UDK 528.33:528.143
Pregledni rad

GENERISANJE USLOVNIH JEDNAČINA KOD IZRAVNANJA NESLOBODNIH GEODETSKIH MREŽA

Zdravko GALIĆ — Sarajevo*

1. UVOD

U procesu izravnanja geodetskih mreža, korisnicima računarskih programa često predstoji obiman posao u definisanju odgovarajućih odnosa između mjerenih i datih veličina, potrebnih odgovarajućim matematskim modelima. Pored toga što takav postupak zahtijeva dodatni napor od korisnika, greške u ovoj fazi rada, kao što su pogrešno određivanje nekog od elemenata matrice koeficijenata jednačina grešaka, vektora slobodnih članova ili težine nekog mjerenja, neminovno dovode do pogrešnih rezultata i do ponavljanja procesa izravnanja. Za razliku od posrednog izravnanja nekih vrsta geodetskih mreža, gdje je angažovanje korisnika softvera u procesu izravnanja moguće svesti na minimum, to kod izravnanja geodetskih mreža uslovnom metodom nije slučaj. Generalno govoreći, korisnik mora kod nekih vrsta mreža izvršiti određenu generalizaciju mjerenja (poligonska i nivelmanska mreža), zamjenjujući niz mjerenja jednim, identifikovati sve potrebne nezavisne uslove i postaviti uslovne jednačine, tj. odrediti vrijednosti elemenata nekih matrica (A, P, W). U tom slučaju softver se u osnovi sastoji od niza sekvencijalnih operacija nad matricama. Očigledno je, a naročito kada su u pitanju veće mreže, koliki je napor neophodno uložiti prije nego se i počne koristiti takav softver, odnosno računar. Zajednička osobina svih geodetskih mreža sa datim veličinama jeste, da je za 'n' datih veličina neophodno postaviti određen broj uslovnih jednačina između njih. Bez obzira na vrstu mreže, u opštem slučaju neophodno je između 'n' tačaka sa nekim poznatim veličinama pronaći 'n—1' puteva koji povezuju te tačke.

U literaturi se često preporučuje da se izaberu takve uslovne jednačine koje sadrže što manji broj mjerenja. Ova sugestija proizilazi iz težnje za što manjim brojem operacija, a korisno ju je prihvatiti i zbog eventualnog razvoja softvera, čiji algoritmi operišu nad rijetkim matricama, tj. matricama koje posjeduju mali broj elemenata, čije su vrijednosti različite od nule, a ta je osobina inherentna matricama koje se pojavljuju kod izravnanja geodetskih mreža.

* Adresa autora: Zdravko Galić, dipl. ing. geod., Građevinski fakultet, Sarajevo, Hasana Brkića 24

U ovom radu će biti prikazan jedan algoritam, potrebne strukture podataka i njegova implementacija, kojim je moguće generisati sve uslovne jednačine između poznatih tačaka u mreži. U širem smislu, algoritam je pogodan za razvoj takvog sofvera, čije korištenje zahtijeva minimalno angažovanje korisnika, čak i slabije upućenog u problematiku samog izravnanja.

2. ALGORITAM

Reprezentujemo li geodetsku mrežu planarnim, neorijentisanim grafom, tada je pomenuti problem pronalaženja 'n—1' puteva koji povezuju tačke sa poznatim veličinama (uz uslov da ti putevi sadrže minimalan broj mjerenja), sličan problemu određivanja najkraćeg puta između dva proizvoljna čvora na grafu. Koristeći pristup rješenju ovog problema u [3], može se u apstraktnoj formi opisati algoritam za generisanje uslovnih jednačina između poznatih tačaka u mreži:

Neka skup S sadrži one tačke u mreži za koje je pronađeno najmanje mjerenja od jedne proizvoljne date tačke 'tØ' (tØ $\in$ S). Za neku tačku 'p' (p $\notin$ S), neka je d[p] minimalan broj mjerenja između 'tØ' i 'p' koja povezuje samo one tačke koje se nalaze u skupu S. Uočimo sljedeće:

(i) Ako je naredno mjerenje prema nekoj tački 'u', tada sva mjerenja prolaze kroz tačke koje su već u S. Da bi dokazali ovu tvrdnju, pretpostavimo da postoji tačka 'p' na ovakvom putu koji predstavlja minimalan broj mjerenja (p $\notin$ S). Tada put od 'tØ' do 'u' sadrži i sva mjerenja od 'tØ' i 'p', koji naravno sadrži manji broj mjerenja nego put od 'tØ' do 'u'. Pretpostavljajući da je najmanji broj mjerenja generisan u njihovom rastućem poretku, skup mjerenja od 'tØ' do 'p' već je morao biti generisan, pa stoga ne može postojati neka tačka koja već ne pripada skupu S.

(ii) Završetak sljedećeg dijela puta, tj. sljedeće mjerenje mora biti prema tački 'u', čiji broj mjerenja d[u] je najmanji od svih tačaka koje se ne nalaze u S. Ovo je posljedica definicije 'd' i stava (i). U slučaju da postoji više tačaka koje ne pripadaju skupu S, a do kojih je isti broj mjerenja od 'tØ', može se izabrati bilo koja između njih.

(iii) Tačka 'u' postaje element skupa S. Broj mjerenja koji povezuju tačku 'p' sa tačkom 'tØ', i koja prolaze kroz tačke iz skupa S, može se smanjiti. Ovo je posljedica toga da su to mjerenja od 'tØ' kroz 'u' do 'p', gdje je put od 'tØ' do 'u' put sa minimalnim brojem mjerenja, a dužina puta od 'u' do 'p' je 1 (jedno mjerenje).

3. IMPLEMENTACIJA

Implementacija svakog algoritma zavisi od korištenih struktura podataka nad kojima taj algoritam operiše, i treba ih posmatrati jedinstveno, s obzirom da ne mogu egzistirati samostalno. Specifičnost ove implementacije se prvenstveno ogleda u tome, što je korišten relacioni model podataka za reprezentaciju svih relevantnih entiteta geodetskih mreža.

Naime, korišten je višekorisnički relacioni sistem za upravljanje bazama podataka EQUAL* (EQL) i relacioni programski jezik Pascal/E. Glavna odlika ovog jezika je mogućnost razvoja relacije kao struktuiranog tipa podataka. Relacija u Pascalu/E ima osobine slične kombinovanim osobinama datoteke (file) i skupa (set), a sam jezik posjeduje bogat skup primitivnih operacija nad relacijama za izražavanje kompleksnih upita i ažuriranja. Specifikacija i kreacija relacija, njihovih primarnih ključeva, uslova integriteta i pristupnih prava su omogućena unutar EQUAL jezika za definisanje podataka, i ovdje neće biti posebno prikazane.

Za potrebe samog algoritma, geodetsku mrežu reprezentujemo jednom složenijom strukturom podataka 'geodetskamreza', čiji razvoj možemo opisati narednom notacijom:

```
const  brojtacaka = 100;

type   string5    = array[1..5] of char;

       mjerenje   = record
                      broj    : integer;
                      tacnost : real;
                      d       : integer
                    end;

       tacka      = record
                      tacka# : string5;
                      tip    : string5;
                      x,y,h  : real
                    end;

       niztacaka  = array[1..brojtacaka] of tacka;

       matrica    = array[1..brojtacaka,1..brojtacaka] of mjerenje;

       geodetskamreza = record
                          tacke : niztacaka;
                          veze  : matrica
                        end;
```

Druga specifičnost ove implementacije je korištenje jedne diskretne matematske strukture (stablo) u cilju generisanja uslovnih jednačina. S obzirom na potrebu specifičnog pretraživanja ove strukture, stablo reprezentujemo na sljedeći način:

```
cvorovi = record
            cvor : tacka;
          prethodnik : string 5
          end;
tipstabla = array [i... brojtacaka] of cvorovi;
```

(Struktura 'geodetskamreza' se generiše za svaki tip mreže na osnovu sadržaja baze podataka).

*EQUAL i Pascal/E su projektovani na osnovu [2] i implementirani na računaru VAX 11/750 sa operativnim sistemom VMS v. 4.1 Implementacija pod operativnim sistemom UNIX je u toku.

Prethodno opisani algoritam prikazan je u preciznoj formi kao procedura programskog jezika Pascal/E.

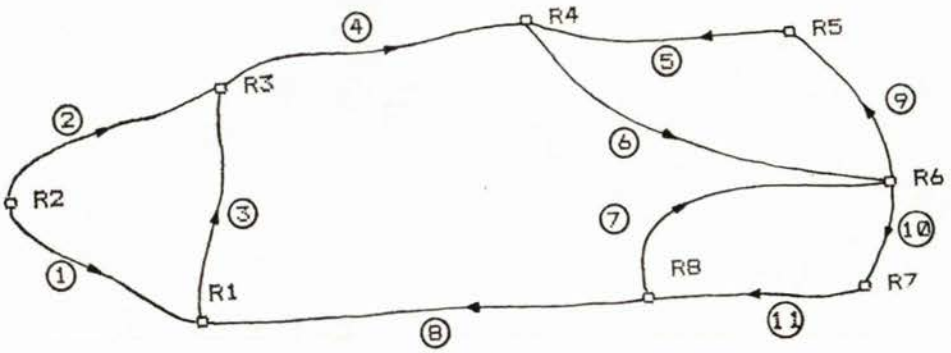
Za ilustraciju, razmotrićemo primjenu ovog algoritma kod izravnanja neslobodnih nivelmanskih mreža. Nivelmansku mrežu možemo reprezentovati sljedećim relacijama u relacionom modelu podataka:

VLAK (VLAK#, POCETNATACKA#, ZAVRSNATACKA#, DUZINAVLAKA, BROJSTANICA)

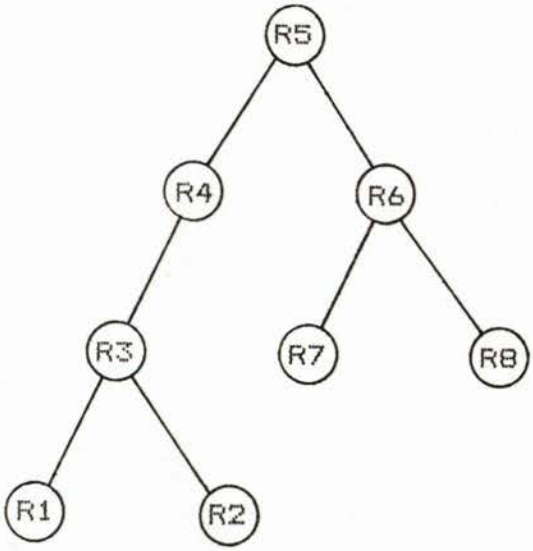
TACKA (TACKA#, TIP, VISINA)

VISINSKARAZLIKA (OD#, DO#, MJERENJE, DUZINA, STANICE, VLAK#, P#1, P#2)

Očigledno je da su i vlakovi i poligoni u potpunosti definisani visinskim razlikama. Naime, atributi relacije VISINSKARAZLIKA određuju pripadnost svake visinske razlike jednom vlaku, odnosno jednom (ili nijednom) ili dva poligona.



Sl. 1.



Sl. 2.

```
procedure generisiuslove(var mreza   : geodetskamreza;
                         var stablo  : tipstabla;
                         korijen     : string5;
                         n           : integer       )

const    nil          =       '';
         velikibroj   =  10E16;
         maxbroj      = 100      (* max. broj tacaka u mrezi *)

type     logickiskup = array[1..maxbroj] of boolean;
         duzine      = array[1..maxbroj] of integer;

var      s            : logickiskup;
         d            : duzine;
         nadjen       : boolean;
         u,v,w,i,j,
         broj,min     : integer;
         privremeno   : tipstabla;

function indeks(tacka# : string5) : integer;
begin
     i:=1;
     nadjen:=false;
     repeat
          if mreza.tacke[i].tacka# = tacka#
          then begin
                    indeks:=i;
                    nadjen:=true
               end
          else i:=i+1
     until nadjen
end;

begin
     for i:=1 to n do
        begin
          s[i]:=false;
          privremeno[i].cvor.tacka#:=mreza.tacke[i].tacka#;
          privremeno[i].prethodnik:=nil;
          for j:=1 to n do
             if mreza.veze[i,j].d = 0
             then mreza.veze[i,j].d:=velikibroj
             else mreza.veze[i,j].d:=1
        end;
        v:=indeks(korijen);
        for i:=1 to n do
           d[i]:=mreza.veze[v,i].d;
        s[v]:=true;
        d[v]:=0;
        stablo[1].cvor.tacka#:=korijen;
        broj:=2;
        while broj < n do
             begin
                  u:=1;
                  nadjen:=false;
                  repeat
                       if not s[u]
                       then begin
                                 nadjen:=true;
                                 min:=d[u]
                            end
                       else u:=u+1
                  until nadjen;
```

```
                for i:=1 to n do
                   if (d[i] < min) and (not(s[i])
                   then begin
                              u:=i;
                              min:=d[i]
                        end;
                s[u]:=true;
                stablo[broj].cvor.tacka#:=mreza.tacke[u].tacka#;
                for w:=1 to n do
                   if not s[w]
                   then if (d[u]+mreza.veze[u,w].d) < d[w]
                        then begin
                                   d[w]:=d[u]+mreza.veze[u,w].d;
                                   privremeno[w].prethodnik:=
                                   stablo[broj].cvor.tacka#
                             end;
                broj:=broj+1
             end; (* while *)
          i:=1;
          nadjen:=false;
          while not nadjen do
               if not s[i]
                 then nadjen:=true
                 else i:=i+1;
          stablo[broj].cvor.tacka#:=mreza.tacke[i].tacka#;
          for i:=1 to n do
             if privremeno[i].prethodnik = nil
               then privremeno[i].prethodnik:=korijen;
          for i:=1 to n do
             begin
                  nadjen:=false;
                  w:=1;
                  repeat
                        if stablo[i].cvor.tacka# =
                           privremeno[w].cvor.tacka#
                        then begin
                                   nadjen:=true;
                                   stablo[i].prethodnik:=
                                   privremeno[w].prethodnik
                             end
                        else w:=w+1
                  until nadjen
             end;
          stablo[1].prethodnik:=nil
end;
```

Za nivelmansku mrežu prikazanu na slici 1., u kojoj su poznate vrijednosti visina repera R5 i R1, prikazanim algoritmom se generiše stablo, prikazano na slici 2. Čvorovi ovog stabla reprezentuju čvorne tačke mreže, a grane stabla vlakove koji povezuju ove tačke, tako da put od svakog čvornog repera u mreži do repera R5 sadrži minimalan broj mjerenja (vlakova).

Jednostavnim pretraživanjem ovakve strukture određuju se mjerenja koja ulaze u odgovarajuće uslovne jednačine, odnosno koeficijenti jednačina grešaka i slobodni članovi. U ovom slučaju potrebno je pretražiti stablo od trećeg nivoa, na kome se nalazi čvor R1 do korijena stabla (R5). Koeficijenti jedna-

čina grešaka se dobiju poređenjem čvorova grana stabla na tom putu sa odgovarajućim vrijednostima atributa POCETNATACKA i ZAVRSNATACKA relacije VLAK (ako su isti, koeficijent ima vrijednost 1, a ako nisu —1). Bez obzira na način reprezentacije stabla, neophodno je obezbijediti takvu strukturu podataka za njegovu reprezentaciju, koja omogućuje pretraživanje od bilo kog nivoa do korijena. Korijen stabla je bilo koja tačka u mreži sa poznatim veličinama.

Treba uočiti da se u opštem slučaju ovim algoritmom ne generiše stablo kojim bi se reprezentovala takva kombinacija uslovnih jednačina koje sadrže minimalan broj mjerenja. Algoritmom se ustvari generiše stablo kojim se reprezentuje 'n—1' puteva između jedne i svih ostalih tačaka u mreži, tako da svaki od tih puteva sadrži minimalan broj mjerenja.

Može se lako pokazati da je kompleksnost prezentiranog algoritma, zbog struktura podataka koje se koriste, reda $O(m^2)$; m — broj tačaka u mreži. Ukoliko se ovaj algoritam želi iskoristiti za generisanje pomenute kombinacije uslovnih jednačina sa minimalnim brojem mjerenja, moguće je generisati 'n' ovakvih stabala, tako da u svakom od njih korijen bude jedna poznata tačka. Njihovim pretraživanjem može se odrediti takva kombinacija uslovnih jednačina, koja će zaista sadržati minimalan broj mjerenja. Treba međutim napomenuti, da je takav algoritam reda kompleksnosti $O(nm^2)$, što neminovno dovodi i do povećanja potrebnog procesorskog vremena.

U svakom slučaju, korištenjem prikazanog algoritma i odgovarajućih struktura podataka, moguće je riješiti jedan od problema kod razvoja složenijeg i savremenijeg softvera za izravnanje geodetskih mreža, kao što je generisanje uslovnih jednačina između tačaka sa poznatim veličinama.

LITERATURA:

[1] Aho, V. A., Hopcroft, E. J., Ullman, D. J.: The Design and Analysis of Computer Algorithms, Addison-Wesley Publishing Company, Reading, Massachusetts 1974.
[2] Alagić, S.: Relational Database Technology, Springer-Verlag, New York 1986.
[3] Dijkstra, W. E.: A note on two problems in connection with graphs, Numerische Mathematik 1, 269—271.
[4] Galić, Z.: Jedan pristup upravljanju podacima prostorne prirode korištenjem relacione tehnologije baza podataka, Magistarski rad, Elektrotehnički fakultet, Beograd 1987.
[5] Horowitz, E., Sahni, S.: Fundamentals of Data Structures, Computer Science Press, Washington 1984.
[6] Prather, E. R.: Discrete Mathematical Structures for Computer Sciences, Houghton Mifflin Company, Boston 1976.
[7] Tenenbaum, M. A., Augustein J. M.: Data Structures Using Pascal, Prentice-Hall, Inc. Englewood Cliffs, New Jersey 1981.

REZIME

U radu je prikazan algoritam koji je, u širem smislu, primjenljiv za razvoj savremenijeg i za korištenje lakšeg i prihvatljivijeg softvera u oblasti uslovnog izravnanja neslobodnih geodetskih mreža. Algoritmom se generiše jedna diskretna matematska struktura (stablo), a njenim pretraživanjem se jednostavno generišu uslovne jednačine, odnosno koeficijenti jednačina grešaka i slobodni članovi.

ABSTRACT

This article presents an algorithm which can be useful to design of advanced and user-friendly oriented software in the geodetic environment. The algorithm generates a discrete mathematical structure (tree), and by traversing the tree it is possible to generate the conditional equations between given points, i.e. equations of errors for adjustment of not free geodetic networks.

Primljeno: 1986-09-08