^3)$.

2　示例分析

为了更清楚地了解算法的原理及应用情况,下面给出 3 个示例来进行分析.

示例 1　求图 1(见下页)中图 G 的最小集合覆盖,其中,$X=\{a,b,c,d,e,f,g,h,i,j,k,l\}$,$F=\{s_1,s_2,s_3,s_4,s_5,s_6\}$,$s_1=\{a,b,e,f,i,j\}$,$s_2=\{f,g,j,k\}$,$s_3=\{a,b,c,d\}$,$s_4=\{e,f,g,h\}$,$s_5=\{i,j,k,l\}$,$s_6=\{d,h\}$,该示例取自文献[1],文献[1]提供的近似算法.

分析方法一:

Step 1　先把图 1 转成如图 2(见下页)所示的二分图;

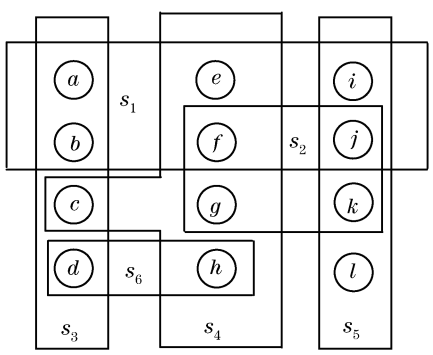

图1 示例1

Fig.1 Instance 1

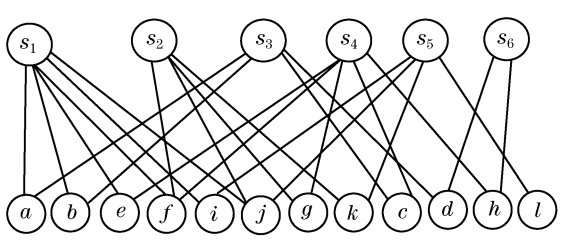

图2 二分图

Fig.2 Bipartite graph

Step 2 调用下界子算法求出下界 $b=3$；

Step 3 按照规则1,对于 X 中度为1的结点 l,将其关联边对应的另一个结点 s_5 加到最小子集 C^* 中,得 $C^*=\{s_5\}$,$b=b-1=2$,并将结点 l 及其关联边从图中删除,把结点 s_5、结点集 $N(s_5)$ 及其关联边从图中删除,并修改对应点的度 $d(v_i)$,修改变量 $n=8$ 和 $m=5$,得到图3所示的图形；

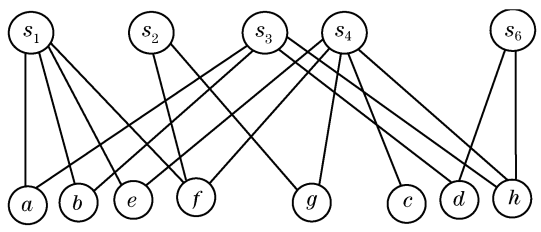

图3 二分图

Fig.3 Bipartite graph

Step 4 按照规则2,对顶点集 F 中的两个元素 s_2 和 s_4,由于 $N(s_2)\subseteq N(s_4)$,所以可以将顶点 s_2 及其关联边从图中删除并修改对应点的度 $d(v_i)$,同时修改变量 $m=5-1=4$,得到图4所示的图形；

Step 5 按照规则1,对于 X 中度为1的结点 g,把其关联边对应的另一个结点 s_4 加到最小子集 C^* 中,得 $C^*=\{s_4,s_5\}$,$b=b-1=1$,并将结点 g 及其关联边从图中删除,把结点 s_4、结点 $N(s_4)$ 及其关联边从图中删除,并修改对应点的度 $d(v_i)$,修改变量 $n=3$ 和 $m=3$,得到图5所示的图形；

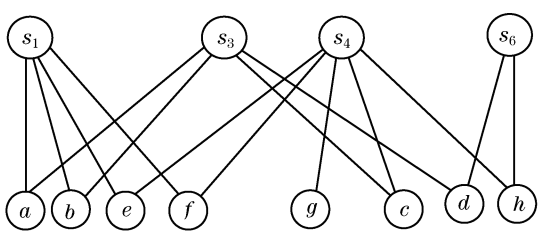

图4 二分图

Fig.4 Bipartite graph

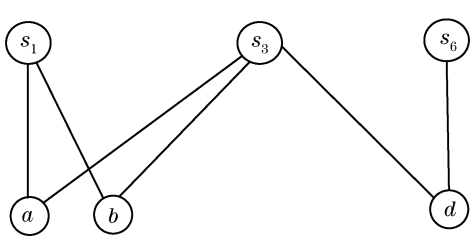

图5 二分图

Fig.5 Bipartite graph

Step 6 执行算法的第f步,求得上界 $u=1$,此时 $u=b$,得到最优解 $C^*=\{s_3,s_4,s_5\}$,整个算法结束.

分析方法二：

为了演示降阶算法中的第h到第n步,下面不使用规则2来降阶,也不使用算法第f步中的 $u=b$ 的方法来降阶.

这里的第1,2,3步同分析方法一中的完全相同,此时得到图3所示的图形.

Step 7 执行算法的第f步,求得上界 $u=2$；

Step 8 执行算法的第i步,对 F 中的元素 s_3,$N_2(s_3)=\{a,b,c,d\}$ 为 $N(s_3)$ 中度为2的结点集合,由于 $N_2(s_3)$ 中结点数量大于等于 u,则跳到算法的第j步；

Step 9 执行算法的第j步,集合 $temp\_c=\Phi$；

Step 10 执行算法的第k步,将 $N_2(s_3)=\{a,b,c,d\}$ 中每个结点 v_i 的邻点集 $N(v_i)$ 加到集合 $temp\_c$ 中,得 $temp\_c=\{s_1,s_3,s_4,s_6\}$；

Step 11 执行算法的第l步,得 $temp\_c=temp\_c-f_i=\{s_1,s_4,s_6\}$

Step 12 执行算法的第m步,$C^*=\{s_3,s_5\}$,将结点 s_3、结点集 $N(s_3)$ 从图中删除,$m=m-1=4$,$n=4$,$u=u-1=1$,得到图6；

Step 13 执行算法的第n步,采用规则4来降阶得到最优解 $C^*=\{s_3,s_4,s_5\}$.

示例2 $X=\{1,2,3,4\}$,$F=\{s_1,s_2,s_3,s_4\}$,其中 $s_1=\{1,2\}$,$s_2=\{2,3\}$,$s_3=\{3,4\}$,$s_4=\{4,1\}$.

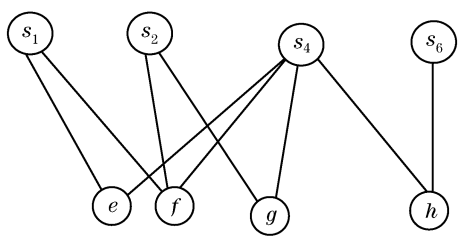

图 6　二分图

Fig.6　Bipartite graph

分析:

由下界子算法求得 $b=2$,当执行算法的第 f 步时,由上界子算法求得 $u=2$,由于 $u=b$,因此,求得最优解为 $C^*=\{s_1,s_3\}$.

示例 3　$X=\{a,b,c,d,e,f\}$,$F=\{s_1, s_2, s_3, s_4, s_5\}$,其中 $s_1=\{a,b\}$, $s_2=\{a,f\}$, $s_3=\{d,e\}$, $s_4=\{c,e\}$, $s_5=\{b,c,d\}$.

分析:

step 1　先将该问题转成如图 7 所示的二分图;

step 2　按照规则 1,并将 s_2 加到最小子集 C^* 中,得 $C^*=\{s_2\}$,将结点 s_2、结点集 $N(s_2)$ 及其关联边从图中删除;

step 3　按照规则 2,将 s_1 及其关联边删除;

step 4　再按照规则 1,将 s_5 加到最小子集 C^* 中,得 $C^*=\{s_2,s_5\}$,将结点 s_5、结点集 $N(s_5)$ 及其关联边从图中删除;

step 5　最后将 s_3 加到最小子集 C^* 中,得到最优解 $C^*=\{s_2, s_3,s_5\}$,或者把 s_4 加到最小子集 C^* 中,得到最优解 $C^*=\{s_2, s_4,s_5\}$.

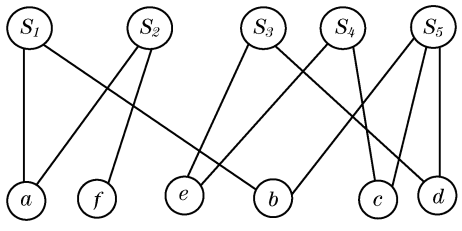

图 7　二分图

Fig.7　Bipartite graph

3　算法对比分析

对于集合覆盖问题,目前国内外主要有两种处理方法:一种是启发式算法[2,3],启发式算法的优点是能够快速得到可行解,其缺点是不能保证解是最优解,也不能提供解的近似比;另外一种是采用精确算法或近似算法[1,4],比如分枝定界法、回溯法等,精确算法的优点是能得到最优解,但时间复杂度都是指数级别的,难以处理规模较大的问题,近似算法在近似程度上提供保障,但不能得到最优解.

本降阶算法的优点在于利用问题的性质及上下界来加快问题的求解速度,使原问题变为规模更小的同性质的子问题,便于进一步的处理;不足之处在于只能对部分实例问题求解得到最终解,对不能得到最终解的问题实例还必须与其它方法结合起来才能得到最终解,比如对于经过降阶算法处理后的子问题,若规模较少,可以用分枝定界法来得到最优解,若规模仍然很大,则可以用启发式算法来求解.

4　结束语

集合覆盖问题是不具有多项式时间算法的 NP 难题,本文提供的降阶算法能在一定程度上降低原问题的求解规模与难度,而且在某些情况下,能直接得出问题的最优解.该方法不仅可以单独使用,而且可以与其它方法结合起来使用,从而加快问题的求解速度.

参考文献:

[1] 王晓东.计算机算法设计与分析[M].3 版.北京:电子工业出版社,2007.

[2] 陈端兵,黄文奇.一种求解集合覆盖问题的启发式算法[J].计算机科学,2007,34(4):133－136.

[3] Chvatal V. A greedy heuristic for the set covering problem[J]. Mathematics of Operations Research, 1979,4(3):233－235.

[4] Hassin R,Levin A. A better than greedy approximation algorithm for the minimum set cover problem[J]. SIAM Journal on Computing,2005,35(1):189－200.