GRÓSZ MIKLÓS

Egy leszámlálási algoritmus a halmaz lefedési probléma megoldására

Számos közgazdasági és műszaki probléma visszavezethető egy speciális ún. halmaz lefedési problémára (Set Covering Problem) [1]. Ezenkívül *Forgó Ferenc* [3] megmutatta, hogy egy tetszőleges nulla-egy kvadratikus programozási feladatot többek között egy halmaz lefedési feladattá lehet transzformálni. A transzformált feladat matrixában egy sorban sincs négynél több egyes. Hasonlóképpen igen ritka mátrixokkal találkozunk az éllefedési probléma [3] esetében, ahol a csúcs-él incidencia mátrix soraiban legfeljebb két egyes található.

Egy áttekintő cikkükben *Christofides* és *Korman* [5] vizsgálják az irodalomban ismertetett módszerek számítástechnikai eredményeit a halmaz lefedési probléma megoldására. Sűrű együttható mátrixok esetén az ismertetett módszerek hatékonynak bizonyultak. Ezek a módszerek azonban általában segédeljárásként egy lineáris programozási feladatot alkalmaznak, ami jelentősen növeli a gépi program memóriaigényét, és nem használják fel e speciális probléma sajátosságát ritka mátrixok esetén.

E dolgozatban bemutatunk egy olyan leszámlálási algoritmust, amely ritka mátrixok esetén annál hatékonyabb, minél ritkább az együttható mátrix, és semmiféle segédeljárást (pl. lineáris programozás) nem vesz igénybe. A halmaz lefedési probléma a következőképpen írható fel:

$$\sum_{j=1}^{n} c_j x_j \to \min$$

$$\sum_{j=1}^{n} a_{ij} x_j \geq 1 \tag{1}$$

$$x_j \in \{0,1\},\ a_{ij} \in \{0,1\}, \qquad (i = 1,2,\ldots,m;\ j = 1,2,\ldots,n).$$

Bevezetjük a következő jelöléseket:

$$\alpha_i^0 = \sum_{j=1}^{n} a_{ij} \quad \text{és} \quad \beta_j^0 = \sum_{i=1}^{m} a_{ij},$$

valamint egy definíciót:

Az **A** mátrix oszlopainak rendszerét *lefedési rendszernek* nevezzük, ha ebben a rendszerben minden egyes sorban legalább egy 1-es van.

Ennek következtében a halmaz lefedési problémát (1) úgy is értelmezhetjük, hogy keresünk egy olyan A' lefedési rendszert, amelyre nézve a célfüggvény értéke minimális. Ezt optimális lefedési rendszernek fogjuk hívni. Minden lefedési rendszernek egyértelműen az (1) probléma következő lehetséges meg-

oldása felel meg:

$$x_j = \begin{cases} 0, & \text{ha a } j\text{-edik oszlop } a_j \notin A' \\ 1, & \text{ha a } j\text{-edik oszlop } a_j \in A'. \end{cases}$$

Ezekután könnyen belátható, hogy $\alpha_i^0 \geq 1$ $(i = 1,2, \ldots, m)$ szükséges és elégséges feltétele az (1) probléma megoldhatóságának.

Ezért a továbbiakban feltételezzük, hogy $\alpha_i^0 \geq 1$, $(i = 1,2, \ldots, m)$. Az **A** mátrix méreteit a következőképpen lehet csökkenteni:

1. Ha $b_t = e_k$ (k-ik egységvektor), akkor $x_k = 1$ minden lehetséges megoldásban és az a_k oszlopot ki lehet hagyni. (b_i az **A** mátrix i-edik sora, a_k az **A** mátrix k-ik oszlopa.)

2. Ha $b_t \geq b_p$ valamely t-re és p-re, akkor a b_t sort ki lehet hagyni.

3. Ha valamilyen S oszlophalmazra és valamely k oszlopra igaz, hogy

$$\sum_{j \in S} a_j \geq a_k \quad \text{és} \quad \sum_{j \in S} c_j \leq c_k,$$

akkor a k-ik oszlopot ki lehet hagyni.

Tehát, az **A** mátrixból egy olyan **B** részmátrixot kapunk, amelyre nézve teljesül:

$$\alpha_i = \sum_{j \in J_0} a_{ij} \quad \text{és} \quad \alpha_i \geq 2, \; i \in I_0,$$

ahol

$$I_0 = \{1,2, \ldots, m'\}, \; J_0 = \{1,2, \ldots, n'\}, \; m' \leq m \text{ és } n' \leq n$$

Ezt az eljárást *1. tesztnek* nevezzük.

Lemke, Salkin és Spielberg [2] egy olyan heurisztikus módszert adtak, amely az optimumhoz közelálló lehetséges megoldást (lefedési rendszert) szolgáltat.

Tekintsük a

$$\gamma_j^k = \begin{cases} \dfrac{c_j}{\beta_j^k}, & \text{ha} \quad \beta_j^k \neq 0 \\ d, & \text{ha} \quad \beta_j^k = 0 \end{cases} \qquad (k = 0,1, \ldots)$$

számokat, ahol d egy tetszőlegesen nagy száma és

$$\beta_j^{k+1} = \beta_j^k - \langle a_{j_k}, a_j \rangle$$

Most kiválasztjuk a γ_j^k-k közül azt a γ_{j_k}-t, amelyre teljesül:

$$\gamma_{j_k} = \min_{j \in J_0} \gamma_j^k.$$

A megfelelő x_{j_k} értékét egyenlővé tesszük 1-el. Azokat a sorokat, ahol az a_{j_k} oszlopban egyes volt, kihagyjuk. Így kapjuk a I_k halmazt:

$$I_k \subset I_{k+1}, \quad (k = 1,2, \ldots).$$

Ezt az eljárást addig folytatjuk, amíg van eleme a I_k halmaznak. Ily módon nyerünk egy $x^0 = (x_1^0, x_2^0, \ldots, x_{n'}^0)$ lehetséges megoldást, amelynek Z_0 célfüggvény érték felel meg. Ezt az értéket kiinduló korlátként fogjuk használni a továbbiakban. Ez lesz a *2. teszt.*

A leszámlálási algoritmus azon az elgondoláson alapul, hogy lépésről lépésre beválasztunk oszlopokat a rendszerbe. Minden p-edik lépésnek megfelel egy I_p és J_p halmaz, valamint y_p célfüggvény érték. Az I_p halmaz tartalmazza azokat a sorindexeket, amelyek még lefedetlenül maradtak a p-edik lépésben. A J_p halmaz viszont tartalmazza azokat az oszlopindexeket, amelyek ebben a lépésben beválaszthatók a rendszerbe. A J_p halmazt úgy kapjuk, hogy az I_{p-1} halmazon kiválasztjuk azt a sort, amelyhez a minimális α_i^{p-1} sorösszeg tartozik:

$$\alpha_i^{p-1} = \min_{i \in J_{p-1}} \{\alpha_i\}, \qquad (p = 1,2, \ldots).$$

E sor nem nulla elemeinek az oszlopindexei a J_p halmazt alkotják.

Az algoritmus végrehajtása során a következő esetek egyike áll elő:

1. Az I_p halmaz nem üres és $y_p < z_0$, akkor a megfelelő J_p halmazból választunk egy j elemet (a kiválasztott elemet elhagyjuk a J_p halmazból) és folytatjuk a leszámlálást a $(p+1)$-edik lépésnél:

$$I_{p+1} := I_p \setminus T_j, \qquad (p = 0,1, \ldots),$$

$$J_p := J_p \setminus \{j\}, \qquad (p = 1,2, \ldots),$$

ahol Tj a j-edik oszlop nem nulla elemeinek megfelelő sorindexek halmaza.

2. Az I_p halmaz üres és $y_p < z_0$, vagyis egy olyan lefedési rendszert kapunk, amelyhez kisebb y_p célfüggvény érték tartozik, mint a z_0 korlát. Ekkor a továbbiakban ezt használjuk korlátként és a kapott lefedési rendszert megjegyezzük.

3. Ha a J_p halmaz üres vagy $y_p \geq z_0$, akkor visszatérünk a $(p-1)$-edik lépésre és visszaállítjuk az x_j^p és x_j^{p-1} értékét nullára.

Minden p-edik lépésben elég csak az i-edik sor (amelyhez minimális α_i^p tartozik), nem nulla elemeinek megfelelő oszlopokat megvizsgálni, mivel lefedési rendszereket keresünk és ezekből legalább egy oszlop eleme lesz minden lefedési rendszernek. A minimális α_i^p sorösszeg minden p-edik lépésben garantálja a minimális variációk számát. Ugyanakkor megkapjuk az összes lefedési rendszereket, amelyekhez kisebb célfüggvény érték tartozik, mint a kiinduló vagy a menet közben kapott korlát.

Az algoritmus leírása

1. Legyen $p = 0$ és $y_0 = 0$.

2. Alkalmazzuk az *1. tesztet* az **A** mátrixra. Kapunk egy **A'** almátrixot, amelyhez a sorindexek I_0 halmaza és az oszlopindexek J_0 halmaza tartozik.

3. Alkalmazzuk a *2. tesztet*, melynek segítségével egy lehetséges megoldást nyerünk z_0 célfüggvény-értékkel.

4. A mátrix sorösszegeiből kiválasztjuk az I_p halmazon a minimálisat:

$$\alpha_i^p = \min_{i \in I_p} \{\alpha_i\}$$

Ha a választás nem egyértelmű, akkor tetszés szerint választunk (pl. kisebb indexűt). A kiválasztott sorban legalább két egyes van ($\alpha_i^p \geq 2$ $i \in I_p$ az 1. teszt alkalmazása után), ezeknek a megfelelő oszlopindexei egy J_{p+1} halmazt alkotnak.

5. A J_{p+1} halmazból tetszőlegesen kiválasztunk egy j elemet és rögzítjük a megfelelő x_j^{p+1} értékét

$$x_j^{p+1} := 1,$$

vagyis beválasztottuk a j-edik oszlopot a rendszerbe.

6. A J_{p+1} halmazból elhagyjuk az 5.-ben kiválasztott j elemet:

$$J_{p+1} := J_{p+1} \setminus \{j\}.$$

7. Kiszámítjuk a célfüggvény értékét a $(p+1)$-edik lépésre:

$$y_{p+1} := y_p + c_j x_j^{p+1}.$$

8. Ha $y_{p+1} < z_0$, akkor 9., egyébként 12.

9. Azokat a sorindexeket, ahol a kiválasztott j-edik oszlopban egyesek vannak, T_j-vel jelöljük. Ezeket a sorokat elhagyjuk az I_p halmazból és így nyerjük a $(p+1)$-edik lépésnek megfelelő I_{p+1} halmazát:

$$I_{p+1} := I_p \setminus T_j \qquad (p = 0,1, \ldots)$$

10. Ha I_{p+1} üres halmaz, akkor kaptunk egy jobb lehetséges megoldást és folytatjuk 11.-nél, egyébként 18.-ra térünk át.

11. A továbbiakban a kapott lehetséges megoldás célfüggvény értékét használjuk korlátként:

$$z_0 := y_{p+1}.$$

Ezt a lehetséges megoldást tároljuk és a 15.-re térünk át.

12. Ha J_{p+1} halmaz üres, akkor visszatérünk a $(p-2)$-edik lépésre, mivel ez azt jelenti, hogy a J_p halmaznak egy eleme volt, de azt már megvizsgáltuk és a 13.-nál folytatjuk. Ha J_{p+1} halmaz nzm üres, akkor 19.-re térünk át.

13. Visszaállítjuk a x_j^{p+1}, x_j^p, x_j^{p-1} és a p paraméter értékét a $(p-2)$-edik lépésnek megfelelően:

$$x_j^{p+1} := x_j^p := x_j^{p-1} := 0, \qquad p := p-2.$$

14. Ha $p \geq 0$, akkor 15., egyébként 20.

15. Ha J_{p+1} halmaz nem üres, akkor visszatérünk 5.-re, egyébként folytatjuk 16.-nál.

16. Mivel J_{p+1} halmaz üres, visszatérünk a $(p-1)$-edik lépésre:

$$x_j^{p+1} := x_j^p := 0 \quad \text{és} \quad p = p-1.$$

17. Visszatérünk a 14.-re

18. Növeljük a p paraméter értékét eggyel:

$$p := p + 1$$

és visszatérünk a 4.-re.

19. Mivel a 12.-ben a J_{p+1} halmaz nem volt üres, ezért visszaállítjuk az x_j^{p+1} értékét:

$$x_j^{p+1} := 0$$

és megvizsgáljuk a többi lehetőségeket. Visszatérünk az 5.-re.

20. A korlátként használt érték a célfüggvény optimális értéke és a megfelelő lehetséges megoldás egyúttal *optimális megoldás* is.

Az ismertetett algoritmust már a gyakorlatban is alkalmazzuk a kisfeszültségű villamos hálózat beruházási költségeinek optimalizálására. A gépi progra-

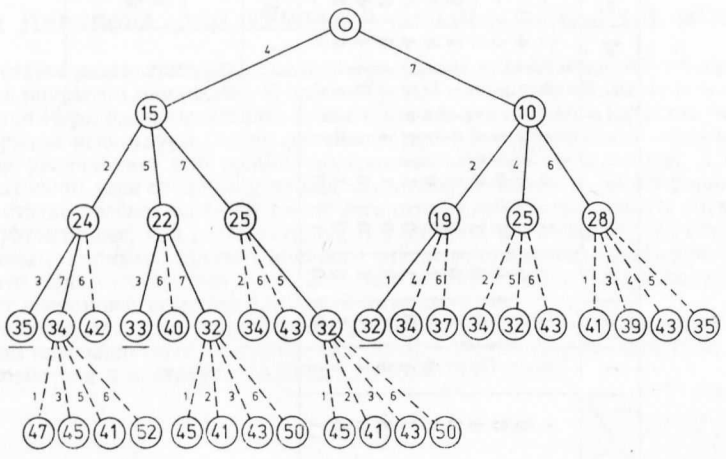
mot a SZÁMGÉP SIEMENS 4004/151 típusú elektronikus számítógépére dolgoztuk ki FORTRAN nyelven. A program kidolgozásánál döntő szempont volt a halmaz lefedési probléma megoldására használt algoritmus memóriaigénye és hatékonysága ritka mátrixok esetén. A rendelkezésünkre álló lineáris programozási feladatokat megoldó könyvtári program memóriaigénye több, mint 100 KB volt. Ez tette szükségessé az ismertetett algoritmus kidolgozását.

Példa: Oldjuk meg az alábbi feladatot:

$$
\begin{aligned}
&13x_1 + 9x_2 + 11x_3 + 15x_4 + 7x_5 + 18x_6 + 10x_7 \to \min \\
&\qquad x_2 + \qquad\quad x_4 + \qquad\quad x_6 \qquad\quad \geq 1 \\
&x_1 + \qquad\quad x_3 + \quad x_4 + \quad x_5 \qquad\quad \geq 1 \\
&x_1 + \qquad\quad\quad x_4 + \qquad\quad x_6 + \quad x_7 \geq 1 \\
&\qquad\quad\quad\quad\quad x_4 + \qquad\quad\quad x_7 \geq 1 \\
&\qquad x_2 + \qquad\quad\quad + \quad x_5 + \qquad\quad x_7 \geq 1 \\
&\qquad\quad\quad x_3 + \qquad\quad\quad\quad\quad x_7 \geq 1 \\
&x_1 + \qquad x_3 + \qquad\quad x_5 + \quad x_6 \qquad\quad \geq 1 \\
&\qquad x_2 + \qquad\quad\quad x_5 + \quad x_6 \qquad\quad \geq 1 \\
&x_1 + \quad x_2 + \quad x_3 + \qquad\quad\quad x_6 \qquad\quad \geq 1 \\
&\qquad x_2 + \qquad\quad\quad x_5 + \qquad\quad x_7 \geq 1 \\
&x_1 + \qquad\quad\quad x_4 + \qquad\quad x_6 \qquad\quad \geq 1 \\
&\qquad\quad x_j \in \{0,1\}, \qquad j = 1,2, \ldots, 7.
\end{aligned}
$$

A 2. teszt alapján az $x^0 = (1,1,0,0,1,0,1)$ lehetséges megoldást kapjuk, amelyhez $z_0 = 39$ célfüggvény érték tartozik.

A megoldás menete jól követhető az 1. táblázaton és az 1. ábrán. Az élek mentén feltüntettük azokat a j indexeket, amelyeknek megfelelő x_j változó



1. ábra

I. táblázat

i \ f	1	2	3	4	5	6	7	α_1^0	α_1^1	α_1^2	α_1^3	α_1^4	α_1^5	α_1^6	α_1^7	α_1^8	α_2^1	α_2^2	α_2^3	α_2^4	α_2^5	α_2^6	α_2^7	α_2^8
1	0	1	0	1	0	1	0	3	—	—	—	—	—	—	—	—	3	—	—	—	—	—	—	—
2	1	0	1	1	1	0	0	4	—	—	—	—	—	—	—	—	4	—	—	—	—	—	—	4
3	1	0	0	1	0	1	1	4	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
4	0	0	0	0	1	0	1	4	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
5	0	0	1	0	0	1	1	2	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
6	0	0	0	1	0	0	1	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
7	1	0	1	0	1	1	0	3	3	3	3	3	3	3	3	3	4	4	4	4	4	4	4	4
8	0	1	0	0	1	1	0	4	4	4	4	4	4	4	4	4	—	—	—	—	—	—	—	—
9	1	1	1	0	0	1	0	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
10	0	1	0	0	1	0	1	3	3	3	3	3	3	3	3	3	4	4	4	4	4	4	4	4
11	1	0	0	1	0	1	0	3	—	—	—	—	—	—	—	—	3	—	—	—	—	—	—	3

x	
$x_4 = 1$	15
$x_2 = 1$	24
$x_3 = 1$	35
$x_4 = 1 \quad x_2 = 1$	24
$x_7 = 1$	34
$x_4 = 1 \quad x_2 = 1$	24
$x_6 = 1$	42
$x_4 = 1$	15
$x_5 = 1$	22
$x_3 = 1$	33
$x_4 = 1 \quad x_5 = 1$	22
$x_7 = 1$	32
$x_4 = 1$	15
$x_7 = 1$	25
$x_5 = 1$	32
$x_7 = 1$	10
$x_2 = 1$	19
$x_1 = 1$	32
$x_7 = 1$	10
$x_4 = 1$	25
$x_7 = 1$	10
$x_6 = 1$	28

értéke 1. A vízszintes vonal azt jelenti, hogy kaptunk egy lehetséges megoldást, amelyhez kisebb célfüggvény érték tartozik, mint a korlát. A körökbe írt szám a célfüggvény értéke, amelyet a megfelelő rendszer határoz meg.

Az optimális megoldás $x = (1,1,0,0,0,0,1)$, amelyhez $z = 32$ célfüggvény érték tartozik.

(Beérkezett: 1975. október 1.)

IRODALOM

1. Garfinkel, R. S.—Nemhauser, G. L.: *Integer Programming.* New York, 1972. John Wiley and Sons.
2. Lemke, C. E.—Salkin, H. M.—Spielberger, K.: *Set covering by single branch enumeration with linear programming subproblems.* Oper. Res. 19 (1971) 998—1022.
3. Forgó, F.: *Egészszámú programozási feladatok néhány transzformációja.* Szigma, VII. évf., 4. sz. 1974.
4. Kovács, L. B.: *A diszkrét programozás kombinatorikus módszerei.* Budapest, 1969.
5. Christofides, N.—Korman, S.: *A computation survey of methods for the set covering problem.* Management Science. Vol. 21, No. 5. 1975.

AN ENUMERATIVE ALGORITHM TO SOLVE THE SET-COVERING PROBLEM

In this paper a special set covering problem of a 0—1 linear program is dealt with. In the case of dense coefficient matrices the methods outlined in the literature proved efficient. The trouble with them is that these methods apply linear programs as subroutines, hence considerably increasing the memory requirements of the computer programme and do not make exploit the peculiarity of this special problem in care of sparse matrices.

The sparser the coefficient matrix, the more efficient the enumerative algorithm introduced in this study is, and it does not require any memory intensive auxiliary procedures (e.g. linear programming). To determine the initial solution a heuristic method is applied yielding near optimal feasible solution.

The introduced algorithm is already employed in practice to optimize the investment costs of low-voltage electric networks. The computer programme has been worked out n FORTRAN language for SIEMENS 4004/151 computer.

АЛГОРИТМ ПЕРЕБОРА ДЛЯ РЕШЕНИЯ ЗАДАЧИ ПОКРЫТИЯ МНОЖЕСТВА

В данной статье рассматривается специальная задача линейного «0—1» программирования — задача покрытия множества. В случае густых матриц методы известные из литературы являются эффективными. Однако в основном все эти методы в качестве вспомогательного алгоритма используют задачу линейного программирования, которая в значительной степени увеличивает для данной программы потребность в памяти, и не используют в достаточной мере специфику данной специальной задачи в случае редких матриц.

В данной статье рассматривается такой алгоритм перебора, который в случае редких матриц тем эффективнее, чем реже матрица, и в котором не используются вспомогательные алгоритмы (например, задача линейного программирования), требующие большого объема памяти. Для нахождения исходного решения используется эвристический метод, который дает возможное решение близкое к оптимальному.

Рассматриваемый алгоритм используется уже на практике при оптимизации капитальных вложений низковольтных электрических сетей. Программа для ЭВМ типа SIEMENS 4004/151 разработана в институте САМГЕП на языке FORTAN.