Neki slučajevi problema mrežnog protoka

Strčić, Doris

Master's thesis / Diplomski rad

2021

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Department of Mathematics / Sveučilište Josipa Jurja Strossmayera u Osijeku, Odjel za matematiku**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:126:842124

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2025-03-31**

Repository / Repozitorij:

Repository of School of Applied Mathematics and Informatics

Sveučilište J.J. Strossmayera u Osijeku
Odjel za matematiku

Doris Strčić

Neki slučajevi problema mrežnog protoka

Diplomski rad

Osijek, 2021.

Sveučilište J.J. Strossmayera u Osijeku
Odjel za matematiku

Doris Strčić

Neki slučajevi problema mrežnog protoka

Diplomski rad

Mentor: izv. prof. dr. sc Tomislav Marošević

Osijek, 2021.

Sadržaj

1 Uvod

Problemi mrežnog protoka najčešće su rješavani problemi linearnog programiranja. Mrežni protok je vrlo važan jer se pomoću njega mogu izraziti različiti problemi. Mreže se koriste za modeliranje prometa u cestovnom sustavu, struje u električnom krugu, financijske mreže i mnogih drugih problema. Problemi mrežnog protoka uključuju kao posebne slučajeve problem transporta, problem najvećeg protoka i problem najkraćeg puta. Poseban je slučaj linearnog programiranja i bilo koji algoritam linearnog programiranja može se izravno primijeniti.
Diplomski rad podijeljen je na četiri dijela. Nakon kratkog uvoda, u drugom poglavlju navode se neki osnovni pojmovi iz teorije grafova, koji su potrebni kod problema mrežnog protoka. Zatim, u trećem dijelu razmatra se općenito problem mrežnog protoka. U zadnjem dijelu bavit ćemo se nekim specijalnim slučajevima problema mrežnog protoka, te algoritmima koji se koriste pri rješavanju istih.

2 Teorija grafova

U ovom dijelu započet ćemo uvođenjem nekih osnovnih pojmova koji će se koristiti u ovom radu. Promotrimo kratki uvod u grafove koji nam pružaju jezik za proučavanje problema protoka mreže i formulaciju tih problema. Problemi mrežnog toka definirani su na grafovima.

Definicija 2.1. *Neusmjereni graf $G = (N, E)$ sastoji se od skupa vrhova N i skupa E (neusmjerenih) lukova ili bridova, gdje je brid e neuređeni par različitih vrhova, odnosno podskup $\{i, j\}$ od N. Također, uočimo kako je neusmjereni luk $\{i, j\}$ isti objekt kao i neusmjereni luk $\{j, i\}$. Luk $\{i, j\}$ je incidentan s vrhovima i* i j, te se ti vrhovi nazivaju krajnjim točkama luka.

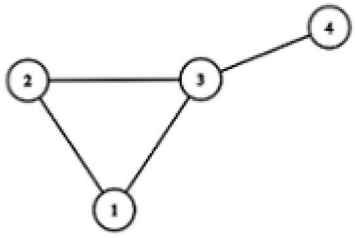
Definicija 2.2. *Stupanj vrha u neusmjerenom grafu je broj lukova koji su incidentni sa tim vrhom. Stupanj neusmjerenog grafa definiran je kao maksimum stupnjeva njegovih vrhova.*

Definicija 2.3. *Šetnja od vrha i_1 do vrha i_t u neusmjerenom grafu definira se kao konačan niz vrhova $i_1, i_2, ..., i_t$ takvih da $\{i_k, i_{k+1}\} \in E, k = 1, 2, ..., t-1$.*

Staza je šetnja u kojoj nema ponovljenih bridova. Ako su na stazi i vrhovi međusobno različiti, onda se ta šetnja naziva put.
Ciklus se definira kao šetnja $i_1, i_2..., i_t$, tako da su vrhovi $i_1, ..., i_{t-1}$ različiti i $i_t = i_1$.

Definicija 2.4. *Za neusmjereni graf kažemo da je povezan ako za bilo koja dva različita vrha i, j iz N postoji staza od i do j.*

Primjer 2.1. *Promotrimo neusmjereni graf koji je prikazan na Slici 1. Primjer jedne šetnje u ovom grafu sastoji se od vrhova 1,2,3,4. To je ujedno i put jer su u toj šetnji svi vrhovi različiti. Primjer ciklusa je niz vrhova 1,2,3,1.*



Slika 1: Neusmjereni graf

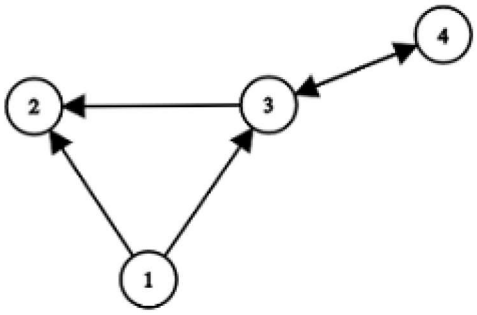
Definicija 2.5. *Usmjereni graf $G = (N, A)$ sastoji se od skupa vrhova N i skupa A (usmjerenih) lukova, gdje je usmjereni luk uređeni par (i, j) različitih vrhova.*

Prethodna definicija dopušta da skup (i, j) kao i skup (j, i) budu elementi skupa A. Za bilo koji luk (i, j) kažemo da je vrh i početni vrh, a vrh j završni vrh tog luka. Možemo definirati dva skupa, $I(i)$ i $O(i)$ na sljedeći način:

$$I(i) = \{j \in N | (j, i) \in A\}$$
$$O(i) = \{j \in N | (i, j) \in A\}$$

gdje je skup $I(i)$ skup ulaznih vrhova lukova koji ulaze u vrh i, a skup $O(i)$ je skup izlaznih vrhova lukova koji izlaze iz vrha i. Šetnja, staza i ciklus se definiraju analogno kao i u neusmjerenom grafu. Za razliku od definicije ciklusa u neusmjerenom grafu, ovdje dopuštamo da se ciklus može sastojati samo od 2 različita vrha.

Primjer 2.2. *Usmjereni graf (prikazan na Slici 2.)*
Primjerice, jedna usmjerena šetnja u ovom grafu sastoji se od niza 1,(1,3),3,(3,4),4. To je ujedno i usmjereni put jer su u toj šetnji svi vrhovi različiti. Usmjereni ciklus sastoji se od niza 3,(3,4),4,(4,3),3.



Slika 2: Usmjereni graf

Također ćemo navesti i definiciju stabla.

Definicija 2.6. *Neusmjereni graf $G = (N, E)$ naziva se stablo ako je povezan i nema ciklusa. Specijalno, ako je vrh stabla stupnja 1, onda se taj vrh naziva list.*

U sljedećem teoremu navesti ćemo neka važna svojstva stabla.

Teorem 2.1. *a) Svako stablo koje ima više od jednog vrha ima barem jedan list.*

b) Neusmjereni graf je stablo ako i samo ako je povezan i ima $|N| - 1$ lukova.

c) Za bilo koja dva različita vrha i i j u stablu postoji jedinstvena staza od i do j.

d) Ako započnemo sa stablom i dodamo novi luk, rezultirajući graf sadrži točno jedan ciklus.

Dokaz:

a) Razmotrimo stablo koje ima više od jednog vrha i pretpostavimo da nema listova. Tada svaki vrh ima stupanj veći od 1. (Da je stupanj vrha 1, taj vrh bi bio list, a da je stupanj vrha 0, graf bi bio nepovezan.) Ako krenemo od nekog vrha i luka kroz koji ulazimo u drugi vrh, možemo pronaći drugi luk kroz koji izlazimo iz tog vrha. Ponavljanjem takvog

postupka, na kraju moramo posjetiti isti vrh dva puta, što implicira da postoji ciklus. To je kontradikcija s definicijom stabla.

b) $\Rightarrow$ Prvo dokazujemo da svako stablo ima $|N| - 1$ lukova. Ako stablo ima samo jedan vrh, dokaz je trivijalan. Razmotrimo sada stablo koje ima više od jednog vrha. Takvo stablo mora imati barem jedan list, zbog tvrdnje iz a).Izbrišemo taj list zajedno sa lukom koji je incidentan sa tim vrhom. Rezultirajući graf je ponovno stablo, zbog toga što brisanjem lista ne može doći do pojavljivanja ciklusa ili uzrokovati da graf bude nepovezan. Ovaj proces se može ponavljati $|N| - 1$ puta, sve dok ne ostane samo jedan vrh i niti jedan luk. S obzirom da je u svakom stadiju bilo samo jedno brisanje luka, zaključujemo kako početno stablo ima $|N| - 1$ lukova.
$\Leftarrow$ Kako bismo dokazali obrat, razmotrimo povezani graf sa $|N| - 1$ lukova. Ako graf sadrži ciklus, možemo izbrisati jedan od lukova u ciklusu, te povezanost i dalje vrijedi. Ponavljamo ovaj proces sve dok ne dobijemo povezani graf bez ciklusa koji je stablo. Već smo dokazali da stablo sa $|N|$ vrhova mora imati $|N| - 1$ lukova, a to pokazuje da krajnje stablo ima lukova koliko i početni graf. Slijedi da niti jedan luk nije izbrisan, te da je početni graf stablo.

c) Pretpostavimo da postoje dvije različite staze koje spajaju iste vrhove i i j. Spajanjem ove dvije staze i brisanjem lukova koji su zajednički za oba vrha, ostaje jedan ili više ciklusa, što je u kontradikciji sa definicijom stabla.

d) Promatramo stablo te dodajemo neusmjereni luk $\{i, j\}$. Koristeći dio b), postojeći graf mora sadržavati $|N|$ lukova. Dakle, taj graf ne može biti stablo, te on mora imati ciklus. Svaki ciklus koji je nastao ovakvim dodavanjem lukova sastoji se od luka $\{i, j\}$, te staze od i do j. S obzirom da postoji jedinstvena staza od i do j, koristeći dio c) zaključujemo da je nastao jedinstveni ciklus.□

Definicija 2.7. *Za dani povezani neusmjereni graf $G = (N, E)$, neka je $E_1 \subset E$ takav da je $T = (N, E_1)$ stablo. Takvo stablo T naziva se razapinjuće stablo.*

3 Problem mrežnog protoka

Mreža je usmjereni graf $G = (N, A)$ zajedno s nekim dodatnim numeričkim podacima, kao što su:

- b_i – vanjska zaliha za svaki vrh $i \in N$
- u_{ij} – nenegativni brojevi koji predstavljaju kapacitet svakog luka $(i, j) \in A$
- c_{ij} – vrijednosti koje predstavljaju cijenu po jedinici protoka duž luka (i, j)
- f_{ij} – količina protoka kroz luk (i, j).

Mrežu vizualiziramo tako da zamišljamo neki materijal koji protječe svakim lukom. Zaliha b_i se može interpretirati kao količina protoka koja ulazi u mrežu u vrhu i. Za vrh i kažemo da je izvor ako je $b_i > 0$, odnosno vrh i je ponor ako je $b_i < 0$. Sada navodimo uvjete koji se postavljaju na varijable protoka $f_{ij}, (i, j) \in A$:

$$b_i + \sum_{j \in I(i)} f_{ji} = \sum_{j \in O(i)} f_{ij}, \qquad \forall i \in N \tag{3.1}$$

$$0 \leq f_{ij} \leq u_{ij}, \qquad \forall (i, j) \in A. \tag{3.2}$$

Prva jednadžba je zakon sačuvanja protoka. Ona znači da količina protoka u vrh i mora biti jednaka ukupnom protoku iz tog istog vrha. Druga jednadžba označava da protok kroz luk mora biti nenegativan, te da ne smije premašiti kapacitet luka. Iz jednadžbe 3.1, sumiranjem obje strane po svim $i \in N$, dobivamo

$$\sum_{i \in N} b_i = 0$$

što znači da ukupan protok iz okoline u mrežu mora biti jednak ukupnom protoku iz mreže u okolinu. Opći problem minimalne cijene problema mrežnog protoka odnosi se na minimizaciju linearne funkcije cijene koja ima sljedeći oblik:

$$\sum_{(i,j) \in A} c_{ij} f_{ij}.$$

Ukoliko vrijedi da je $u_{ij} = \infty, \forall (i, j) \in A$, kažemo da je problem nekapacitiran, tj. bez ograničenja kapaciteta. U suprotnom kažemo da je problem kapacitiran.

3.1 Problem mrežnog protoka u matričnom obliku

U ovom dijelu vidjet ćemo kako problem mrežnog protoka možemo predstaviti i u matričnom obliku.
Neka je $N = \{1, \ldots, n\}$, te m broj lukova. Također, s $\boldsymbol{f}$ označimo vektor protoka s komponentama f_{ij}. Uvodimo matricu incidencije vrh-luk $\boldsymbol{A}$ dimenzije $n \times m$, gdje n predstavlja broj vrhova i m broj lukova. Elemente matrice $\boldsymbol{A}$ označavamo sa a_{ik}, te ima sljedeći oblik:

$$a_{ik} = \begin{cases} 1 & \text{ako je } i \text{ početni vrh k-tog luka} \\ -1 & \text{ako je } i \text{ završni luk k-tog luka} \\ 0 & \text{inače} \end{cases}$$

Primjer 3.1. *Matrica incidencije*
Koristeći graf iz Primjera 2.2., vidimo kako u tom grafu imamo lukove (1,2), (1,3),(3,2),(3,4),(4,3). Matrica $\boldsymbol{A}$ *je sljedećeg oblika*

$$\boldsymbol{A} = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ -1 & 0 & -1 & 0 & 0 \\ 0 & -1 & 1 & 1 & -1 \\ 0 & 0 & 0 & -1 & 1 \end{pmatrix}.$$

Neka nam sada $\boldsymbol{a_i'}$ predstavlja i-ti redak matrice $\boldsymbol{A}$. Nenul vrijednosti te matrice označavaju da je neki luk incidentan sa vrhom i. Prema tome $\boldsymbol{a_i'}$ možemo zapisati na sljedeći način:

$$\boldsymbol{a_i' f} = \sum_{j \in O(i)} f_{ij} - \sum_{j \in I(i)} f_{ji}.$$

Sada možemo vidjeti kako se zakon sačuvanja protoka u vrhu i može zapisati kao

$$\boldsymbol{a_i' f} = b_i,$$

tj. u matričnom obliku

$$\boldsymbol{Af} = \boldsymbol{b},$$

gdje je $\boldsymbol{b}$=$(b_1, \ldots, b_n)$.

Kod problema mrežnog protoka u matričnom obliku cilj nam je minimizirati funkciju koštanja

$$\sum_{(i,j)\in A} c_{ij} f_{ij}$$

uz ograničenja

$$\begin{aligned} \boldsymbol{Af} &= \boldsymbol{b} \\ u_{ij} \geq f_{ij} &\geq 0, \qquad \forall (i,j) \in A. \end{aligned}$$

4 Specijalni slučajevi problema mrežnog protoka

U nastavku ćemo navesti nekoliko specijalnih slučajeva problema mrežnog protoka, te ćemo ih ukratko opisati.

4.1 Problem najkraćeg puta

Problem najkraćeg puta važan je problem s primjenama u transportnim mrežama, komunikacijskim mrežama, itd., koji se mogu postaviti kao problem mrežnog protoka. Pri rješavanju problema najkraćeg puta, praktične metode ne oslanjaju se na formulaciju mrežnog protoka. Umjesto toga koristit će se skup optimalnih uvjeta, poznatijih kao Bellmanova jednadžba koju ćemo u nastavku i opisati. U ovom dijelu cilj nam je odrediti put koji spaja izvor i ponor, tako da zbroj troškova na putu bude minimalan.
Polazimo od usmjerenog grafa $G = (N, A)$ koji ima n vrhova i m lukova. Za svaki luk $(i, j) \in A$, c_{ij} predstavlja trošak ili duljinu. Pri tome se duljina nekog puta, šetnje ili ciklusa definira kao suma duljina lukova. Pretpostavit ćemo kako je n završni vrh, te da postoji barem jedan put iz svakog vrha $i \neq n$ do vrha n. Također, bez smanjenja općenitosti pretpostavljamo kako ne postoje lukovi koji izlaze iz vrha n. Graf G promatramo kao mrežu čiji su kapaciteti lukova beskonačni. Pretpostavimo da svaki od preostalih $n-1$ vrhova je izvor s jednom jedinicom ulaza (zaliha), te da je vrh n jedini ponor s $n-1$ jedinica izlaza (zahtjeva). Ako postavimo problem minimiziranja $\sum_{(i,j)\in A} c_{ij} f_{ij}$ po svim dopustivim vektorima protoka, možemo zaključiti kako za svaki vrh i različit od vrha n, jedna jedinica protoka treba biti poslana od vrha i do vrha n, tako da bude najmanji trošak. Sve dok ne postoje ciklusi

negativne duljine, to treba činiti najkraćim putem. Ako s druge strane postoje ciklusi negativne duljine, optimalni trošak u problemu mrežnog protoka jest $-\infty$, jer bismo mogli progurati proizvoljno veliku količinu protoka oko takvog ciklusa. O tome govori sljedeći teorem.

Teorem 4.1. *Promotrimo problem najkraćeg puta u usmjerenom grafu sa n vrhova i pridruženi problem mrežnog protoka. Pretpostavimo da postoji put do vrha n iz svakog drugog vrha i da vrh n nema izlazne lukove.*

a) Ako postoji ciklus negativne duljine, optimalni trošak u problemu mrežnog protoka je $-\infty$.

b) Pretpostavimo da svi ciklusi imaju nenegativnu duljinu. Ako je dopustivo rješenje stabla optimalno, tada je odgovarajuće stablo stablo najkraćeg puta.

c) Pretpostavimo da svi ciklusi imaju nenegativnu duljinu. Ako fiksiramo dualnu varijablu p_n na 0, dualni problem ima jedinstveno rješenje $\mathbf{p}^*$, te je p_i^* najkraći put od vrha i.

Dokaz:

a) Trivijalno

b) Prvo primijetimo kako (po pretpostavci) u dopustivom rješenju stabla svi lukovi u tom stablu moraju biti orijentirani od listova prema korijenu. Razlog tome je zato što ukoliko je luk (i, j) u stablu orijentiran u suprotnom smjeru, protok kroz taj luk je negativan što je u kontradikciji s dopustivošću. Ako za neki vrh i postoji put od i do n čija je duljina manja od duljine puta u stablu, dopustivo rješenje stabla nije optimalno, jer bi se protok mogao preusmjeriti na taj put. Stoga nam optimalno dopustivo rješenje stabla pruža stablo s najkraćim putovima (stazama).

c) Neka je dualna varijabla $p_n^* = 0$ te neka je $(p_1^*, \ldots, p_n^*)$ vektor dualnih varijabli povezanih s optimalnim dopustivim rješenjem stabla. Za svaki luk na stablu vrijedi $p_i^* = c_{ij} + p_j^*$. Budući su svi lukovi orijentirani prema korijenu, možemo dodati jednakosti $p_i^* = c_{ij} + p_j^*$ duž puta sadržanog u stablu, te zaključimo kako je p_i^* duljina puta od vrha i do vrha n. Uočimo kako je to najkraći put, budući se radi o optimalnom dopustivom stablu. Napokon primjećujemo da je svako dopustivo

rješenje stabla nedegenerirano. To je zato što bilo koji luk (i, j) u stablu mora nositi zalihu (ulaz) na vrhu i. Slijedi da dualni problem ima jedinstveno rješenje.□

4.1.1 Bellmanova jednadžba

Neka su $b_1 = \ldots = b_{n-1} = 1$. Pod pretpostavkom da je $p_n = 0$, dualni problem je oblika

$$max \sum_{i=1}^{n-1} p_i$$

uz uvjet

$$p_i \leq c_{ij} + p_j, \quad \forall (i, j) \in A.$$

Optimalno rješenje $\mathbf{p}^*$ dualnog problema, koje je isto kao i vektor najkraćih duljina puta, zadovoljava

$$p_i^* = min_{k \in O(i)} \{c_{ik} + p_k^*\}, \quad i = 1, \ldots, n-1$$

gdje je $p_n^* = 0$. To je sustav od $n-1$ nelinearnih jednadžbi sa $n-1$ nepoznanica, poznatijih kao Bellmanova jednadžba.
Pretpostavimo da tražimo putove koji kreću iz vrha i do vrha n, ali uz ograničenje da put započinje s lukom (i, k). Tada je najbolje što možemo učiniti, pronaći najkraći put od vrha k do vrha n ukupne duljine $c_{ik} + p_k^*$. Cilj ove jednadžbe je postići optimalnost. Ako postoji najkraći put od i do n koji prolazi kroz vrh k, tada je i dio puta od k do n najkraći put.
Postoje mnogi načini za izračun Bellmanove jednadžbe. To se može učiniti direktnim rješavanjem jednadžbe, ali ona daje nekoliko rješenja od kojih samo jedno daje duljinu najkraćeg puta. Iz tog razloga za rješavanje ove jednadžbe koristit ćemo Bellman-Ford algoritam.

4.1.2 Bellman-Ford algoritam

Bellman-Ford algoritam je algoritam koji se koristi pri rješavanju problema najkraćeg puta, od početnog vrha do svih ostalih vrhova u težinskom grafu. U usporedbi s Dijkstrinim algoritmom, težinske vrijednosti lukova mogu imati negativne vrijednosti. Ako graf sadrži negativni ciklus koji je dostupan iz izvora, tada algoritam ne može riješiti problem najkraćeg puta.
Pretpostavimo da vrh n nema izlazne lukove. Neka p_t predstavlja duljinu najkraćeg puta od vrha i do vrha n koja koristi najviše t lukova. Ukoliko

takva šetnja ne postoji, onda je $p_i(t) = \infty$. Neka je $p_n(t) = 0, \forall t$ i $p_i(0) = \infty, \forall i \neq n$. Primijetimo kako je $p_i(t+1) \leq p_i(t), \forall i, t$. Razlog tome je što povećavanjem t, postoji sve više putova koji se mogu izabrati. Najkraći put od vrha i do vrha n koji se sastoji od najviše $t+1$ lukova, sadrži početni luk (i, k) i šetnju od vrha k do vrha n koja se sastoji od najviše t lukova. Šetnja mora biti što kraća, te duljina te šetnje iznosi $p_k(t)$. Budući bi vrh k trebao biti odabran na što profitabilniji način imamo

$$p_i(t) = min_{k \in O(i)}\{c_{ik} + p_k(t)\}, i = 1, \ldots, n-1$$

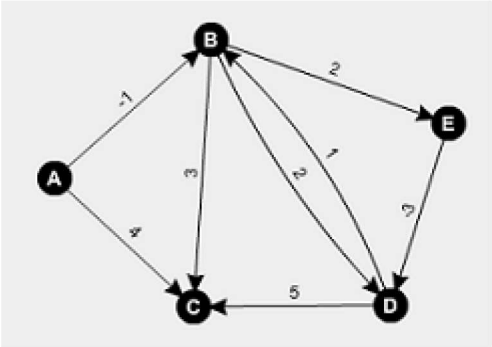
te je ovom jednadžbom definiran Bellman-Ford algoritam.
Sada ćemo navesti u kojim slučajevima završava navedeni algoritam:

1. Ako pretpostavimo da ne postoje ciklusi negativne duljine. Tada postoji najkraća šetnja, koja je također i najkraći put te ima najviše $n-1$ lukova. Vrijedi da je $p_i(n-1) = p_i^*$. Ukoliko dopustimo šetnju koja ima n ili više lukova, ukupna duljina se ne može smanjiti, te vrijedi $p_i(n) = p_i(n-1)$ za sve vrhove.

2. Ako pretpostavimo da postoje ciklusi negativne duljine. Također pretpostavimo i da vrijedi $\mathbf{p}(n) = \mathbf{p}(n-1)$. To povlači da je $\mathbf{p}(t) = \mathbf{p}(n), \forall t \geq n$ te je duljina svake šetnje ograničena odozdo. Uz prisutnost ciklusa negativne duljine, postoje šetnje čija duljina teži u $-\infty$. To je kontradikcija, što dokazuje da je $\mathbf{p}(n) \neq \mathbf{p}(n-1)$.

Usporedbom ova dva slučaja, vidimo kako nije potrebno više od n iteracija. Ako je $\mathbf{p}(n) = \mathbf{p}(n-1)$, tada $\mathbf{p}(n)$ vektor duljine najkraćeg puta. Ako je $\mathbf{p}(n) \neq \mathbf{p}(n-1)$ tada postoji ciklus negativne duljine.

Primjer 4.1. *U ovom primjeru tražimo najkraću udaljenost od vrha A do svih ostalih vrhova u grafu.*



U početku postavimo vrijednost u A na 0, dok sve preostale vrijednosti postavimo na ∞. U prvom koraku tražimo putove sa jednim bridom, zatim u idućem koraku putove sa dva brida, itd. U svakom sljedećem koraku prolazi se kroz svaki čvor i ažurira tablica najkraćeg puta. S obzirom da naš graf ima 5 vrhova, algoritam završava nakon 4 koraka. Rješenje ovog primjera možemo vidjeti u sljedećoj tablici:

A	B	C	D	E
0	∞	∞	∞	∞
0	*-1*	*4*	∞	∞
0	*-1*	*2*	*1*	*1*
0	*-1*	*2*	*-2*	*1*
0	*-1*	*2*	*-2*	*1*

Kako su p(4) = p(3), zaključujemo da je p(4) vektor duljine najkraćeg puta.

4.1.3 Algoritam ispravljanja oznaka

Metoda ispravljanja oznaka jedan je od načina traženja najkraćeg puta. Vrlo je sličan Bellman-Ford algoritmu. Ideja koja stoji iza ove metode je održati na svakom vrhu j oznaku p_j, koja je jednaka duljini do sada otkrivene najkraće šetnje od vrha j do vrha n. S obzirom na postojeću šetnju od j do n duljine p_j, postoji šetnja od i do n duljine $c_{ij} + p_j$. Svaki put kada se p_j

premjesti prema dolje, također revidiramo prema dolje oznake svih vrhova i koji imaju izlazni luk prema vrhu j. Algoritam održava listu S svih vrhova čije su oznake revidirane prema dolje, te revizija još nije proširena na njihove prethodnike.
Sada ćemo taj algoritam zapisati formalno:

Algoritam s ispravljanjem oznaka:
Algoritam je inicijaliziran sa $S = \{n\}, p_n = 0, p_i = \infty, \forall i \neq n$.

1. Ukloni vrh j iz S.

2. Za svaki vrh $i \neq n$ takav da je (i, j) luk, učini sljedeće:
   Neka je $p_i = min\{p_i, c_{ij} + p_j\}$. Ako je nova vrijednost od p_i manja, dodaj vrh i u S.

3. Ako je skup S prazan, algoritam završava. Inače se vrati na korak 1.

Oznaka vrha uvijek je jednaka duljini neke šetnje do vrha n. U slučaju kada put još nije otkriven, oznaka je onda beskonačno.
Sljedeći teorem govori o uvjetima zaustavljanja prethodno navedenog algoritma.

Teorem 4.2. *Pretpostavimo da postoji put od svakog vrha do vrha n, te da svi ciklusi imaju nenegativnu duljinu. Tada algoritam ispravljanja oznaka konačno završava s oznakom p_i svakog vrha koja je jednaka duljini najkraćeg puta p_i^*.*

Dokaz: Pretpostavimo da je $i_1, i_2, \ldots, i_t = n$ najkraći put od nekog vrha i_1 do vrha n. Iz definicije algoritma, vrijedi $p_n = 0 = p_n^*$ u svakom trenutku. U prvom koraku algoritma vrijedi $S = \{n\}$, ispitani su prethodnici od n, te stavljamo $p_{i_{t-1}} = c_{i_{t-1}n}$ što je jednako $p_{i_{t-1}}^*$. Ovo vrijedi zato što je posljednji luk u najkraćem putu sam po sebi već najkraći put. Sada razmotrimo srednji vrh i_k u putu, te pretpostavimo kako je konačna oznaka p_{i_k} jednaka $p_{i_k}^*$. Kako je p_{i_k} u početku bio beskonačan, njegova oznaka se promijenila barem jednom. Posljednji put kada se p_{i_k} promijenio i postavljen je na $p_{i_k}^*$, vrh i_k je ušao u skup S. Kada u nekom sljedećem koraku i_k izađe iz skupa S, postavljamo $p_{i_{k-1}}$ na $min\{p_{i_{k-1}}, c_{i_{k-1}i_k} + p_{i_k}^*\}$. Ta vrijednost je manja ili jednaka od $c_{i_{k-1}i_k} + p_{i_k}^* = p_{i_{k-1}}^*$. Sa druge strane, $p_{i_{k-1}}$ predstavlja duljinu neke šetnje, te ona ne može biti manja od $p_{i_k}^*$. Time je dokazano da je $p_{i_k} = p_{i_k}^*$ za svaki k. □

4.1.4 Dijkstrin algoritam

Dijkstrin algoritam je algoritam koji se koristi za rješavanje problema najkraćeg puta u grafu čije vrijednosti duljine lukova imaju nenegativne vrijednosti. Ideja ovog algoritma je identificirati vrhove po redoslijedu odgovarajućih najkraćih duljina puta, počevši od najkraće.
S c_{ij} označavat ćemo cijenu protoka duž usmjerenog luka (i, j), te c_{ij} je definiran za svaki par (i, j) različitih vrhova, a može i poprimiti vrijednost ∞ za neke parove.
U sljedećem teoremu pokazat ćemo kako pronalazimo najkraći put za neki vrh l.

Teorem 4.3. *Pretpostavimo da je* $c_{ij} \geq 0, \forall i, j$. *Neka je* $l \neq n$ *takav da vrijedi*

$$c_{ln} = min_{i \neq n} c_{in}.$$

Tada, $p_l^* = c_{ln}$ *i* $p_l^* \leq p_k^*, \forall k \neq n$.

Dokaz: Bilo koji put do vrha n ima luk (i, n) čija duljina c_{in} iznosi barem c_{ln}. Prema tome, $p_k^* \geq c_{ln}, \forall k \neq n$. Za vrh l također vrijedi $p_l^* \leq c_{ln}$. Zaključujemo kako je $p_l^* = c_{ln} \leq p_k^*, \forall k \neq n$. □

Dijkstrin algoritam:

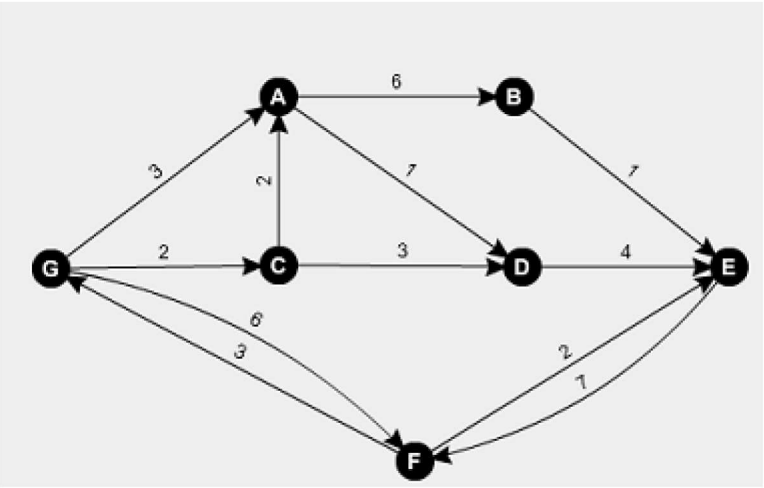
1. Pronađi vrh $l \neq n$ takav da je $c_{ln} \leq c_{in}$ za svaki $i \neq n$. Postavi $p_l^* = c_{ln}$
2. Za svaki vrh $i \neq l, n$, postavi

$$c_{in} = min\{c_{in}, c_{il} + c_{ln}\}$$

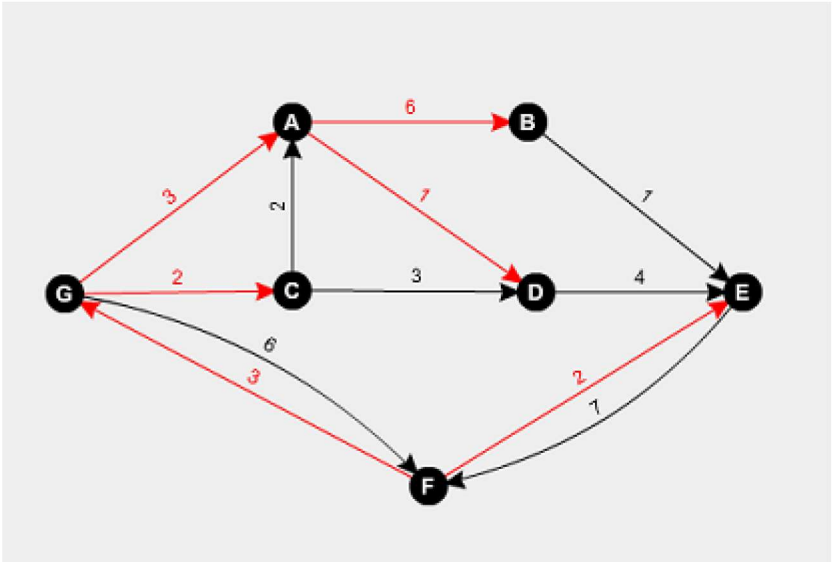
3. Ukloni vrh l iz grafa te ponovi iste korake.

Primjenu ovog algoritma pokažimo na sljedećem primjeru.

Primjer 4.2. *U ovom primjeru tražimo najkraći put od danog vrha do svih ostalih vrhova. Za početni vrh uzet ćemo vrh F.*



Na početku postavimo vrijednost u F na 0, a sve preostale vrijednosti postavimo na ∞. Prvo gledamo koji su susjedni vrhovi vrha F. Vidimo kako su to vrhovi G i E. Udaljenost od F do G iznosi 3, a udaljenost od F do E iznosi 2. Biramo $F \to E$ jer ima manju duljinu. Zatim kao sljedeću mogućnost uzimamo $F \to G$ jer je to i jedina mogućnost.
Sada kao sljedeće mogućnosti imamo $G \to A$, tj. put $F \to G \to A$ duljine 6 i $G \to C$, tj. put $F \to G \to C$ duljine 5. Odabiremo $G \to C$.
Zatim imamo mogućnosti $G \to A$ duljine 6, $C \to A$ duljine 7 i $C \to D$ duljine 8. Odabiremo $G \to A$.
Kao sljedeće mogućnosti imamo $A \to B$ duljine 12, $C \to A$ duljine 7, $A \to D$ duljine 7, te $C \to D$ duljine 8. Odabiremo $A \to D$.
Na kraju nam još jedino preostaje $A \to B$ čija duljina iznosi 12.
Na idućoj slici možemo vidjeti najkraće putove od vrha F do preostalih vrhova.



4.2 Problem najvećeg protoka

Neka je $G = (N, A)$ usmjereni graf, te neka $u_{ij} \in [0, \infty >$ predstavlja kapacitet usmjerenog brida za svaki brid $(i, j) \in A$. Neka su s i t izvor i ponor grafa. Cilj ovog problema je pronaći najveći mogući protok koji može biti poslan kroz mrežu, od s do t. Problem najvećeg protoka može se modelirati kao problem mrežnog protoka. Vrhovi predstavljaju izvor, ponor, spojna mjesta, dok lukovi (bridovi) predstavljaju kanale kroz koje se prevozi materijal.
Sljedeća definicija govori o putovima kroz koje se može progurati (staviti) dodatni protok.

Definicija 4.1. *Neka je f vektor dopustivog protoka. Povećavajući put je put od s do t takav da je $f_{ij} \leq u_{ij}$ za sve lukove od naprijed, te $f_{ij} \geq 0$ za sve lukove od nazad na putu (stazi).*

Ako s P označimo povećavajući put, tada s $\delta(P)$ označavamo količinu protoka koji se može dodati (staviti, pogurati) kroz P:

$$\delta(P) = min\{min_{(i,j)\in F}(u_{ij} - f_{ij}), min_{(i,j)\in B} f_{ij}\},$$

gdje F i B predstavljaju skupove izlaznih i ulaznih lukova (tj. lukova od naprijed, te lukova od nazad) u P. Ako svi lukovi na putu imaju neograničen kapacitet, tada nema ograničenja iznosa protoka i $\delta(P) = \infty$.
Sada uvodimo algoritam kojeg ćemo koristiti za problem najvećeg protoka.

Radi se o Ford-Fulkerson algoritmu kojeg su 1956. godine otkrili Ford i Fulkerson. Ideja ovog algoritma je da sve dok postoji put od izvora do ponora, s raspoloživim kapacitetom na svim rubovima staze, šaljemo protok duž jedne staze. Zatim pronađemo drugi put, i tako nastavljamo dalje. Algoritam završava onda kada više ne možemo pronaći put koji poboljšava protok.

Ford-Fulkerson algoritam:

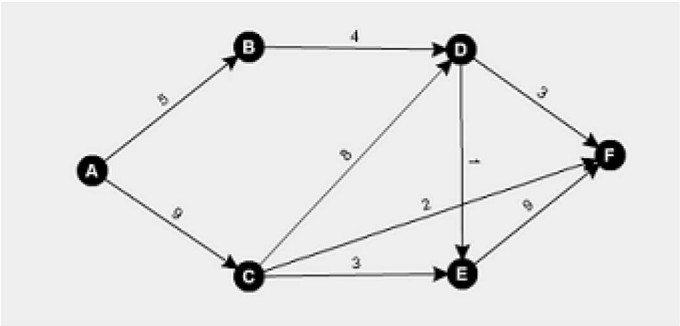
1. Započni s dopustivim protokom f.
2. Pronađi povećavajući put.
3. Ako ne postoji povećavajući put, algoritam završava.
4. Ako postoji povećavajući put P, onda:
   a) Ako je $\delta(P) < \infty$, pošalji $\delta(P)$ jedinica protoka kroz P, zatim se vrati na korak 2.
   b) Ako je $\delta(P) = \infty$, algoritam završava.

U sljedećem teoremu navest ćemo svojstva koja uvjetuju završavanje prethodnog algoritma.

Teorem 4.4. *Pretpostavimo da su kapaciteti lukova u_{ij} cjelobrojne ili beskonačne vrijednosti, te da je Ford-Fulkerson algoritam inicijaliziran cjelobrojnim vektorom protoka. Tada, varijable protoka lukova ostaju cjelobrojne kroz cijeli algoritam i ako je optimalna vrijednost konačna, algoritam završava nakon konačno mnogo koraka.*

Dokaz: Ako imamo cjelobrojni dopustivi protok i ako su svi kapaciteti lukova cjelobrojni ili beskonačni, onda je i $\delta(P)$ cjelobrojan ili beskonačan. Svaka iteracija algoritma povećava vrijednost protoka za barem 1. Iz tog razloga se vrijednost protoka povećava do beskonačnosti ili algoritam mora završiti. □

Primjer 4.3. *U ovom primjeru tražimo maksimalan protok kroz mrežu primjenom Ford- Fulkerson algoritma. Želimo odrediti maksimalan protok između vrhova A i F.*



Tražimo stazu od A do F kojom možemo poslati pozitivan protok. Gledamo stazu $A \to C \to E \to F$. Maksimalan protok koji možemo poslati duž ove staze je k=3. Sljedeća staza kojom možemo poslati pozitivan protok je $A \to C \to F$. Maksimalan protok koji šaljemo duž ove staze je k=2. Zatim imamo stazu $A \to B \to D \to F$, kojom šaljemo maksimalan protok k=3. Sljedeća staza kojom možemo poslati pozitivan protok je $A \to B \to D \to E \to F$. Maksimalan protok koji možemo poslati duž ove staze je k=1. Vidimo kako više ne postoji staza na dobivenoj mreži koja dopušta pozitivan protok. Maksimalna količina materijala koja se može poslati do vrha A do vrha F iznosi 9.

4.2.1 Povećavajući put

Traženje povećavajućeg puta u mreži jednostavno možemo odrediti pomoću metode označavanja. Pretpostavimo da je **f** dopustiv protok. Razmotrimo put od izvora s do nekog vrha k, takav da vrijedi $f_{ij} < u_{ij}$ za sve lukove prema naprijed na putu, te $f_{ij} > 0$ za sve lukove prema nazad na tom putu. Za takav put kažemo da je nezasićeni.
Za neki vrh i ćemo reći da je označen ako smo utvrdili kako postoji nezasićeni put od s do i. Sada ćemo navesti algoritam za označavanje:

Algoritam za označavanje:

1. Algortiam je inicijaliziran sa $I = \{s\}$, gdje je s jedini označeni vrh.

2. Iteracija započinje skupom I označenih, ali još neispitanih vrhova. Ako je $t \in I$ ili $I = \emptyset$, algoritam završava. Inače, izaberi vrh $i \in I$ za skeniranje, i izbriši ga iz skupa I. Treba ispitati sve lukove oblika (i, j) ili (j, i).

3. Ako je $(i,j) \in A, f_{ij} < u_{ij}$ i j nije označen, tada označi j i dodaj ga u skup I.

4. Ako $(j,i) \in A, f_{ji} > 0$, i j nije označen, tada označi j i dodaj ga u skup I.

4.3 Problem pridruživanja i aukcijski algoritam

Problem pridruživanja javlja se kao specijalni slučaj problema transporta. Iz toga razloga ćemo prvo definirati problem transporta.
Pretpostavimo da imamo m opskrbljivača i n potrošača. Neka $s_i, i = 1, \ldots, m$ predstavlja količinu jedinica robe koju i-ti opskrbljivač može isporučiti, dok je $d_j, j = 1, \ldots, n$ količina jedinica koju potražuje j-ti potrošač. Pri tome pretpostavljamo kako je $\sum_{i=1}^{m} s_i = \sum_{j=1}^{n} d_j$. Pretpostavimo kako trošak transporta robe po jedinici robe od i-tog proizvođača do j-tog potrošača iznosi c_{ij}. S f_{ij} označit ćemo količinu proizvoda koju transportiramo od i-tog proizvođača do j-tog potrošača. Naš problem transporta svodi se na to kako transportirati robu od proizvođača do potrošača po minimalnoj cijeni, tj.

$$min \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} f_{ij}$$

uz uvjet

$$\sum_{i=1}^{m} f_{ij} = d_j, \quad \forall j = 1, \ldots, n$$

$$\sum_{j=1}^{n} f_{ij} = s_i, \quad \forall i = 1, \ldots, m$$

$$f_{ij} \geq 0, \quad \forall i, j.$$

Kao što smo već i spomenuli, problem pridruživanja specijalan je slučaj problema transporta, gdje ima n proizvođača i n potrošača, te svaki proizvođač ima jedinicu robe i svaki potrošač ima jedinicu zahtjeva, pri čemu treba minimizirati linearnu funkciju troška. Pri tome je $f_{ij} = 1$ ako je i-ti proizvođač pridružen j-tom potrošaču, a $f_{ij} = 0$ u suprotnom. Dakle, problem pridruživanja je

$$min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} f_{ij}$$

uz uvjet

$$\sum_{i=1}^{n} f_{ij} = 1, \quad j = 1, \ldots, n$$

$$\sum_{j=1}^{n} f_{ij} = 1, \quad i = 1, \ldots, n$$

$$f_{ij} \geq 0, \quad \forall i, j.$$

Broj proizvođača jednak je broju potrošača. Može se pokazati da uvijek postoji optimalno rješenje u kojem je svaki f_{ij} jednak ili 1 ili 0.
Sada ćemo gledati dualni problem problema pridruživanja. Povezujemo dualnu varijablu r_i sa svakim ograničenjem $\sum_{j=1}^{n} f_{ij} = 1$, te dualnu varijablu p_j sa svakim ograničenjem $\sum_{i=1}^{n} f_{ij} = 1$. Tada dualni problem početnog problema ima sljedeći oblik

$$max \quad \sum_{i=1}^{n} r_i + \sum_{j=1}^{n} p_j$$

uz uvjet

$$r_i + p_j \leq c_{ij}, \qquad \forall i, j.$$

Iz dualnih ograničenja vidi se da jednom kada su vrijednosti $p_1, \ldots, p_n$ određene, $\sum_{i=1}^{n} r_i$ je maksimizirana ako svaki r_i postavimo na najveću vrijednost koja je dopuštena uz uvjet $r_i + p_j \leq c_{ij}$, tj.

$$r_i = min_{j=1,\ldots,n}\{c_{ij} - p_j\}. \tag{4.1}$$

To vodi do sljedećeg ekvivalentnog dualnog problema:

$$max \quad \sum_{j=1}^{n} p_j + \sum_{i=1}^{n} min_j\{c_{ij} - p_j\} \tag{4.2}$$

Sada ćemo razmotriti komplementarne uvjete ("complementary slackness conditions", tj. uvjete oslabljene komplementarnosti) za problem pridruživanja:

a) protok mora biti očuvan

b) ako je $f_{ij} > 0$, tada je $r_i + p_j = c_{ij}$.

Korištenjem jednadžbe 4.1 za eliminaciju r_i, drugi komplementarni uvjet ekvivalentan je sljedećem uvjetu

$$\text{ako je } f_{ij} > 0, \text{ onda je } p_j - c_{ij} = -r_i = max_k\{p_k - c_{ik}\}.$$

Uzmimo jednu interpretaciju problema pridruživanja, kao da postoji n osoba i n projekata, te da treba svakom pojedinom projektu pridružiti različitu osobu za minimiziranje linearne funkcije troška. Tada prethodni uvjet znači da bi svaka osoba trebala biti pridružena projektu koji je najviše profitabilan (jer razlika $p_k - c_{ik}$ predstavlja profit (zaradu) osobe i koja provodi projekt k).

4.3.1 Naivni aukcijski algoritam

Aukcijski algoritam koristi se pri rješavanju problema pridruživanja. Koristi se u poslovnom okruženju za određivanje najboljih cijena niza proizvoda koji su ponuđeni za više kupaca. Radi se o iterativnom postupku u kojem se generira niz optimalnih cijena i pridruživanja. Također, komplementarni uvjet $p_j - c_{ij} = -r_i = max_k\{p_k - c_{ik}\}$ vrijedi za sve parove (i, j) iz pridruživanja.

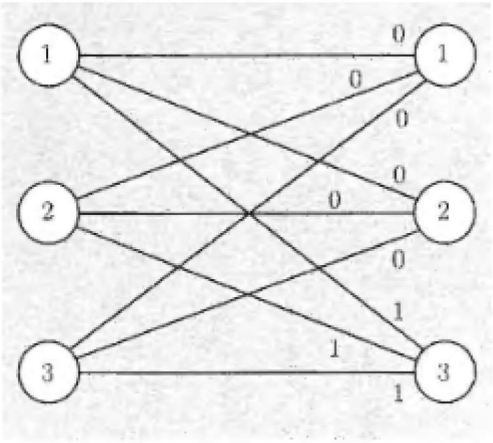
Naivni aukcijski algoritam:

1. Neka je $p_1, ..., p_n$ dani skup cijena za različite projekte, te neka su osobe djelomično pridružene objektima. Svaka osoba koja nije dodijeljena projektu traži najbolji projekt koji maksimizira profit $p_j c_{ij}$. Osoba licitira za taj projekt, pri čemu prihvaća nižu cijenu. Cijena se spušta za razliku izmedu profita najboljeg projekta i profita drugog najboljeg projekta.

2. U ovoj fazi svaki projekt dodjeljuje se ponuđaču s najmanjom ponudom (ako postoji). Nova cijena svakog projekta postavlja se na vrijednost najniže ponude. Osoba koja je bila dodijeljena nekom projektu na početku, sada postaje nedodijeljena.

Nažalost, ovaj algoritam se ne može primijeniti uvijek. To ćemo vidjeti na sljedećem primjeru.

Primjer 4.4. *Pretpostavimo da imamo tri osobe i tri projekta. Neka su troškovi za prva dva projekta jednaki nula za svaku osobu i, tj.* $c_{i1} = 0$, $c_{i2} = 0$, $\forall i = 1, 2, 3$, *a za treći projekt je* $c_{i3} = 1$, $\forall i$.



U ovom slučaju problem pridruživanja ima oblik

$$minimizirati \sum_{i=1}^{3}\sum_{j=1}^{3} c_{ij}f_{ij} = c_{13}f_{13} + c_{23}f_{23} + c_{33}f_{33},$$

uz uvjete

$$\sum_{i=1}^{3} f_{ij} = 1, \quad j = 1, 2, 3,$$

$$\sum_{j=1}^{3} f_{ij} = 1, \quad i = 1, 2, 3,$$

$$f_{ij} \geq 0, \quad \forall i, j = 1, 2, 3.$$

Neka su svi p_i *jednaki 1, osoba 1 dodijeljena je projektu 1, osoba 2 projektu 2 i osoba 3 projektu 3.*

Osoba 3 računa koliki profit imaju preostala dva projekta. Za prvi i drugi projekt profit iznosi $1 - 0 = 1$, *a za treći* $1 - 1 = 0$. *Kako su profiti za prvi i drugi projekt jednaki, osoba 3 licitira za drugi projekt, te je pridružena tom projektu. U sljedećem koraku, osoba 2 koja sada nema dodijeljen projekt,*

računa profit na isti način kao i osoba 3. Dobivene su iste vrijednosti, te se osoba 2 odlučuje licitirati za projekt 2. Cijena je opet nepromijenjena. Algoritam završava s istim vrijednostima koje su bile i na početku.

Ovo je primjer kada naivni aukcijski algoritam nije primjenjiv. Međutim, pokazat ćemo da uz neke korekcije algoritam funkcionira.

4.3.2 Aukcijski algoritam

Kao što je već i spomenuto u prethodnom naslovu, uz dodatne modifikacije naivni aukcijski algoritam funkcionira. Iz tog razloga, fiksiramo pozitivan broj ϵ.
Sada je licitacija (ponuda) za neki projekt manja za taj ϵ, od one koja bi bila da smo željeli da projekt ostane najbolji. Uočimo kako licitacija smanjuje cijenu projekta ispod razine na kojoj bi taj projekt bio profitabilniji. Iz tog razloga, osobe općenito neće biti dodijeljene najprofitabilnijem projektu.
Komplementarni uvjet u ovom slučaju definiran je na sljedeći način.

Definicija 4.2. *Razmotrimo skup cijena p_j i djelomično pridruživanje u kojem je svaka pridružena osoba $i \in S$ pridružena projektu j_i. Kažemo da uvjet ϵ-komplementarnosti vrijedi ako je*

$$p_{j_i} - c_{ij_i} \geq max_k\{p_k - c_{ik}\} - \epsilon, \quad \forall i \in S.$$

Aukcijski algoritam:

1. Iteracija započinje skupom cijena $p_1, \ldots, p_n$ za različite projekte, skupom S u kojem se nalaze pridružene osobe, te projektom j_i koji je pridružen svakoj osobi $i \in S$, pa je $f_{ij_i} = 1$ za $i \in S$. (Na početku algoritma je skup S prazan).

2. Svaka nepridružena osoba $i \notin S$ pronalazi najbolji projekt k_i tako što maksimizira profit $p_k - c_{ik}$, za svaki k. Neka $k_i^{'}$ predstavlja drugi najbolji projekt za kojeg je

   $$p_{k_i'} - c_{ik_i'} \geq p_k - c_{ik}, \quad \forall k \neq k_i$$

   Neka je $\Delta_{k_i} = (p_{k_i} - c_{ik_i}) - (p_{k_i'} - c_{ik_i'})$.
   Osoba i licitira $p_{k_i} - \Delta_{k_i} - \epsilon$ za projekt i.

3. Svaki projekt za koji postoji najmanje jedna licitacija dodjeljuje se ponuđaču s najnižom cijenom; stari nositelj projekta (ako postoji) postaje nedodijeljen. Nova cijena p_i svakog pojedinog projekta koji je primio barem jednu ponudu postavlja se na vrijednost najniže ponude.

Može se pokazati da ovaj aukcijski algoritam završava nakon konačno mnogo koraka.

Teorem 4.5. *Aukcijski algoritam završava nakon konačno mnogo koraka s dopustivim pridruživanjem.*

Dokaz: Dokaz ovog teorema zasniva se na sljedećim zapažanjima:

a) Jednom kada je dobivena licitacija (ponuda) za projekt, on je dodijeljen nekoj osobi. Jednom kada je taj projekt dodijeljen, on kasnije može biti dodijeljen nekoj drugoj osobi, ali nikada neće biti nedodijeljen. Ako su dobivene licitacije za sve projekte, svi projekti su dodijeljeni, a samim time su i sve osobe dodijeljene (pridružene).

b) Ako su sve osobe dodijeljene, više nema licitacija (ponuda) i algoritam završava.

c) Ako algoritam ne završava, onda neki projekt nikada nije dodijeljen. Takav projekt nikad nije bio licitiran, te je njegova cijena fiksirana na početnu vrijednost.

d) Ako algoritam ne završava, neki projekti zaprimaju beskonačno mnogo licitacija. Budući da svaka uzastopna ponuda snižava cijenu za najmanje ϵ, cijena takvog projekta pada na $-\infty$.

Koristeći *c*) i *d*), može se zaključiti da projekt koji nikada nije bio licitiran mora postati profitabilniji od bilo kojeg projekta za kojeg je zaprimljeno beskonačno mnogo licitacija. Sa druge strane, vrijedi i suprotna tvrdnja. A to je kontradikcija, te utvrđujemo kako će svaki projekt na kraju biti licitiran (tj. primiti ponudu). Koristeći *a*) i *b*), algoritam mora konačno završiti, kada su svim osobama dodijeljeni projekti. □

Završetkom aukcijskog teorema dobili smo pridruživanje koje zadovoljava ϵ-komplementarnost, ali se pitamo je li to pridruživanje optimalno. Odgovor na to pitanje daje veličina ϵ. Može se pokazati da ako je ϵ proizvoljno mali,

aukcijski algoritam ima optimalno rješenje.
Idući teorem nam govori da kada je $\epsilon < \frac{1}{n}$, a svi c_{ij} su cjelobrojne vrijednosti, pridruživanje koje je dobiveno završetkom aukcijskog algoritma je optimalno.

Teorem 4.6. *Ako su koeficijenti troška c_{ij} cijeli brojevi i ako vrijedi*

$$0 < \epsilon < \frac{1}{n},$$

onda aukcijski algoritam završava s optimalnim rješenjem.

Dokaz: Neka j_i predstavlja projekt pridružen osobi i u trenutku završetka algoritma. Koristeći ϵ-komplementarnost imamo

$$p_{j_i} - c_{ij_i} \geq max_j\{p_j - c_{ij}\} - \epsilon, \quad \forall i.$$

Sumiranjem nejednakosti po svim i, te preslagivanjem dobivamo

$$\sum_{i=1}^{n} c_{ij_i} \leq \sum_{i=1}^{n}(p_{j_i} - max_j\{p_j - c_{ij}\}) + n\epsilon = \sum_{i=1}^{n}(p_{j_i} + min_j\{c_{ij} - p_j\}) + n\epsilon.$$

Sada neka je z trošak optimalnog pridruživanja. Vidimo kako je suma na desnoj strani nejednakosti jednaka dualnom obliku problema

$$max \quad \sum_{j=1}^{n} p_j + \sum_{i=1}^{n} min_j\{c_{ij} - p_j\},$$

pa prema slaboj dualnosti slijedi da je ograničena odozgo sa optimalnim troškom z. To implicira da je

$$\sum_{i=1}^{n} c_{ij_i} \leq z + n\epsilon < z + 1.$$

S druge strane, prema definiciji od z vrijedi da je

$$\sum_{i=1}^{n} c_{ij_i} \geq z.$$

Kako su z i c_{ij_i} cijeli brojevi, zaključujemo da je

$$\sum_{i=1}^{n} c_{ij_i} = z,$$

te vrijedi optimalnost.□

Primjer 4.5. *U ovom primjeru primijenit ćemo aukcijski algoritam na problem kojeg smo već razmatrali u primjeru 4.4. Pretpostavimo da su osobe 1 i 2 dodijeljene redom projektima 1 i 2, te da su sve početne cijene jednake 1. Osoba 3 licitira za projekt 2, te smanjuje njegovu cijenu na* $1 - \epsilon$*. Osoba 2 postaje nedodijeljena, te izračunava dobit za različite projekte; te vrijednosti su redom jednake:* $1 - 0 = 1, (1 - \epsilon) - 0 = 1 - \epsilon, 1 - 1 = 0$.
Vidimo kako je projekt 1 najprofitabilniji. Njegove cijene treba sniziti tako da mu profit postane jednak dobiti drugog najboljeg projekta, umanjen za ϵ*. Stoga, ponuda iznosi* $1 - 2\epsilon$.
U idućoj iteraciji, osoba 1, koja je nedodijeljena, licitira za projekt 2 i smanjuje njegovu vrijednost na $1 - 3\epsilon$*. Pri svakoj iteraciji projekti 1 i 2 imaju cijene koje su unutar* ϵ *jedna od druge. Nedodijeljena osoba uvijek licitira za onaj projekt koji ima veću cijenu i snižava joj cijenu za* 2ϵ*. Nakon određenog broja ponavljanja, cijene projekata 1 i 2 postaju negativne. U tom trenutku, projekt 3 konačno postaje profitabilan, dobiva ponudu, postaje dodijeljen, te algoritam završava.*

Literatura

[1] R. K. Ahuja, T. L. Magnanti, J. B. Orlin, Network flows: Theory, Algorithms and applications, Prentice Hall, 1993.
dostupno na: http://cs.yazd.ac.ir/hasheminezhad/STSCS4R1.pdf

[2] D. P. Bertsekas, Network Optimization: Continuous and Discrete Models, Network Optimization: Continuous and Discrete Models, Athena Scientific, Belmont, 1998.

[3] D. Bertsimas, J. N. Tsitsiklis, Introduction to Linear Optimization, Athena Scientific, Belmont, Massachusetts, 1997.

[4] D. P. Williamson, Network Flow Algorithms, Cambridge University Press, Cambridge, 2019.,
dostupno na: https://www.networkflowalgs.com/book.pdf

Sažetak

U ovom diplomskom radu razmatrani su neki slučajevi problema mrežnog protoka. Na samom početku definiran je općenito problem mrežnog protoka. U glavnom dijelu rada definirani su problem najkraćeg puta, problem najvećeg protoka, te problem pridruživanja. Također, detaljno su objašnjeni i algoritmi koji se koriste pri rješavanju navedenih problema mrežnog protoka.

Ključne riječi:
usmjereni graf, mrežni protok, problem najkraćeg puta, Bellman-Ford algoritam, Dijkstrin algoritam, problem najvećeg protoka, Ford-Fulkerson algoritam, problem pridruživanja, aukcijski algoritam.

Summary

In this thesis, some cases of network flow problems are considered. At the very beginning, the problem of network flow was defined in general. The main part of the paper defines the shortest path problem, the maximum flow problem and the assignment problem. Also, the algorithms used in solving the mentioned network flow problems are explained in detail.

Key words:
directed graph, network flow, shortest path problem, Bellman-Ford algorithm, Dijkstra algorithm, maximum flow problem, Ford-Fulkerson algorithm, assignment problem, auction algorithm.

Životopis

Rođena sam 20.06.1990. u Rijeci. Završila sam OŠ Fran Krsto Frankopan na Krku, te srednju ekonomsku školu u Krku. Nakon srednje škole upisujem Preddiplomski studij matematike na Odjelu za matematiku Sveučilišta u Rijeci. Preddiplomski studij sam završila 2015. godine sa temom završnog rada "Šetnje, staze, putovi i ciklusi u usmjerenim grafovima" pod mentorstvom dr. sc. Deana Crnkovića. Iste godine upisujem diplomski studij na Odjelu za matematiku u Rijeci, nastavnički smjer. Nakon odslušanog semestra, odlučujem se iduće godine za upis diplomskog studija na Odjelu za matematiku u Osijeku, smjer Financijska matematika i statistika. Tijekom studija odradila sam stručnu praksu u Erste banci u Rijeci.