Bakó András—Kas Péter

A k-adik legrövidebb útvonal meghatározása mátrix módszerrel

A dolgozat első részében áttekintjük a k-adik legrövidebb útvonal feladat megoldási módszereit. A második rész tartalmazza a feladat megoldására adott új algoritmust és az eljárás jogosságát. A módszer az összes pontpár között kn^3 összeadással és $k^{3/2}\ n^3$ összehasonlítással adja meg az első k legrövidebb utat.

Az irodalomban az általunk megoldott és a gyakorlatban is felmerülő feladatra nem közölnek eljárást. Ha a két pont közötti legrövidebb út eljárások valamelyikét alkalmazzuk erre a feladatra, akkor az egy nagyságrenddel több számítási lépést igényel, mint az itt közölt módszer.

1. Bevezetés

Gyakorlati feladatok megoldásánál sokszor szükséges az optimális megoldás mellett az optimumhoz közel álló megoldások ismerete is.

Így a k-adik legrövidebb út feladatának megoldása tervütemhálók esetén a különböző tervvariánsok kiszámítására, az objektum építése során bekövetkezett időelcsúszások analizálására, valamint a teljes átfutási idő redukciójára használható. Mivel tervütemhálóban kör nincs, a feladat megoldása is egyszerű. Az összeadások és összehasonlítások száma megegyezik a CPM-time feladat megoldásához szükséges lépések számával. A tárolási igény természetesen nő: minden egyes ponthoz hozzá kell rendelni egy cimke vektort és egy potenciál vektort, amelyek hossza éppen k (lásd *Bakó* [1]).

A feladatok nagy részénél a környélküliség nincs biztosítva, és ekkor a megoldás is bonyolultabb. Ez fordul elő például az úthálózatok tervezésénél, az ún. ráterhelési feladatnál. Itt ismert a megadott vagy tervezett úthálózat minden i pontjából minden j pontjába a forgalom és meg kell határozni az útszakaszokon folyó forgalmat. A megoldás egyik módszere a pontpárok közötti első k legrövidebb útvonal ismeretén alapszik.

A forgalom egy részét az első, egy másik részét a második, végül az utolsó részét a k-adik legrövidebb úthoz rendelik hozzá.

A 2. pontban megadjuk a probléma matematikai megfogalmazását és összefoglaljuk a megoldási módszereket. A 3. pontban írjuk le az új algoritmust, és bebizonyítjuk az eljárás helyességét. A 4. pontban egy hatékony tárolási módszert adunk meg és néhány számítástechnikai megjegyzést teszünk.

2. A feladat megfogalmazása és megoldási módszerei

Legyen $N = \{x_1, x_2, \ldots, x_n\}$ egy irányított gráf (digráf) pontjainak, $E = \{(x_i, x_j)\}$ az irányított éleinek a halmaza. Legyen megadva a digráf élein egy $t(x_i, x_j) \geq 0$, $(x_i, x_j) \in E$ távolságfüggvény. Az (N, E, t) együttest hálózatnak nevezzük. Legyen $P = (x_1, x_2, \ldots, x_m)$ az x_1 pontból az x_m pontba vezethető út.

A P út $l(P)$ hosszán az alábbiakat értjük:

$$l(P) = \sum_{i=1}^{m-1} t(x_i, x_{i+1}), \tag{1}$$

A k-adik legrövidebb út feladata: határozzuk meg egy rögzített x_1 pontból egy másik rögzített x_m pontba az első k legrövidebb útvonal hosszát, és magukat az útvonalakat is.

A feladat megoldási módszerei két csoportra oszthatók. Az első csoportba tartozó módszerek olyan utakat adnak, amelyek kört is tartalmazhatnak. Ekkor a feladat megoldása viszonylag egyszerű, de a kör kiszűrése további fáradsággal jár. Az első ilyen algoritmust *Hoffmann—Pavley* [7] közölte 1959-ben a 2. legrövidebb út megoldására. A módszernél a 2. legrövidebb út meghatározásához $M \cdot K$ művelet szükséges, ahol M az egy pontból kimenő élek átlagos száma, K a legrövidebb út éleinek a száma. Ezt a módszert általánosította a k-adik legrövidebb út meghatározására *Pollack* [14] és *Dreyfus* [4]. Egy érdekes dinamikus programozási módszert közöl *Bellman—Kalaba* [2], amely viszont több számolást igényel, mint az eredeti *Hoffmann—Pavley* módszer.

Ha kör nem lehet a meghatározott utakban, az algoritmus is bonyolultabb. Az első kör nélküli algoritmust *Clarke—Krikorian—Rausen* [3] készítette 1963-ban. A feladat megoldásához a korlátozás és szétválasztás módszerével az összes, bizonyos feltételeknek eleget tevő utat végig kell nézni. Ezek száma nagyméretű digráfok esetén nagy, így a módszer gyakorlatilag általában alkalmazhatatlan.

Az első, gyakorlatban is használható algoritmust *Yen* [18] adta. Ennél az algoritmusnál az összeadások és összehasonlítások száma $\mathcal{O}(k \cdot n^4)$ és $\mathcal{O}(k \cdot n^3)$ közé esik. A módszert *Lawler* [10] egyszerűsítette és a lépésszámot $\mathcal{O}(kn^3)$-re szorította le. A legutóbb publikált *Weigand* [16] algoritmus nagy gráfok esetén is gyors, de az algoritmus bonyolultsága miatt nehezen programozható.

Annak eldöntésére, hogy a digráf tartalmaz-e kört egy sor eljárás ismeretes (lásd például *Klein* [9], *Florian* [5], és *Yen* [19]). *Yen* [18] dolgozatában a *Yen*, *Pollack* [13], *Bock*, *Clarke* [3] és *Sakarovich* módszereit hasonlítja össze a szükséges összeadások, összehasonlítások és memóriaigény szempontjából.

A k-adik legrövidebb út feladatának megoldási módszereiről áttekintést ad *Pollack* [13] és *Pierce* [12]. A legrövidebb út feladata és a k-adik legrövidebb út feladata átfogalmazható hozzárendelési feladatra is. Ehhez rögzítsük le a hálózat $\mathcal{L}$ pontját; legyen ez x_1 és x_n és vegyük hozzá a hálózathoz az (x_n, x_1) élt egy nagy negatív szállítási költséggel. A hálózatban a minimális hosszúságú kör éppen az x_1-ből x_n-be vezető legrövidebb útból és az (x_n, x_1) élből áll. Ezen kör megkeresése ekvivalens egy hozzárendelési feladat megoldásával.

A feladat fenti megközelítését *Murty* [11] vetette fel. *Weintraub* [17] a k-adik legrövidebb út feladatának megoldását az optimális megoldáshoz

tartozó T mátrix egy sorának megváltoztatásával és az új optimum meghatározásával számolja ki, legfeljebb n^2 elemi operációval.

A következő fejezet egy új eljárást ismertet, amely mátrix eljárással egyszerre az összes pontpár között megadja a k-adik legrövidebb útvonalat.

3. Multiterminális feladat megoldása

A bevezetőben és a 2. fejezetben ismertetett módszerek a feladatot két kitüntetett pont között oldják meg, vagy egy pontból az összes többi pontba adják meg a k-adik legrövidebb utat. A multiterminális feladatnál az összes pontpár közötti k-adik legrövidebb utakat kell meghatározni. Ez megoldható az eddig ismertetett módszerek bármelyikével, de a leggyorsabb algoritmust is n-szer kell végigszámolni. Így a megoldáshoz legalább $\mathcal{O}(kn^4)$ összeadásra és összehasonlításra van szükség.

Az alábbiakban ismertetünk egy gyors mátrix eljárást, amely összesen $\mathcal{O}(kn^3)$ összeadásból és összehasonlításból áll. A feladat megoldásakor kört tartalmazó utak is adódhatnak, és egyenlő hosszúságú utak is előfordulhatnak a meghatározott utak között.

3.1. Megoldási algoritmus

Jelölje $t_{ijk}^{(l)}$ az x_i pontból az x_j pontba vezető k-adik olyan legrövidebb út hosszát, amely közbülső pontokat legfeljebb az $N^{(l)} = \{x_1, x_2, \ldots, x_l\}$ halmazból tartalmazhat.

A feladat megoldása során rendre meghatározunk $t_{ijk}^{(1)}, t_{ijk}^{(2)} \ldots$ mennyiségeket és megmutatjuk, hogy $t_{ijk}^{(n)}$ a multiterminális feladat megoldása.

Jelöljük a $v_1, v_2, \ldots v_r$ egész számok p-edik legkisebb elemét $\min_p(v_1, v_2, \ldots, v_r)$-el.

Az algoritmus ezek után a következő lépésekből áll:

kiinduló lépés

$$t_{ijl}^{(0)} = t(x_i, x_j), \qquad i, j = 1, \ldots, n \tag{3.1}$$

$$t_{ijp}^{(0)} = \infty, \qquad p = 2, 3, \ldots, k$$

l-edik lépés

$$t_{ijp}^{(l)} = \min_p \left(\{t_{ijq}^{(l-1)} | q = 1, 2, \ldots, k\}, \{t_{ilq'}^{(l-1)} + t_{ljq''} | q', q'' = 1, 2, \ldots k\}\right) \tag{3.2}$$

Tétel: a fenti algoritmus eredményeképp kapott $t_{ijp}^{(n)}$ a p-edik legrövidebb út hossza.

Bizonyítás: A tételt az l szerinti teljes indukcióval mutatjuk meg.

$l = 1$ esetben legfeljebb két utat kaphatunk, amelyek hossza az alábbi két mennyiség

$$t_{ij1}^{(1)} = \min_1 (t_{ij}, t_{i1} + t_{1j})$$

$$t_{ij2}^{(1)} = \min_2 (t_{ij}, t_{i1} + t_{1j})$$

l — 1-re feltesszük a tétel állítását, azaz, hogy a $t_{ijp}^{(l-1)}$ mennyiség az x_i pontból az x_j pontba vezető, $N^{(l-1)}$ halmazból közbülső pontokat tartalmazó utak közül a p-edik legrövidebb út hossza. l esetére megmutatjuk a tételt.

Tegyük fel, hogy van olyan P^* út az x_i pontból az x_j pontba, amely közbülső pontokat legfeljebb csak az $N_{(l)}$ halmazból tartalmaz

de $$l(P^*) < t_{ijk}^{(l)} \tag{3.3}$$

és $$l(P^*) \neq t_{ijp}^{(l)}, \qquad p = 1, 2, \ldots, k. \tag{3.4}$$

Ha $l \notin P^*$, akkor P^* közbülső pontokat csak az $N^{(l-1)}$ halmazból tartalmazhat, így a minimum kiválasztásakor a (3.2)-ben az első számsorból kellett volna választanunk (3.3) miatt, így $l \in (P^*)$. Ekkor viszont (3.2)-ben a második tagban kellett lennie, de mi a minimum számolásakor nem ezt választottuk.

3.2. Számítástechnikai megjegyzések

A 3.1 pontban elmondott algoritmus csak az útvonalak hosszát adja meg. A cimkézési technika megfelelő módosításával azonban maguk az útvonalak is meghatározhatók.

Legyen $H = (h_{ijp}^{(l)})$ a cimkemátrix.

A kezdeti lépéskor $h_{ijp}^{(0)} = j$, $i = 1, 2, \ldots, n$; $j = 1, 2, \ldots, n$.

Az l-edik lépésben a cimkemátrix elemeit a következőképp módosítjuk

$$h_{ijp}^{(l)} = \begin{cases} h_{ijp}^{(l-1)}, & \text{ha } t_{ijp}^{(l)} = t_{ijp}^{(l-1)} \\ h_{ilp}^{(l-1)} & \text{egyébként.} \end{cases} \tag{3.5}$$

A címmátrix segítségével tetszés szerinti két pont között meghatározhatjuk a p-edik legrövidebb útvonal közbülső pontjait. Tegyük fel, hogy az s pontbó: kiinduló és a t pontban végződő útvonal meghatározása a feladat.

Az útvonal visszakeresése az alábbi lépésekből áll:

— az s pont utáni első pont a p-edik legrövidebb úton a $h_{stp} = c_1$;
— a c_1 pont után következő pont $h_{c_1tp} = c_2$, . . .

A fenti eljárás akkor ér véget, ha elérünk a t pontba, azaz ha valamelyik c_i-re $h_{c_itp} = t$. Az eljárás eredményeképp megkaptuk az s-ben kezdődő t-ben végződő és az $s = c_0, c_1, c_2, \ldots c_i = t$ pontokból álló p-edik legrövidebb útvonalat.

A 3.2-ben leírt képlet szerint minden elem kiszámításához k^2 összeadásra, és $k^2 + k$ elemű vektor rendezésére van szükség. A 4.2. első, legfeljebb k eleme nagyság szerint növekvő sorrendben van. A második rész két összeadandója szintén monoton nő. Ezt a tulajdonságot kihasználva nem szükséges a k^2 összeadást és a vektor rendezését elvégezni, elegendő egy sokkal egyszerűbb feladat ötletes megoldása. Tekintsünk két nagyság szerint növekvő sorrendben rendezett számsort az $a_1, a_2, \ldots a_m$ és a $b_1, b_2, \ldots b_m$ vektorokat. Képezzük a két vektor diadikus összegét, azaz az alábbi táblázatot:

	b_1	b_2		b_m
a_2	$a_1 + b_1$	$a_1 + b_2$	...	$a_1 + b_m$
a_1	$a_2 + b_1$	$a_2 + b_2$	...	$a_2 + b_m$
.	.....			
.	.....			
.	.....			
a_m	$a_m + b_1$	$a_m + b_2$	...	$a_m + b_m$

A feladat: határozzunk meg minimális számú elemi művelettel (összeadással és összehasonlítással) a tábla első t legkisebb elemét.

A következő eljárással kapjuk a megoldást:

Az első elem nyilván $a_1 + b_1$. A második elem $a_1 + b_2$ és $a_2 + b_1$ közül kerül ki, legyen mondjuk $a_1 + b_2$. A következő elem $a_1 + b_3$, $a_2 + b_1$ minimuma. mondjuk $a_2 + b_1$. A negyedik elem választásához már $a_1 + b_3$, $a_3 + b_1$ és az $a_2 + b_2$ elemeket is össze kell hasonlítani.

Így 3.2 összeadást tartalmazó második részét minimálisan $k + 1$ összeadással és összehasonlítással, maximálisan $k + 1$ összeadással és $k\sqrt{k}$ összehasonlítással kaphatjuk meg. A két rendezett sorozat összefésülését a diadikus összegek meghatározása során $\Theta(k)$ lépésben végezhetjük el. Így a feladat megoldásához szükséges minimális lépésszáma következő: Egy elem meghatározásához minimálisan $k + 1$ összeadás és $2k$ összehasonlítás, egy lépésben $(k + 1)(n - 1)^2$ összeadás és $2k(n - 1)^2$ összeadás, a feladat megoldásához pedig $n(k + 1)(n - 1)^2$ összeadás és $2kn(n - 1)^2$ összehasonlítás szükséges. A műveletek felső korlátja: $n(k + 1)(n - 1)^2$ összeadás és $nk\sqrt{k}(n - 1)^2$ összehasonlítás.

Csak központi memóriát használó algoritmushoz a tárolási igény $2k \cdot n^2$, mivel n^2 elem kell egy táblázathoz, de k mátrixunk van és a cimkemátrixok tárolásához további kn^2 tárolóhely szükséges. Nagyméretű hálózatok esetén két módosítási lehetőség kínálkozik:

a) Alkalmazzuk a *Hu—Torres* [8] által javasolt dekompozíciós sémát. Ez a tárolási igényt nagymértékben lecsökkenti, és a számítási lépések száma is kevesebb lesz. Hátránya viszont, hogy a programozása meglehetősen bonyolult.

b) A mátrixok sorait és oszlopait is külső tárolón tartjuk. Az algoritmust soronként végezzük, és egyszerre csak az egy sor kitöltéséhez szükséges vektorokat tartjuk bent. Ez összesen $2n^2k$ vektor memóriába való mozgatását és ugyanannyi külső memóriába való írást jelent.

A kör kiszűrésére az alábbi eljárást alkalmazhatjuk: minden lépésben végignézzük, hogy a két részből összerapott utak nem tartalmaznak-e kört, és ha igen, úgy a kört tartalmazót kihagyjuk és vesszük a következő, nála hosszabb utat. Ez azt jelenti, hogy r kör esetén a 3.2 lépésben nem k legkisebb elemet kell meghatároznunk, hanem $k + r$ elemet. Azokat az értékeket, amelyek kört tartalmazó úthoz tartoznak, ki kell hagyni az út mátrixából. Hasonlóképp ki kell törölni az ezen elemekhez tartozó cimkemátrix elemeket is. Ez az eljárás növeli az elemi operációk számát.

A multiterminális eljárás városi és közúti úthálózat tervezésének egyik modelljéhez szükséges. Gyakorlati feladatok megoldásához elkészítettük a gépi programját is FORTRAN nyelven a CDC 3300 számítógépre.

(Beérkezett: 1976. augusztus 15.)

IRODALOM

[1] Bakó, A.: *Tervütemhálók sub-kritikus útjainak meghatározása.* Alkalmazott Matematikai Lapok (megjelenés alatt).
[2] Bellman, R.—Kalaba, R.: *On the Kth Best Policies.* Journal of SIAM 8(1960), 582—588.
[3] Clarke, S.—Krikorian, A.—Rausen, J.: *Computing the N Best Loopless Path in a Network.* Journal of SIAM 11(1963), 1196—1202.
[4] Dreyfus, S. E.: *An Appraisal of Some Shortest Path Algorithms.* Operations Research 17(1969), 395—412.
[5] Florian, M.—Robert, P.: *A Direct Search Method to Locate Negative Cycles.* Management Science 17(1971).
[6] Floyd, R. N.: *Algorithm 97, Shortest Path.* Communication ACM 5(1962), 345.
[7] Hoffman, W.—Pavley, R.: *A Method for the Solution of the Nth Best Path Problem.* Journal of ACM 6(1959), 506—514.
[8] Hu, T. C.—Torres, W. T.: *Shortcut in Decomposition Algorithm for Shortest Paths in a Network.* IBM Journal of Research and Development 13(1969), 387—390.
[9] Klein, M.—Tibrewala, R. K.: *Technical Contributions Finding Negative Cycles.* INFOR 11(1973), 59—65.
[10] Lawler, E. L.: *A Procedure for Computing the K Best Solutions to Discrete Optimization Problems and its Application to the Shortest Path Problem.* Management Science 18(1972) 401—405.
[11] Murty, K. G.: *An Algorithm for Ranking All the Assignments in Increasing Order of Cost.* Operations Research 16(1968), 682—687.
[12] Pierce, A. R.: *Bibliography on Algorithms for Shortest Path, Shortest Spanning Tree, and Related Circuit Routing Problems (1956—1974).* Networks 5(1975), 129—149.
[13] Pollack, M.: *Solution of the k-th Best Route Through a Network.* A Review, Journal of Mathematical Analysis and Application 3(1961), 547—559.
[14] Pollack, M.: *The k-th Best Route through a Network.* Operations Research 8(1960), 224—230.
[15] Warshall, S.: *A Theorem on Boolean Matrices.* Journal of ACM 9(1962), 11—12.
[16] Weigand, M. M.: *Ein neuer Algorithmus zur Bestimmung von k-kürzesten Wegen in einem Graphen.* Computing 16(1976), 139—151.
[17] Weintraub, A.: *The Shortest and K-Shortest Routes as Assignment Problems.* Networks 3(1973), 61—73.
[18] Yen, J. Y.: *Finding the K Shortest Path in a Network.* Management Science 17(1971), 712—716.
[19] Yen, J. Y.: *On the Efficiencies of Algorithms for Detecting Negative Loops in Networks.* Santa Clara Business Review 2(1971), 52—58.

DETERMINING THE K-TH SHORTEST PATH BY MATRIX METHOD

In the paper an algorithm is given for the multiterminal version of the k-th shortest path problem.

The algorithm calculates in n steps the length of the first k paths between all the point pairs of a digraph.

We describe the labelling technique required for the determination of the paths and the procedure supplying the intermediate points belonging to the paths.

Altogether the calculation consists of kn^3 additions and $k^{3/2}n^3$ comparisons. This number is considerably lower than what is necessary for the repeated application of the k-th shortest ath palgorithm of two end points known in literature.

ОПРЕДЕЛЕНИЕ НАИБОЛЕЕ КОРОТКОГО ПУТИ «К» МЕТОДОМ МАТРИЦЫ

В рассматриваемой работе дается один алгоритм на мультитерминальный вариант наиболее короткого пути «к».

Данный алгоритм в действии «n» расчитывать протяженность наиболее короткого пути «$к$» между всеми парными точками диграфа.

Описывается техника установления наименований, необходимых для определения путей, а также и методика определения промежуточных пунктов, относящихся к трассам.

Собствоенно расчет складывается из суммирования kn^3 и сравнения $k^{3/4}\ n^3$ Эта цифра является значительно меньшей, чем это необходимо для многократного использования известных по литературе алгоритмов наиболее короткого пути «$к$» с дувмя конечными точками.