Néhány tapasztalat a nagyméretű lineáris programozási feladatok megoldásával kapcsolatban

Srajber Benedek

1. Általános megjegyzések

Napjainkban a népgazdaság legkülönbözőbb területein merülnek fel lineáris feltételrendszerrel megfogalmazható optimalizálási feladatok. Ezek megoldására a leggyakrabban használható és jól bevált algoritmusok az egyszerű és módosított szimplex módszer. Ezeket használják fel a speciális nagyméretű Dantzig-Wolfe féle – lépcsős – és láncolt szerkezetű feladatok megoldására szolgáló eljárások is. Az alábbiakban a nagyméretű feladatok szemszögéből nézve vizsgáljuk az említett módszerek hatékonyságát.

Legyen a megoldandó feladat

$$Ax = b$$

(1)

$$c \cdot x \longrightarrow \max ,$$

ahol A m sorból (feltételből) és n oszlopból (változóból) álló mátrix, x a változók, b a szokásos jobboldal, c pedig a célfüggvény koefficienseinek vektora. Tekintetbe kell vennünk a következő korlátozó tényezőket:

a./ a rendelkezésünkre álló gép központi memóriájának kapacitása,

b./ a gép műveleti sebessége,

c./ a külső perifériák viszonylag hosszú elérési ideje,

d./ stabilitási kérdések.

2. Az alapmátrix zéró elemeinek kihasználása

Az a./ nehézségen érdemlegesen akkor segíthetünk, ha az (1) feladat A mátrixában sok a nulla elem, ami egyébként gyakori eset. A táblázat zéró elemeinek kihasználását a nem nulla elemek koordinátás tárolásával oldottuk meg. Egy módosított szimplex változatban az A mátrix tárolását egy n-dimenziós és két k-dimenziós (k az A nem nulla elemeinek száma) vektor bevezetésével helyettesítettük. Az így megoldható feladatok méreteire vonatkozóan k függvényében, (az ICT-1900-as gépet alapul véve és az inverzet mágnesszalagon tárolva), a következő becslések adódnak:

* A szerző 1966-68 évi, GIER és ICT. 1900 számítógépeken szerzett tapasztalatai, amikor még Orchard-Hays "Advanced Linear-Programming Computing Techniques" és E.M.L. Beale "Topics in Operational Research Series" könyve nem állt rendelkezésre.

Helyfoglalás a memóriában: $3k + 13m + 3n +$ program. Célravezető a koordinátás tárolás, ha

$$k < \frac{n \cdot m - n}{2}, \tag{2}$$

ami gyakorlatilag azt jelenti, hogy A elemeinek legalább a fele nulla. Néhány szóbajöhető alkalmazást tüntet fel a következő táblázat, amelynek első sorában összehasonlítás végett a teljes mátrix tárolásának esetét tüntettük fel:

	k	m	n
1.	Teljes táblázat	100	150
2.	6750	100	150
3.	6160	200	300
4.	5000	300	1000
5.	5000	400	600

3. A Dantzig-Wolfe féle dekompozíciós algoritmus programozási tapasztalatairól

A megoldandó feladat:

$$\begin{aligned} A_1x_1 + A_2x_2 + \cdots + A_kx_k &= b_0 \\ B_1x_1 &= b_1 \\ B_2x_2 &= b_2 \\ &\vdots \\ B_kx_k &= b_k \end{aligned}$$

$$c \cdot x \longrightarrow \max$$

ahol a B_i mátrixok szektorokat, az A_i -k pedig a központ szektoroknak megfelelően particionált mátrixai. Az x_i -k a változók particionált vektorai, így az összes változót tartalmazó vektor $x = (x_1, x_2, \ldots, x_k)$; c a célfüggvény együtthatók vektora.

A megoldásra szolgáló algoritmus, amelynek lényege, hogy a teljes megoldást a szektorok megoldásainak konvex kombinációjaként állítja elő, két részre bontható: szektorszintű és központi feladatra. A számításokat ennek megfelelően végeztük el a GIER számológépen.

Ismeretes, hogy a feladat megoldására a módosított szimplex módszer használata a kézenfekvő. Minthogy azonban korábbi tapasztalataink (az ELLIOTT-803 gépen) a módszer konvergenciájának rendkívüli lassúságát igazolták, valamelyest azzal is gyorsítani kívántunk, hogy a szektorszintű számításokat egyszerű szimplex módszerrel végeztük. (Az egyszerű szimplex módszer jelentősen gyorsabb a módosított szimplexnél). Egy viszonylag kisméretű ($m = 50$, $n = 80$) háromszektoros fel-

adatot ALGOL programmal futtattunk le és az több órát vett igénybe. Ugyanaz a feladat az egyszerű szimplexszel 25, a módosított szimplexszel 40 percig tartott. Nyilvánvaló, hogy a tetemes időigény miatt ezres méret körüli feladatok szóba sem jöhettek. A módszer eredményessége csak rendkivül gyors, nagy memóriájú és jó perifériákkal rendelkező géppel biztosítható. A gyors központi memóriával és lemeztárolókkal rendelkező CDC 3300 bizonyára alkalmas lesz a módszer eredményessé tételére.

4. Egy lehetőség nagyméretű feladatok megoldására szimplex módszerrel

Tapasztalatunk azt mutatja, hogy a szimplex algoritmus alkalmazása körülbelül 200 x 200-as méretig a legtöbb esetben pontos eredményt szolgáltat. E fölött azonban egyre inkább jelentkezik az l.d./ probléma. A táblázat stabilitásának megőrzésére több lehetőség van. Például változtatható szóhosszakkal operálni, vagy a módosított szimplex esetén a szorzatalakban tárolt inverz időnkénti ujraképzésével. Az alábbi, egyszerű szimplex módszer esetén alkalmazott fogás bizonyos esetekben ugyancsak célravezetőnek bizonyul.

Abból a megállapításból indulunk ki, hogy a legdurvább hibák a szimplex táblázat transzformációjánál akkor lépnek fel, midőn egy bevonásra kerülő (pivot) oszlop generáló eleme zérushoz nagyon közel eső érték (pl. $< 10^{-6}$ -nál). Egy-két ilyen oszlop bevonása teljesen lehetetlenné teheti az iteráció folytatását. Hasonlóan bajt okozhat, ha a pivot elem túlságosan nagy. Jelentős hiábát okozhat olyan pivot elem is, amely nagyságrendileg erősen különbözik a táblázat elemeitől. Ezeket a nagy hibaforrásokat úgy csökkenthetjük jelentősen, ha az ilyen jellegű pivot elemeket elkerüljük (ha lehetséges!). Ezért minden egyes iterációnál a legjobban javító oszlopok közül, több oszlopot (számuk megadható) veszünk figyelembe, összehasonlítjuk a szóbajöhető generáló elemeket és azon oszlopot vonjuk be a bázisba, amelyre a karakterisztikus hányados 1-től való eltérése a legkisebb.

Pontosabban:

Legyenek $P_1, P_2, \ldots, P_k$ a bevonható oszlopvektorok, koordinátáik p_{ij} -k és x a pillanatnyi bázismegoldás. (Valamennyi vektor mérete m.)

Bevonásra kerül az a P_{j_0} oszlop, amelyre a

$$\min_{(j)} \left| 1 - \min_{\substack{(i) \\ p_{ij}>0}} \frac{x_i}{p_{ij}} \right| \qquad (j = 1, 2, \ldots, k) \qquad (i = 1, 2, \ldots, m)$$

teljesül. Nyilvánvaló, hogy ez a legkedvezőbb választás a következő iterációs lépésre nézve.

Igaz, a válogatás időveszteséggel jár és tény az is, hogy így nem a maximális javítás irányában közeledünk az optimumhoz, de ezt ellensúlyozhatjuk azzal, hogy azon k darab vektort, amelyek közül előzőleg választottunk, megjegyezzük és a következő iterációnál azok közül vonjuk be a legkedvezőbbet. Igy egy-egy válogatást követően több iteráción keresztül is elkerülhetjük az összes oszlop vizsgálatát, ami bőven pótolja az okozott időveszteséget.

Irodalom

[1] Walter W. Garvin, Introduction to Linear Programming Mc Graw-Hill Book Company, 26-37 191-203, 1960.

[2] Dantzig, Linear Programming and Extensions Princeton, 1963.

[3] Dantzig – Wolfe, Decomposition Principle for Linear Programs. Operations Research 8: 101-111 (1960).

[4] G. Hadley, Linear Programming. Addison-Wesley Publishing Company, 1962. Addison-Wesley Publishing Company, 1962.

Summary

Some experiences in connection with solving linear programming problems of large size.

The paper deals with the utilization of the zero places of a large matrix, while the non-zero elements of the matrix are stored in co-ordinata form in the memory.

The third point involves some practical experiences about the algorithm of Dantzig–Wolfe. The last part gives a possibility for solving problems of large size with the simplex method. This possibility includes the selection of the pivot element maintaining the stability of the problem and simultaneously entering the basis with more than one variable.

Резюме

Несколько опытов, связанных с решением задач линейного программирования большого размера.

Статья занимается вопросом оптимального использования наличия в матрицах большого порядка большого числа нулевых элементов; ненулевые элементы матрицы содержатся в памяти машины в координатном виде.

В третьем пункте находится описание нескольких практических опытов, связанных с алгоритмом Данцига-Волфа.

Последняя часть показывает одну возможность для решения задач большого размера с помощью симплексного метода. Эта возможность содержит в себе выбор такого пивот-элемента, который сохраняет устойчивость задачи и вступление в базис больше одного переменного при одной итерации.