1. OPTIMIZACIJA NA MREŽAMA

1.1. Uvod

Modeli optimizacije na grafovima i mrežama predstavljaju značajnu oblast interesovanja u operacionim istraživanjima i primenjenoj mateatici. U poslednjih par decenija raste broj radova koji se u toku jedne godine objavi na temu mrežnih optimizacionih modela. Razlog tome su teorijski zanimljivi problemi koji nastaju pri formulisanju matematičkih modela za rešavanje važnih praktičnih zadataka. Pored onih koji se odnose na realne mreže, kao što su putne, električne, telekomunikacione, računarske i druge, postoji i čitav niz drugih problema koji se mogu formulisati kao zadaci optimizacije na grafu ili mreži. Na primer, upravljanje projektom se standardno radi uz pomoć mrežnog planiranja, tehnike CPM i PERT, neki zadaci u planiranju proizvodnje i raspoređivanja poslova mogu da se formulišu kao zadaci optimizacije na grafu itd.

Veliki broj problema optimizacije na mrežama može da se svede na neki od karakterističnih zadataka za koje postoje razvijeni efikasni algoritmi. Osnovne grupe standardnih zadataka i algoritmi kojima se mogu rešiti, dati u zagradi, prema jednoj klasifikaciji [44] su:

1. Nalaženje najkraćeg puta između dva zadata čvora u mreži (algoritmi Dijkstre i Belmana);
2. Nalaženje najkraćeg puta između jednog zadatog i svih ostalih čvorova u mreži (isti algoritmi);
3. Nalaženje najkraćeg puta između svaka dva čvora u mreži (Flojdov algoritam);
4. Nalaženje K najkraćih puteva između dva zadata čvora u mreži (Jenov algoritam);
5. Određivanje najkraćeg puta sa unapred zadatim brojem čvorova (algoritmi za rešavanje problema trgovačkog putnika);
6. Određivanje maksimalnih (minimalnih) protoka u mrežama (Ford-

-Falkersonov algoritam).

Problemi od 1 do 5 spadaju u grupu određivanja *optimalnih puteva* u mrežama, dok šesti predstavlja problem *ekstremizacije protoka* u mrežama. U tekstu koji sledi izložiće se navedeni modeli i algoritmi.

U rešavanju zadatka optimizacije na mrežama koriste se modeli i rezultati teorije grafova [13]. Pretpostavlja se da čitalac vlada osnovnim pojmovima iz ove teorije.

Graf ćemo obeležavati sa $G = (N, L)$, gde je $N = \{1, 2, \dots, n\}$ skup čvorova, a $L \subseteq \{(i,j) \mid i \in N, j \in N\}$ skup grana. Alternativno, graf se obeležava i sa $G = (N, \Gamma)$, gde je $N = \{1, 2, \dots, n\}$ takođe skup čvorova, a Γ preslikavanje skupa N na $P(N)$, gde je $P(N)$ partitivni skup skupa N, tj. skup svih podskupova skupa N. Ako se elementima skupa čvorova N i/ili elementima skupa grana L konačnog grafa $G = (N, L)$ pridruže određeni brojevi ili funkcije, onda se takav graf naziva *mreža*. Pritom se svakom čvoru ili grani može pridružiti jedan ili više brojeva.

U ovom tekstu koristićemo sledeće dodatne oznake:

c_{ij} - dužina (cena) grane (i, j);

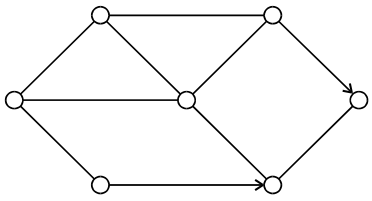
$\Gamma(i)$ - skup čvorova koji slede čvor i, tj. $\Gamma(i) = \{j \in N | (i, j) \in L\}$;

$\Gamma^{-1}(i)$ - skup čvorova koji prethode čvoru i, tj. $\Gamma^{-1}(i) = \{j \in N | (j, i) \in L\}$

s - početni (startni) čvor;

t - završni (ciljni) čvor.

U geometrijskoj predstavi grafa, slika 1.1, čvorovi se prikazuju kao tačke ili kružići, a grane kao orijentisane ili neorijentisane linije. Orijentacija linije se grafički prikazuje strelicom. Ako $(i,j) \in L$ i $(j, i) \in L$, tada je linija između tačaka i i j neorijentisana. Ako je $(i,j) \in L$, ali $(j, i) \notin L$, tada je linija između tačaka i i j orijentisana od i ka j.



Slika 1.1.

U modelima koji slede koristiće se, po pravilu, *potpuni grafovi*, tj. $L=\{(i,j) \mid i \in N, j \in N\}$ kao što se to obično radi u mrežnim modelima. Najčešće se polazi od pretpostavke da je mreža *povezana*. Za mrežu se kaže da je povezana ako između bilo koja dva čvora mreže postoji put. Ovom pretpostavkom se ne gubi na opštosti jer ako u realnom problemu ne postoji objekat (saobraćajnica, vod, linija i sl) koji odgovara nekoj grani (i, j), onda se toj grani pridruži težina c_{ij} takva da posmatrana grana ne može da utiče na optimalno rešenje. Na primer,

ako se rešava problem najkraćeg puta, staviće se $c_{ij} = \infty$, a ako se rešava problem protoka, biće $c_{ij} = 0$. Mrežu za koju važi da je $c_{ij} = c_{ji}$ za svako $(i, j) \in L$ zovemo neusmerenom ili simetričnom.

1.2. Nalaženje najkraćeg puta između dva zadata čvora u mreži

Neka je dat graf $G = (N, L)$ i neka su granama pridružene dužine c_{ij}, $(i, j) \in L$. Potrebno je naći najkraći put između zadatih čvorova s i t. *Put* između zadatih čvorova s i t definiše se kao niz grana od kojih prva polazi iz čvora s, svaka sledeća grana u nizu počinje u onom čvoru u kojem se završava prethodna, a poslednja se završava u čvoru t:

$$[(s, i_1), (i_1, i_2), \dots , (i_{k-1}, i_k), (i_k, t)]$$

Alternativno, put može da se definiše kao niz čvorova kroz koje prolazi:

$$(s, i_1, i_2, \dots , i_{k-1}, i_k, t)$$

Dužina puta se definiše kao zbir dužina grana koje pripadaju putu. Najkraći put između čvorova s i t je put najmanje dužine.

Za nalaženje najkraćih puteva kroz mrežu od interesa su samo elementarni putevi. *Elementarni put* je put koji kroz svaki čvor grafa prolazi najviše jedanput.

Dijkstrin algoritam

Ovaj algoritam [14] se smatra najefikasnijim za određivanje najkraćeg puta između dva čvora kada je ispunjen uslov da su dužine grana pozitivne, $c_{ij} > 0$, za svako $(i, j) \in L$.

Ideja algoritma i dokaz da se njime dolazi do optimalnog rešenja zasnivaju se na *principu optimalnosti* koji se za ovu priliku pojednostavljeno izražava stavom: optimalni put je i u delovima optimalan. U algoritmu se polazi od početnog čvora i iterativno približava krajnjem pri čemu se vodi računa da put kojim se ide zadovoljava princip optimalnosti.

U prvoj iteraciji se određuje čvor koji je najbliži početnom. U drugoj iteraciji treba odrediti čvor koji je sledeći po redu (drugi) najbliži čvor početnom čvoru. Do njega se može stići ili direktnom granom od početnog čvora, ili preko čvora za koji je u prethodnoj iteraciji utvrđeno da je najbliži početnom. Na taj način su određeni najkraći putevi od početnog do dva čvora u mreži. Dalje se proširuje skup čvorova do kojih su određeni najkraći putevi koristeći sledeće opšte pravilo: do nekog čvora se najkraćim putem dolazi ili direktno od početnog čvora ili preko nekog drugog čvora za koji je već određen najkraći put. Postupak se završava kada se utvrdi najkraći put do krajnjeg čvora.

U svrhu implementacije principa optimalnosti u Dijkstrinom algoritmu se koristi koncept *obeležavanja* čvorova. *Obeležje* ili oznaka $d(j)$ čvora j može biti *privremeno* (promenljivo) i piše se $d^-(j)$ ili *stalno* (nepromenljivo), kada se piše $d^+(j)$. Ovo poslednje predstavlja dužinu najkraćeg puta od početnog do čvora j.

Algoritam

1° Inicijalizacija
Čvorovima dodeljujemo početna obeležja na sledeći način:
- početnom čvoru s dodeljujemo stalno obeležje $d^+(s) = 0$;
- svim ostalim čvorovima dodeljujemo privremena obeležja $d^-(j) = \infty$, $j \in N \setminus \{s\}$;
- stavimo da je $i = s$.

2° Odrediti skup A_i čvorova koji slede čvor i i koji nemaju stalno obeležje
$$A_i = \{j \mid j \in \Gamma(i) \wedge d(j) = d^-(j)\}.$$

3° Za svako $j \in A_i$ odrediti nova privremena obeležja
$$d^-(j) = \min\{d^-(j), d^+(i) + c_{ij}\}$$

4° Od svih čvorova na mreži koji su obeleženi privremenim obeležjem, samo jedan j^* dobija stalno obeležje i to onaj za koji je
$$d^-(j^*) = \min_{j \in N} \{d^-(j)\},$$
pa je $d^+(j^*) = d^-(j^*)$.

5° Proveriti da li je $j^* = t$, tj. da li je završni čvor obeležen stalnim obeležjem.
Ako nije, staviti $i = j^*$ i vratiti se na korak 2°.
Ako jeste, određena je dužina najkraćeg puta $d^+(t)$ i sada treba rekonstruisati najkraći put kojim se od s stiglo do t.

6° Najkraći put $p = (s, j_1, j_2, \ldots, j_k, t)$ određujemo vraćajući se unazad od čvora t ka čvoru s: $t \to j_k \to j_{k-1} \to \ldots \to s$, tako da važi:

$$j_k: \quad d^+(t) - d^+(j_k) = c_{j_k t}$$

$$j_{k-1}: \quad d^+(j_k) - d^+(j_{k-1}) = c_{j_{k-1} j_k}$$

. . . . .
. . . . .

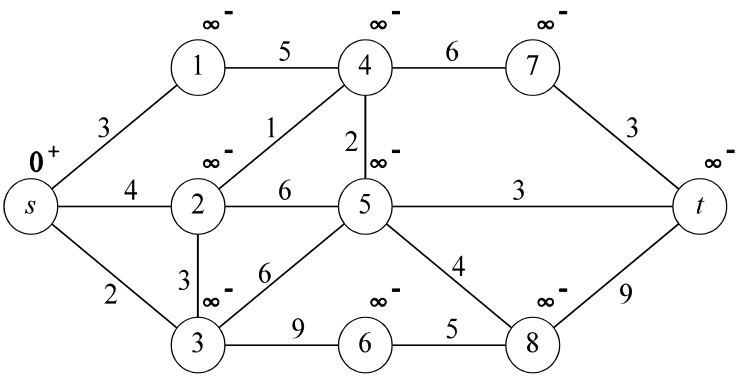
$$s: \quad d^+(j_1) - d^+(s) = c_{s j_1}$$

Jasno je da važi: $d^+(t) = c_{s j_1} + c_{j_1 j_2} + \ldots + c_{j_{k-1} j_k} + c_{j_k t}$. ■

Napomena: Poslednji korak algoritma, određivanje najkraćeg puta,

pojednostavljuje se ako se u koraku 4°, pored stalnog obeležja, zapamti i indeks čvora koji prethodi posmatranom čvoru i na osnovu kojeg je dobijeno to obeležje.

Primer 1.1. Zadata je mreža za koju je potrebno odrediti najkraći put od čvora *s* do čvora *t* koristeći Dijkstrin algoritam.



Slika 1.2.

Rešenje: Zadata mreža sa početnim oznakama je data na slici 1.2. Prelazimo na iterativni deo promene obeležja:

1) Čvor *s* slede čvorovi: 1, 2 i 3, $A_s = \{1, 2, 3\}$

$d^-(1) = \min\{0+3, \infty\} = 3$

$d^-(2) = \min\{0+4, \infty\} = 4$

$d^-(3) = \min\{0+2, \infty\} = 2$

$\min\{d^-(1), d^-(2), d^-(3)\} = d^-(3) = 2 \Rightarrow d^+(3) = d^-(3)$

2) Čvor 3 slede: 2, 5 i 6, $A_3 = \{2, 5, 6\}$

$d^-(2) = \min\{2+3, 4\} = 4$

$d^-(5) = \min\{2+6, \infty\} = 8$

$d^-(6) = \min\{2+9, \infty\} = 11$

$\min\{d^-(1), d^-(2), d^-(5), d^-(6)\} = d^-(1) = 3 \Rightarrow$

$\Rightarrow d^+(1) = d^-(1)$

3) $A_1 = \{4\}$

$d^-(4) = \min\{3+5, \infty\} = 8$

$\min\{d^-(2), d^-(4), d^-(5), d^-(6)\} = d^-(2) = 4 \Rightarrow$

$\Rightarrow d^+(2) = d^-(2)$

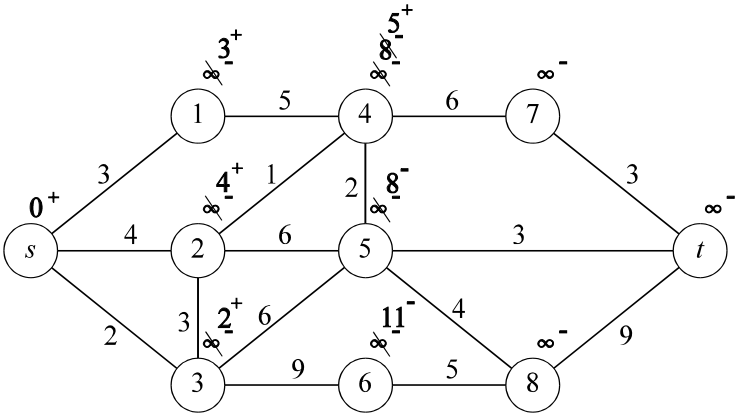
4) $A_2 = \{4, 5\}$

$d^-(4) = \min\{4+1, 8\} = 5$

$d^-(5) = \min\{4+6, 8\} = 8$

$\min\{d^-(4), d^-(5), d^-(6)\} = d^-(4) = 5 \Rightarrow d^+(4) = d^-(4)$

Stanje obeležja na grafu je, u ovom trenutku, kao na slici 1.3.



Slika 1.3.

5) $A_4 = \{5, 7\}$

$d^-(5) = \min\{5+2, 8\} = 7$

$d^-(7) = \min\{5+6, \infty\} = 11$

$\min\{d^-(5), d^-(6), d^-(7)\} = d^-(5) = 7 \Rightarrow d^+(5) = d^-(5)$

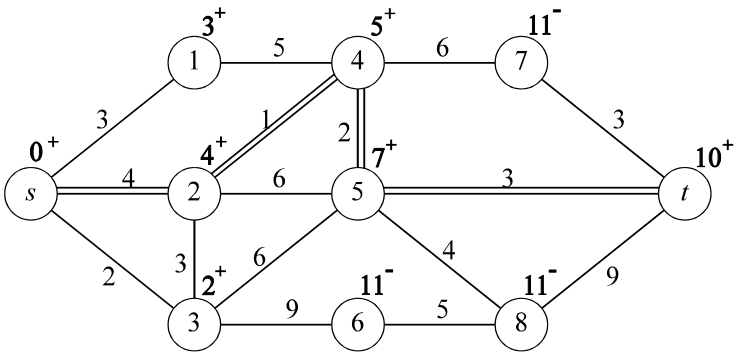
6) $A_5 = \{8, t\}$

$d^-(8) = \min\{7+4, \infty\} = 11$

$d^-(t) = \min\{7+3, \infty\} = 10$

$\min\{d^-(6), d^-(7), d^-(8), d^-(t)\} = d^-(t) = 10 \Rightarrow d^+(t) = d^-(t)$

$d^+(t)$ - nepromenljivo obeležje => KRAJ iterativnog postupka. Dužina najkraćeg puta od čvora s do t iznosi 10. Čvorovi na mreži su obeleženi kako je prikazano na slici 1.4.



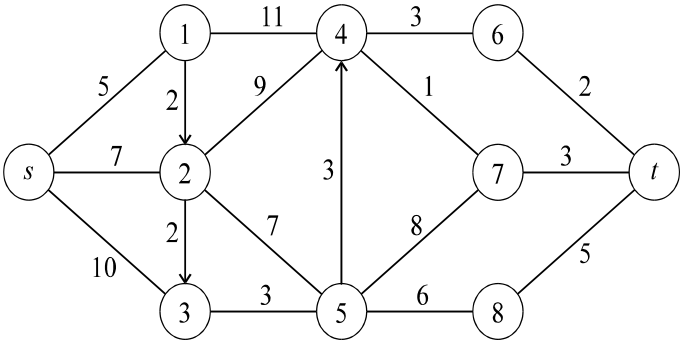
Slika 1.4.

Prelazimo na određivanje najkraćeg puta:
10 - 3 = 7 => čvor 5,
7 - 2 = 5 => čvor 4,
5 - 1 = 4 => čvor 2 i
4 - 4 = 0 => vratili smo se u početni čvor *s*.
Najkraći put u mreži je: (*s*,2, 4, 5, *t*).
Korisno je još proveriti da li dužina dobijenog puta odgovara obeležju $d^+(t)$: 3 + 2 + 1 + 4 = 10. ■

Potrebno je napomenuti da stalno obeležje čvora predstavlja dužinu najkraćeg puta od čvora *s* do tog čvora. Kada se rešava problem određivanja najkraćeg puta između početnog i svih ostalih čvorova u mreži, potrebno je produžiti postupak obeležavanja dok svi čvorovi ne dobiju stalna obeležja.

Pošto je ovo bio prvi zadatak ove vrste, postupak računanja smo izvodili postupno pišući svaku iteraciju zasebno. U sledećim primerima ćemo obeležja unositi na samu mrežu, vodeći računa o redosledu dodeljivanja ovih vrednosti.

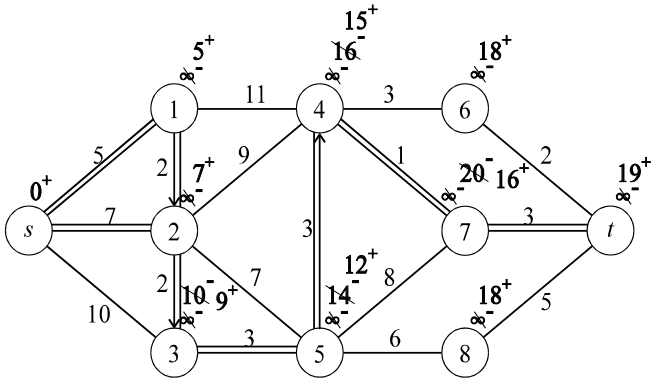
Primer 1.2. Data je mreža na slici 1.5:



Slika 1.5.

a) Odrediti najkraći put između čvorova *s* i *t* i njegovu dužinu koristeći Dijkstrin algoritam.
b) Odrediti dužine najkraćih puteva između čvora *s* i svih preostalih čvorova u mreži.

Rešenje: a) Zadatak ćemo rešiti na samoj mreži:



Slika 1.6.

Dužina najkraćeg puta je 19. Kod određivanja puta otkrivamo da je rešenje višestruko jer u čvoru 2 postoje dva prethodna čvora koja zadovoljavaju uslov: $d^+(2) - d^+(j) = c_{j2}$, gde j predstavlja susedne čvorove čvora 2. Ti putevi su:
I : (*s*, 1, 2, 3, 5, 4, 7, *t*)
II : (*s*, 2, 3, 5, 4, 7, *t*).

b) Ovaj deo zadatka je implicitno već rešen u postupku pod (a).

Vidimo da su svi čvorovi na mreži (slika 1.6.) obeleženi stalnim obeležjima koja predstavljaju dužine najkraćih puteva od čvora s do njih. Dakle, odgovarajuće dužine iznose:

s - 1	: 5	s - 5	: 12	s - t	: 19
s - 2	: 7	s - 6	: 18		
s - 3	: 9	s - 7	: 16		
s - 4	: 15	s - 8	: 18		

Pošto se pod (b) tražilo da se odrede samo dužine puteva, ovim je zadatak rešen. ■

Belmanov algoritam

Za primenu Belmanovog algoritma [4] ne postoji ograničenje da dužine grana c_{ij}, $(i,j) \in L$ budu pozitivne. To znači da one mogu da budu i manje od nule, ali u mreži ne sme da postoji kontura negativne dužine. Kada bi takva kontura postojala, obilaženje te konture bi smanjivalo dužinu puta; to znači da u tom slučaju, bez dodatnih uslova, zadatak ne bi imao rešenje.

U Belmanovom algoritmu se kao i u Dijkstrinom, koristi koncept obeležavanja čvorova. Ovde se obeležja vezuju za iteracije i važno je voditi računa o obeležjima koja su se promenila u prethodnoj iteraciji. Do rešenja se dolazi u najviše n - 1 iteracija, a u slučaju da u mreži postoji kontura negativne dužine, to će biti otkriveno.

U ovom algoritmu koristimo sledeće dodatne oznake:
k - brojač iteracija, $k = 1, \ldots, n\text{-}1$;
$d^k(i)$ - obeležje čvora i u k-toj iteraciji;
$B_i = \Gamma^{-1}(i)$ - skup čvorova koji prethode čvoru i;
R - skup čvorova kojima je u prethodnoj operaciji promenjeno obeležje;
$\Gamma(R)$ - skup svih čvorova koji slede čvorove iz skupa R.

Algoritam

1° Inicijalizacija:
$k = 1$; formirati skup $R = \Gamma(s)$;
$d^1(s) = 0$;
$d^1(i) = c_{si}$, $i \in R$;
$d^1(i) = \infty$, $\forall\, i \in N \setminus R \setminus \{s\}$.

2° Odrediti skup čvorova $\Gamma(R)$.

3° Za svako $i \in \Gamma(R)$, odrediti skup čvorova koji prethode čvoru i, $B_i = \Gamma^{-1}(i)$ i izračunati novo obeležje

$$d^{k+1}(i) = \min\left\{d^k(i),\ \min_{j \in B_i \cap R}\left[d^k(j) + c_{ji}\right]\right\} .$$

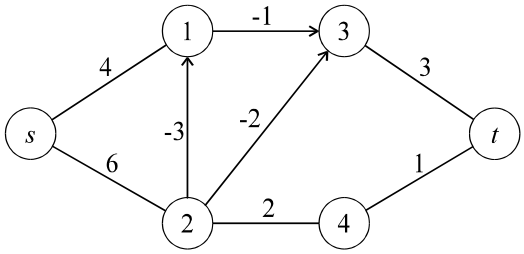
4° Formirati skup čvorova kojima je u prethodnom koraku promenjeno obeležje

$$R = \{i \mid d^{k+1}(i) \neq d^k(i),\ i \in N\}.$$

5° Provera:
Ako je $R = \varnothing$, pronađeno je optimalno rešenje ($d^k(i)$ predstavlja minimalnu dužinu puta od čvora s do čvora $i \in N$; $d^k(t)$ predstavlja rešenje postavljenog problema), preći na korak 6°;
ako je $R \neq \varnothing$, proveriti da li je $k < n - 1$:
ako jeste, $k \leftarrow k + 1$, vratiti se na korak 2°;
ako je $k = n - 1$, tada u mreži postoji kontura negativne dužine i ne postoji konačno rešenje problema; KRAJ.

6° Rekonstruisati najkraći put kao u slučaju Dijkstrinog algoritma. ■

Primer 1.3. Odrediti najkraći put i njegovu dužinu za čvorove s i t mreže prikazane na slici 1.7.



Slika 1.7.

Rešenje: $k = 1$, $R = \{1, 2\}$

$d^1(s) = 0$ $\quad d^1(3) = \infty$
$d^1(1) = 4$ $\quad d^1(4) = \infty$
$d^1(2) = 6$ $\quad d^1(t) = \infty$

Čvorovi koji slede skup čvorova R su: $\Gamma(R) = \{s, 1, 3, 4\}$ i za njih računamo nove oznake:

$d^2(s) = \min\{0, \min[4+4, 6+6]\} = 0$

$d^2(1) = \min\{4, \min[0+4, 6-3]\} = \min\{4, 3\} = 3$ ☜

$d^2(3) = \min\{\infty, 4-1, 6-2\} = 3$ ☜

$d^2(4) = \min\{\infty, 6+2\} = 8$ ☜

Pošto je došlo do promene bar jednog $d^{k+1}(i)$ (označeni sa „☜“), nastavljamo sa iteracijama.

$k = 2$, $R = \{1, 3, 4\}$
$\Gamma(R) = \{s, 2, 3, t\}$
$d^3(s) = \min\{0, 4+4, 6+6\} = 0$
$d^3(2) = \min\{6, 8+2\} = 6$
$d^3(3) = \min\{3, 6-2, 3-1\} = 2$ ☜
$d^3(t) = \min\{\infty, 3+3, 8+1\} = 6$ ☜

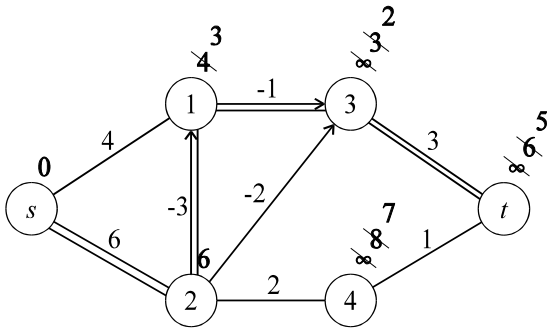
$k = 3$, $R = \{3, t\}$
$\Gamma(R) = \{3, 4, t\}$
$d^4(3) = \min\{2, 6-2, 3-1, 6+3\} = 2$
$d^4(4) = \min\{8, 6+2, 6+1\} = 7$ ☜
$d^4(t) = \min\{6, 2+3, 8+1\} = 5$ ☜

$k = 4$, $R = \{4, t\}$
$i \in \{2, 3, 4, t\}$
$d^5(2) = \min\{6, 7+2\} = 6$
$d^5(3) = \min\{2, 6-2, 3-1, 5+3\} = 2$
$d^5(4) = \min\{7, 6+2, 5+1\} = 6$ ☜
$d^5(t) = \min\{5, 2+3, 8+1\} = 5$

$k = 5$, $R = \{4\}$
$i \in \{2, t\}$
$d^6(2) = \min\{6, 7+2\} = 6$
$d^6(t) = \min\{5, 2+3, 7+1\} = 5$

Pošto u poslednjoj iteraciji nije promenjeno ni jedno obeležje, a k je manje od $n = 6$, zaključujemo da je postupak završen. Dužina najkraćeg puta od čvora s do t iznosi 5, a put je: (s, 2, 1, 3, t). Poslednje obeležje čvora predstavlja dužinu najkraćeg puta od čvora s do tog čvora.

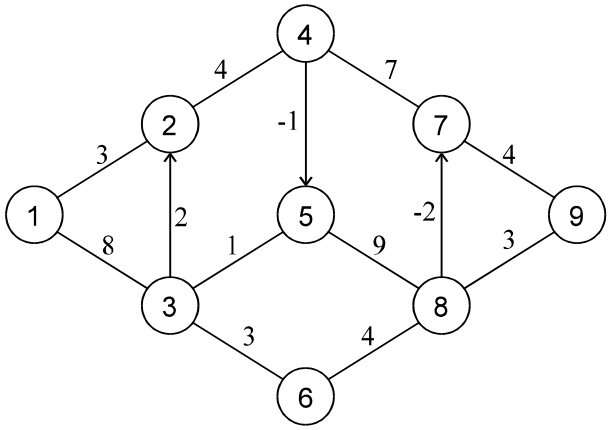
Postupak određivanja puta i njegove dužine se može sprovesti i na samoj mreži:



Slika 1.8.

■

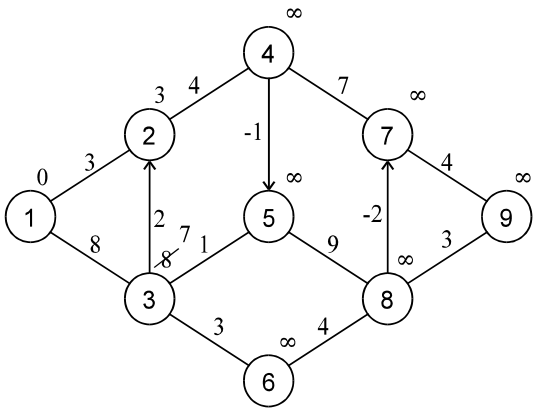
Primer 1.4. Data je sledeća mreža:



Slika 1.9.

a) Naći najkraći put od čvora 1 do 9.
b) Naći najkraći put od čvora 9 do 1.

Rešenje: a)



Slika 1.10.

Najkraći put od čvora 1 do 9 je:
(1, 2, 4, 5, 3, 6, 8, 7, 9); dužina puta je 16.

b) Najkraći put od čvora 9 do 1 je: (9, 8, 7, 4, 5, 3, 2, 1); dužina puta je 13. ■

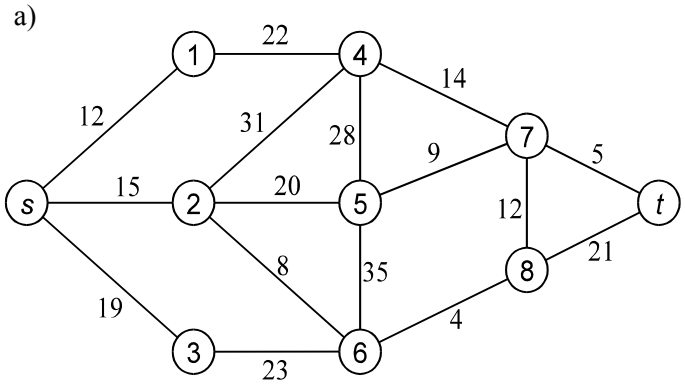
Primer 1.5. Zadata je neorijentisana mreža sledećim rastojanjima između čvorova:

Od čvora:	*s*	*s*	*s*	1	2	2	2	3	4	4	5	5	6	7	7	8
Do čvora:	1	2	3	4	4	5	6	6	5	7	6	7	8	8	*t*	*t*
Udaljenost:	12	15	19	22	31	20	8	23	28	14	35	9	4	12	5	21

Potrebno je:

a) Nacrtati zadatu mrežu (čvorove numerisati, a grane obeležiti njihovim dužinama).
b) Naći najkraći put između čvorova *s* i *t*.
c) Naći najkraći put između čvorova 1 i 8.

Rešenje: a)



Slika 1.11.

Ovaj zadatak se može rešiti bilo algoritmima Dijkstre ili Belmana, a dobija se sledeće rešenje:

b) Najkraći put od *s* do *t* je (*s*, 2, 6, 8, 7, *t*) dužine 44.

c) Najkraći put od 1 do 8 je (1, *s*, 2, 6, 8) dužine 39. ■

1.3. Nalaženje najkraćih puteva između svaka dva čvora u mreži

Ovaj problem je moguće rešiti uzastopnom primenom Dijkstrinog ili Belmanovog algoritma *n* puta (za svaki čvor po jednom). Svako od *n* dobijenih rešenja predstavljalo bi udaljenost između *k*-tog ($k = 1, 2, \ldots, n$) čvora i svih ostalih. Problem nalaženja najkraćih puteva između svaka dva čvora u mreži može se efikasnije rešiti posebnim algoritmima koji su takođe razvijeni na principu optimalnosti.

Flojdov algoritam

Definisaćemo *matricu rastojanja čvorova* $C = (c_{ij})$ čiji elementi c_{ij} predstavljaju dužinu grana između čvorova *i* i *j*. Uvešćemo sledeće pretpostavke o dužinama grana, tj. o vrednostima elemenata matrice *C*:

1) $c_{ij} \neq 0 \; \forall \; (i, j) \in L$
2) $c_{ii} = 0 \; \forall \; i \in N$
3) $c_{ij} = \infty \; \forall \; (i, j) \notin L, \; i \neq j.$

U Flojdovom algoritmu [17] se koristi kvadratna matrica $C^k = (c_{ij}^{\,k})_{n \times n}$, $k = 1, 2, \ldots, n$, čiji elementi $c_{ij}^{\,k}$ predstavljaju dužine najkraćih puteva između čvorova *i* i *j*, ali samo za $i, j \in \{1, 2, \ldots, k\}$. Kada je $k = n$, tada će matrica C^n

predstavljati dužine najkraćih puteva između svaka dva čvora u mreži.

Flojdov algoritam se može primeniti i kada su dužine grana negativne, ali u mreži ne sme da postoji kontura negativne dužine. Zato grane sa negativnim dužinama moraju biti orijentisane.

Algoritam:

1° Postaviti $k = 0$ i formirati matricu C^o takvu da je $C^o = C$ tj. $c_{ij}^{\,o} = c_{ij}$, $\forall\, i, j = 1, 2, \ldots, n$.

2° $k = k + 1$;

3° Odrediti skupove I i J takve da:

$$I = \{i \mid i \neq k\,,\ c_{ik}^{\,k} \neq \infty\}$$
$$J = \{j \mid j \neq k\,,\ c_{kj}^{\,k} \neq \infty\}$$

4° Izračunati elemente matrice C^k po formuli:

$$c_{ij}^{\,k} = \min\left\{c_{ij}^{\,k-1},\ c_{ik}^{\,k-1} + c_{kj}^{\,k-1}\right\} \quad \forall\ i \in I \text{ i } \forall\ j \in J\ ,$$

dok ostali elementi ostaju nepromenjeni.

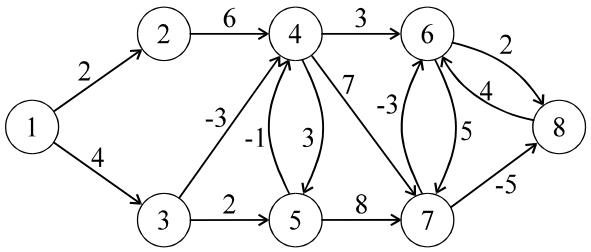
5° Mogu da nastupe tri slučaja:

(a) ako je $k < n$ i $c_{ii}^{\,k} \geq 0\ \forall\ i \in N$, ići na korak 2°;

(b) ako je $k \leq n$ i $\exists\ i \in N : c_{ii}^{\,k} < 0$, tada u mreži postoji kontura negativne dužine i nije moguće dobiti konačno rešenje;

(c) ako je $k = n$ i $c_{ii}^{\,n} \geq 0\ \forall\ i \in N$ => KRAJ; matrica C^n predstavlja rešenje problema. ■

Primer 1.6. Za datu mrežu naći dužine najkraćih puteva između svaka dva čvora.



Slika 1.12.

Rešenje:

$k = 0$

$c_{ij}^{\,o} = c_{ij}\ \forall\ (i, j) \in L$

$c_{ii}^{\,o} = 0\ \forall\ i \in N$

$c_{ij}^{\,o} = \infty\ \forall\ (i, j) \notin L$

$$C^0 = C = \begin{array}{c} \\ J\rightarrow \\ \\ \\ \\ \\ \\ \\ \end{array} \begin{array}{c} \begin{array}{cccccccc} I & & & & & & & \\ \downarrow & & & & & & & \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array} \\ \begin{bmatrix} 0 & 2 & 4 & \infty & \infty & \infty & \infty & \infty \\ \infty & 0 & \infty & 6 & \infty & \infty & \infty & \infty \\ \infty & \infty & 0 & -3 & 2 & \infty & \infty & \infty \\ \infty & \infty & \infty & 0 & 3 & 3 & 7 & \infty \\ \infty & \infty & \infty & -1 & 0 & \infty & 8 & \infty \\ \infty & \infty & \infty & \infty & \infty & 0 & 5 & 2 \\ \infty & \infty & \infty & \infty & \infty & -3 & 0 & -5 \\ \infty & \infty & \infty & \infty & \infty & 4 & \infty & 0 \end{bmatrix} \end{array}$$

$k = 1$
$I = \varnothing$ (elementi k-te kolone različiti od ∞, a da se ne nalaze na dijagonali),
$J = \{2, 3\}$ (elementi k-tog reda različiti od ∞, a da se ne nalaze na dijagonali)
Pošto je $I = \varnothing$, matrica C^1 će ostati ista kao C^0.

$k = 2$, $I = \{1\}$, $J = \{4\}$, razmatramo samo element:

$$c_{14}^2 = \min\left\{c_{14}^1,\ c_{12}^1 + c_{24}^1\right\} = \min\{\infty,\ 2+6\} = 8$$

C^2 se razlikuje od C^0 i C^1 samo u elementu c_{14} koji je bio ∞, a sada je 8.

$k = 3$, $I = \{1\}$, $J = \{4, 5\}$, posmatramo:

$$c_{14}^3 = \min\left\{c_{14}^2,\ c_{13}^2 + c_{34}^2\right\} = \min\{8,\ 4-3\} = 1$$

$$c_{15}^3 = \min\left\{c_{15}^2,\ c_{13}^2 + c_{35}^2\right\} = \min\{\infty,\ 4+2\} = 6$$

$$C^3 = \begin{bmatrix} 0 & 2 & 4 & 1 & 6 & \infty & \infty & \infty \\ \infty & 0 & \infty & 6 & \infty & \infty & \infty & \infty \\ \infty & \infty & 0 & -3 & 2 & \infty & \infty & \infty \\ \infty & \infty & \infty & 0 & 3 & 3 & 7 & \infty \\ \infty & \infty & \infty & -1 & 0 & \infty & 8 & \infty \\ \infty & \infty & \infty & \infty & \infty & 0 & 5 & 2 \\ \infty & \infty & \infty & \infty & \infty & -3 & 0 & -5 \\ \infty & \infty & \infty & \infty & \infty & 4 & \infty & 0 \end{bmatrix}$$

$k = 4$, $I = \{1, 2, 3, 5\}$, $J = \{5, 6, 7\}$, izračunavamo nove vrednosti za 12 elemenata matrice C^4:

$$c_{15}^4 = \min\left\{c_{15}^3,\ c_{14}^3 + c_{45}^3\right\} = \min\{6,\ 1+3\} = 4$$

$$c_{16}^4 = \min\left\{c_{16}^3,\ c_{14}^3 + c_{46}^3\right\} = \min\{\infty,\ 1+3\} = 4$$

$$c_{17}^4 = \min\left\{c_{17}^3,\ c_{14}^3 + c_{47}^3\right\} = \min\{\infty,\ 1+7\} = 8$$

$$c_{25}^4 = \min\left\{c_{25}^3,\ c_{24}^3 + c_{45}^3\right\} = \min\{\infty,\ 6+3\} = 9$$

$$c_{26}^4 = \min\left\{c_{26}^3,\ c_{24}^3 + c_{46}^3\right\} = \min\{\infty,\ 6+3\} = 9$$

$$c_{27}^4 = \min\left\{c_{27}^3,\ c_{24}^3 + c_{47}^3\right\} = \min\{\infty,\ 6+7\} = 13$$

$$c_{35}^4 = \min\left\{c_{35}^3,\ c_{34}^3 + c_{45}^3\right\} = \min\{2,\ -3+3\} = 0$$

$$c_{36}^4 = \min\left\{c_{36}^3,\ c_{34}^3 + c_{46}^3\right\} = \min\{\infty,\ -3+3\} = 0$$

$$c_{37}^4 = \min\left\{c_{37}^3,\ c_{34}^3 + c_{47}^3\right\} = \min\{\infty,\ -3+7\} = 4$$

$$c_{55}^4 = \min\left\{c_{55}^3,\ c_{54}^3 + c_{45}^3\right\} = \min\{0,\ -1+3\} = 0$$

$$c_{56}^{4} = \min\left\{c_{56}^{3},\ c_{54}^{3} + c_{46}^{3}\right\} = \min\{\infty, -1+3\} = 2$$

$$c_{57}^{4} = \min\left\{c_{57}^{3},\ c_{54}^{3} + c_{47}^{3}\right\} = \min\{8, -1+7\} = 6$$

$$C^{4} = \begin{bmatrix} 0 & 2 & 4 & 1 & 4 & 4 & 8 & \infty \\ \infty & 0 & \infty & 6 & 9 & 9 & 13 & \infty \\ \infty & \infty & 0 & -3 & 0 & 0 & 4 & \infty \\ \infty & \infty & \infty & 0 & 3 & 3 & 7 & \infty \\ \infty & \infty & \infty & -1 & 0 & 2 & 6 & \infty \\ \infty & \infty & \infty & \infty & \infty & 0 & 5 & 2 \\ \infty & \infty & \infty & \infty & \infty & -3 & 0 & -5 \\ \infty & \infty & \infty & \infty & \infty & 4 & \infty & 0 \end{bmatrix}$$

Zbog opširnosti ćemo preskočiti sledećih nekoliko iteracija. Za $k = 7$ se dobija sledeća matrica rastojanja:

$$C^{7} = \begin{bmatrix} 0 & 2 & 4 & 1 & 4 & 4 & 8 & 3 \\ \infty & 0 & \infty & 6 & 9 & 9 & 13 & 8 \\ \infty & \infty & 0 & -3 & 0 & 0 & 4 & -1 \\ \infty & \infty & \infty & 0 & 3 & 3 & 7 & 2 \\ \infty & \infty & \infty & -1 & 0 & 2 & 6 & 1 \\ \infty & \infty & \infty & \infty & \infty & 0 & 5 & 0 \\ \infty & \infty & \infty & \infty & \infty & -3 & 0 & -5 \\ \infty & \infty & \infty & \infty & \infty & 4 & 9 & 0 \end{bmatrix}$$

$k = 8$, $I = \{1, 2, 3, 4, 5, 6, 7\}$, $J = \{6, 7\}$. Za sledeće elemente matrice C^{8} izračunavamo nove vrednosti:

$c_{16}^{8} = 4$ $\qquad$ $c_{17}^{8} = 8$

$c_{26}^{8} = 9$ $\qquad$ $c_{27}^{8} = 13$

$c_{36}^{8} = 0$ $c_{37}^{8} = 4$

$c_{46}^{8} = 3$ $c_{47}^{8} = 7$

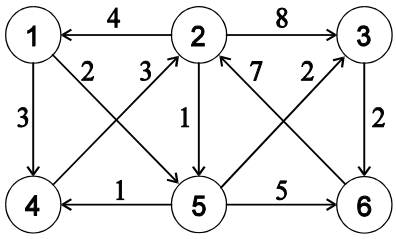
$c_{56}^{8} = 2$ $c_{57}^{8} = 6$

$c_{66}^{8} = 0$ $c_{67}^{8} = 5$

$c_{76}^{8} = -3$ $c_{77}^{8} = 0$

Možemo konstatovati da je $C^8 = C^7$. Pošto je $k = n$ i svi $c_{ii}^{\ 8} = 0$, $i = 1, \ldots, 8$ => KRAJ. Poslednja matrica predstavlja rešenje zadatka. ■

Primer 1.7. Data je mreža:



Slika 1.13.

Naći najkraće puteve između svaka dva čvora u mreži.

Rešenje:

$$C^0 = C = \begin{bmatrix} 0 & \infty & \infty & 3 & 2 & \infty \\ 4 & 0 & 8 & \infty & 1 & \infty \\ \infty & \infty & 0 & \infty & \infty & 2 \\ \infty & 3 & \infty & 0 & \infty & \infty \\ \infty & \infty & 2 & 1 & 0 & 5 \\ \infty & 7 & \infty & \infty & \infty & 0 \end{bmatrix}$$

$k = 1, I = \{2\}, J = \{4, 5\}$
$c_{24} = \min\{\infty, 4+3\} = 7$ *
$c_{25} = \min\{1, 4+2\} = 1$

U matrici je promenjena vrednost jedino elementa c_{24} (označen zvezdicom) i to sa ∞ na 7. Svi ostali elementi matrice C^1 su isti kao elementi matrice C^0.

$k = 2, I = \{4, 6\}, J = \{1, 3, 4, 5\}$
$c_{41} = \min\{\infty, 3+4\} = 7$ *
$c_{43} = \min\{\infty, 3+8\} = 11$ *
$c_{44} = \min\{0, 3+7\} = 0$
$c_{45} = \min\{\infty, 3+1\} = 4$ *
$c_{61} = \min\{\infty, 7+4\} = 11$ *
$c_{63} = \min\{\infty, 7+8\} = 15$ *
$c_{64} = \min\{\infty, 7+7\} = 14$ *
$c_{65} = \min\{\infty, 7+1\} = 8$ *

Matrica C^2 ima oblik:

$$C^2 = \begin{bmatrix} 0 & \infty & \infty & 3 & 2 & \infty \\ 4 & 0 & 8 & 7 & 1 & \infty \\ \infty & \infty & 0 & \infty & \infty & 2 \\ 7 & 3 & 11 & 0 & 4 & \infty \\ \infty & \infty & 2 & 1 & 0 & 5 \\ 11 & 7 & 15 & 14 & 8 & 0 \end{bmatrix}$$

U sledećim koracima biće date samo matrice C^k, $k = 3, ...$

$$C^3 = \begin{bmatrix} 0 & \infty & \infty & 3 & 2 & \infty \\ 4 & 0 & 8 & 7 & 1 & 10 \\ \infty & \infty & 0 & \infty & \infty & 2 \\ 7 & 3 & 11 & 0 & 4 & 13 \\ \infty & \infty & 2 & 1 & 0 & 4 \\ 11 & 7 & 15 & 14 & 8 & 0 \end{bmatrix} \qquad C^4 = \begin{bmatrix} 0 & 6 & 14 & 3 & 2 & 16 \\ 4 & 0 & 8 & 7 & 1 & 10 \\ \infty & \infty & 0 & \infty & \infty & 2 \\ 7 & 3 & 11 & 0 & 4 & 13 \\ 8 & 4 & 2 & 1 & 0 & 4 \\ 11 & 7 & 15 & 14 & 8 & 0 \end{bmatrix}$$

$$C^5 = \begin{bmatrix} 0 & 6 & 4 & 3 & 2 & 6 \\ 4 & 0 & 3 & 2 & 1 & 5 \\ \infty & \infty & 0 & \infty & \infty & 2 \\ 7 & 3 & 6 & 0 & 4 & 8 \\ 8 & 4 & 2 & 1 & 0 & 4 \\ 11 & 7 & 10 & 9 & 8 & 0 \end{bmatrix} \qquad C^6 = \begin{bmatrix} 0 & 6 & 4 & 3 & 2 & 6 \\ 4 & 0 & 3 & 2 & 1 & 5 \\ 13 & 9 & 0 & 11 & 10 & 2 \\ 7 & 3 & 6 & 0 & 4 & 8 \\ 8 & 4 & 2 & 1 & 0 & 4 \\ 11 & 7 & 10 & 9 & 8 & 0 \end{bmatrix}$$

KRAJ. Matrica C^6 je rešenje zadatka. ■

1.4. Nalaženje *K* najkraćih puteva u mreži

U rešavanju praktičnih problema određivanja puta kroz mrežu nisu retki slučajevi da se pored najkraćeg razmatra i sledeći po redu najkraći put, odnosno nekoliko najkraćih puteva. Razlozi mogu biti različiti, npr. preopterećenost ili neraspoloživost optimalnog puta ili njegovog dela, kao kod prekida ili zauzetosti telekomunikacionih linija, nepovoljna karakteristika najkraćeg puta po nekom drugom kriterijumu, itd. Pored toga, neki problemi asignacije i celobrojnog programiranja mogu da se formulišu kao zadaci nalaženja *K*-najkraćih puteva.

Ponovo posmatrajmo graf $G=(N, L)$. Ovde je pogodno čvorove obeležiti sa x_i, a grane sa (x_i, x_j). Svakoj grani $(x_i, x_j) \in L$ pridružujemo dužinu $c(x_i, x_j)$. Ako $(x_i, x_j) \notin L$ onda je $c(x_i, x_j) = \infty$. Zadatak je naći K najkraćih puteva između dva zadata čvora s i t, gde $s \in N$ i $t \in N$. Drugim rečima, potrebno je naći najkraći put između zadatih čvorova, zatim sledeći po redu najkraći i tako dalje dok se ne nađe K najkraćih puteva.

Neka je najkraći put $p^1 = (x_1^1, x_2^1, ..., x_i^1, ... , x_{q1}^1)$ pri čemu su $x_1^1 = s$, a $x_{q1}^1 = t$ i njegova dužina $d(p^1) = d_1$. Uopšte, neka je k-ti po redu najkraći put $p^k = (x_1^k, x_2^k, ... , x_i^k, ... , x_{qk}^k)$ pri $x_1^k = s$, $x_{qk}^k = t$, $d(p^k) = d_k$, $k = 1, ... , K$ i $d_1 \le d_2 \le ... \le d_k \le ... \le d_K$. Zadatak je odrediti puteve $p^1, p^2, ... , p^k, ... , p^K$ i njihove dužine $d_1, d_2, ... , d_k, ... d_K$.

Jenov algoritam

Za određivanje K najkraćih puteva u mreži smatra se da je najpogodniji i najopštiji algoritam Jena [49]. I ovaj algoritam počiva na principu optimalnosti i sistematizovanom pretraživanju mogućih alternativa.

Pretpostavimo da smo odredili najkraći put $p^1 = (x_1^1, x_2^1, ... , x_i^1, ...$

... , $x_{q_1}^1$). U svakom čvoru na ovom putu, sem u poslednjem, dakle za $i = 1, 2, \ldots, q_1\text{-}1$, može se skrenuti od najkraćeg puta tako što bi se od čvora x_i^1 nastavilo, ne ka čvoru x_{i+1}^1, nego ka nekom drugom čvoru. Ovaj čvor se obeležava sa x_{i+1}^2, a određuje se tako da put od x_i^1 do t bude minimalan ali da ne sadrži granu (x_i^1, x_{i+1}^1); to se može postići privremenim uklanjanjem ove grane iz grafa, odnosno stavljanjem da je $c(x_i^1, x_{i+1}^1) = \infty$. Tako dobijeni put zove se *odstupanje* (devijacija) od puta p^1 u čvoru x_i^1 i obeležava se sa p_i^2. Očigledno je da od svih odstupanja p_i^2, $i = 1, 2, \ldots, q_1\text{-}1$, ono za koje je $d(p_i^2)$ najmanje, upravo drugi po redu najkraći put, tj.

$$d(p^2) = \min\{d(p_i^2) \mid i = 1, \ldots, q_1\text{-}1\}.$$

U opštem slučaju, odstupanje se definiše na sledeći način: odstupanje p_i^k od puta p^{k-1} u čvoru x_i^{k-1} je put od s do t koji se poklapa sa putem p^{k-1} od početnog čvora s do čvora x_i^{k-1}, a dalje ide ka čvoru x_{i+1}^k koji sledi x_i^{k-1} i koji se razlikuje od čvora x_{i+1}^{k-1} ; pri tome se od čvora x_{i+1}^k do čvora t ide najkraćim putem. Kojim putem će se nastaviti od čvora x_i^{k-1} do t određuje se tako što se uklone sve grane koje polaze iz čvora x_i^{k-1}, a završavaju u čvoru koji je u nekoj od prethodno dobijenih odstupanja p_j^r, $r \in \{1, \ldots, k\text{-}1\}$, $j \in \{1, \ldots, q_r\text{-}1\}$, sledio čvor x_i^{k-1}. Tada se nađe najkraći put od čvora x_i^{k-1} do t.

U implementaciji algoritma koriste se dve liste: L_0 i L_1. Na listi L_0 nalaze se najkraći putevi, a na listi L_1 odstupanja. Za nalaženje puta p^k potrebno je prethodno odrediti sve puteve $p^1, p^2, \ldots, p^{k-1}$ i njihova odstupanja. Najkraće od odstupanja je sledeći po redu najkraći put p^k.

Algoritam

1° Inicijalizacija:
- primeniti neki od algoritama za nalaženje najkraćeg puta od s do t;
- ako postoji samo jedan najkraći put, staviti ga na listu L_0, postaviti brojač $k = 1$ i preći na korak 2°;
- ako postoji $h < K$ najkraćih puteva, staviti na listu L_0 svih h puteva, brojač $k = h$ i peći na korak 2°;
- ako postoji $h \geq K$ najkraćih puteva, staviti na listu L_0 proizvoljnih K i završiti algoritam.

2° $k = k + 1$;
Naći sva odstupanja p_i^k od $(k\text{-}1)$-og najkraćeg puta p^{k-1}, za $i = 1, \ldots, q_{k-1}\text{-}1$ izvršavajući korake od 3° do 6°.

3° Proveriti da li se potput (deo puta) koji sačinjavaju prvih i čvorova puta p^{k-1} poklapa sa potputem koji sačinjavaju prvih i čvorova bilo kojeg od puteva sa liste L_0 ili L_1, tj. bilo kojeg p_j^r, $r \in \{1, \ldots, k\text{-}1\}$, $j \in \{1, \ldots, q_r\text{-}1\}$. Ako je ovaj uslov ispunjen za neko r, staviti da je $c(x_i^{k-1}, x_{i+1}^r) = \infty$ (jer je već ranije određeno odstupanje od tog čvora); u suprotnom, ništa se ne menja.

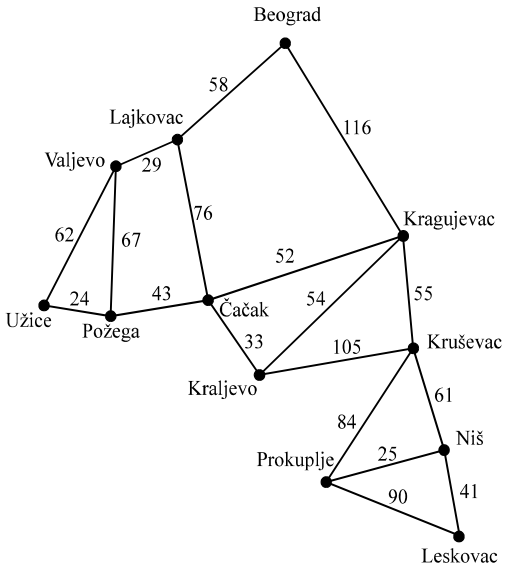
4º Isključujući iz razmatranja čvorove s, x_2^{k-1}, x_3^{k-1}, ... , x_{i-1}^{k-1}, korišćenjem nekog od poznatih algoritama naći najkraći put od x_i^{k-1} do t, i obeležiti ih sa $S_i^k = (x_i^{k-1}, x_{I+1}^{k}, \dots , t)$. (Ako se otkrije više najkraćih puteva od x_i^{k-1} do t, uzeti proizvoljno jedan od njih.)

5º Formirati odstupanje p_i^k, tako što se S_i^k put nadoveže na $(s, x_2^{k-1}, x_3^{k-1}, \dots , x_i^{k-1})$ i prebaciti p_i^k na listu L_1.

6º Ako su u koraku 3º menjane vrednosti za $c(x_i, x_j)$, sada ih vratiti na originalne vrednosti i vratiti se na korak 3º za sledeće i.

7º Naći najkraći put na listi L_1. Označiti taj put sa p^k i prebaciti ga sa liste L_1 na listu L_0.
Ako je $k = K$, završiti algoritam, na listi L_0 se nalazi tačno K najkraćih puteva; u suprotnom vratiti se na korak 2º. ■

Primer 1.8. Data je deo putne mreže Srbije. Udaljenosti između pojedinih gradova su date u kilometrima. Potrebno je odrediti 4 najkraća puta između Beograda i Leskovca.



Slika 1.14.

Rešenje: Primenom Jenovog algoritma dobijeno je rešenje prikazano u sledećoj tabeli.

Rb	Duž. puta	Put
1	237	Beograd - Kragujevac - Kruševac - Niš - Leskovac
2	317	Beograd - Kragujevac - Kruševac - Prokuplje - Leskovac
3	319	Beograd - Kragujevac - Kruševac - Niš - Prokuplje - Leskovac
4	321	Beograd - Kragujevac - Kruševac - Prokuplje - Niš - Leskovac

Napomena: Ovako dobijeno rešenje se može koristiti kada treba da se odredi najkraći put u slučajevima da je neka od deonica u mreži neraspoloživa. Na primer, ako znamo da je put Niš - Leskovac trenutno zatvoren, ovu deonicu ćemo najekonomičnije zaobići idući preko Prokuplja. Ako bismo hteli da zaobiđemo deonicu Beograd - Kragujevac, morali bismo da pronađemo još sledećih po redu najkraćih puteva. Prvi od njih koji zadovolji postavljene zahteve biće traženi put. ■

1.5. Najkraće razapinjuće stablo (SST)

Posmatramo mrežu datu sa $G = (N, L)$ i $C = (c_{ij})$, $(i, j) \in L$ pri čemu je $c_{ij} = c_{ji}$. Razapinjuće stablo grafa G je povezan podgraf grafa G, takav da sadrži sve čvorove kao i G i ne sadrži ni jednu konturu. Očigledno da je broj grana u razapinjućem stablu $l = n - 1$.

Problem određivanja *najkraćeg razapinjućeg stabla* SST (*Shortest Spaning Tree*) sastoji se u izdvajanju onog razapinjućeg stabla grafa G čiji je zbir dužina (težina) grana minimalan.

Za rešavanje problema najkraćeg razapinjućeg stabla efikasno se koristi Kraskalov algoritam [24].

Algoritam Kraskala

1° Inicijalizacija: početi sa grafom koji sačinjavaju samo čvorovi grafa G, tj. iz originalnog grafa ukloniti sve grane.

2° Sortirati sve grane L grafa G u neopadajući niz prema njihovim dužinama.

3° Dodavati grane inicijalnom grafu po sortiranom redosledu vodeći računa o tome da se ne formira kontura.

4° Ponavljati korak 3° sve dok broj dodatih grana ne bude $n - 1$. ■

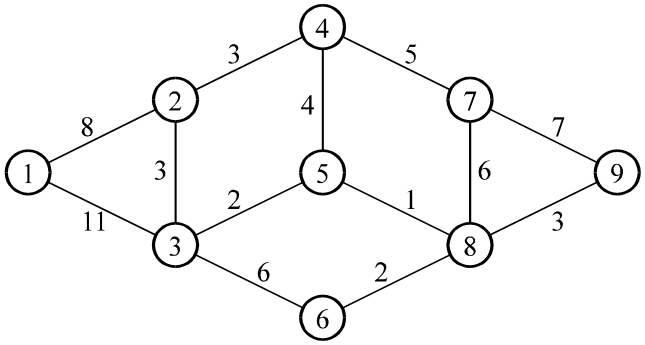
Drugi način

1º Uočiti bilo koju konturu grafa.

2º Iz uočene konture isključiti granu sa najvećom dužinom.

3º Ponavljati korake 1º i 2º sve dok ne ostane n - 1 grana, tj. dok ne bude više kontura. ■

Primer 1.9. Za graf prikazan na slika 1.15. odrediti najkraće razapinjuće stablo.



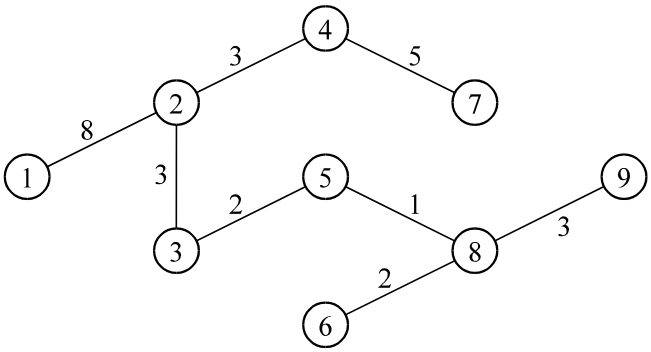
Slika 1.15.

Rešenje: I način:

Popisaćemo sve grane grafa i njihove dužine i sortirati ih u neopadajući niz:

(1, 2) : 8		(5, 8) : 1
(1, 3) : 11		(3, 5) : 2
(2, 3) : 3		(6, 8) : 2
(2, 4) : 3		(2, 3) : 3
(3, 5) : 2		(2,4) : 3
(3, 6) : 6		(8, 9) : 3
(4, 5) : 4	sortiramo ☞	(4, 5) : 4
(4, 7) : 5		(4, 7) : 5
(5, 8) : 1		(3, 6) : 6
(6, 8) : 2		(7, 8) : 6
(7, 8) : 6		(7, 9) : 7
(7, 9) : 7		(1, 2) : 8
(8, 9) : 3		(1, 3) : 11

Primenjujući Kraskalov algoritam, dobija se rešenje prikazano na slici 1.16. Grane koje nisu korišćene jer bi formirale konturu su: (4, 5),(3, 6), (7, 8), (7, 9) i (1, 3). Primenom ovog postupka dobijamo sledeće rešenje:

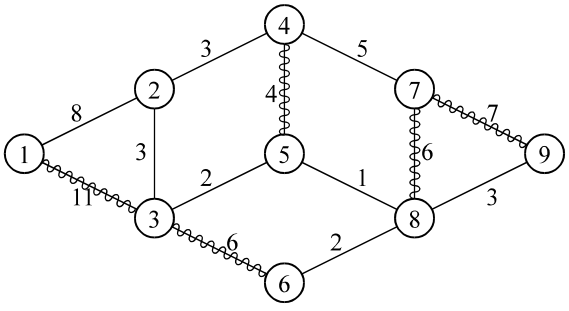


Slika 1.16.

II način:
Počećemo od zadatog grafa i uočiti npr. konturu (1, 2, 3, 1). Od grana koje sačinjavaju ovu konturu biramo onu sa najvećom dužinom i brišemo je. To je grana (1, 3). Sve uočene konture i izbrisane grane su date u sledećoj tabeli:

Kontura	Grana koja se briše
(1, 2, 3, 1)	(1, 3)
(2, 3, 5, 4, 2)	(4, 5)
(2, 3, 5, 8, 7, 4, 2)	(7, 8)
(2, 3, 5, 8, 9, 7, 4, 2)	(7, 9)
(3, 5, 8, 6, 3)	(3, 6)

Nakon ovog postupka dobili smo graf:



Slika 1.17.

Vidimo da je rezultujuće stablo istovetno sa ranije dobijenim. ■

1.6. Problem trgovačkog putnika (TSP)

Zadatak nalaženja najkraće *Hamiltonove konture* mreže naziva se *problem trgovačkog putnika* TSP (*Traveling Salesman Problem*). Hamiltonova kontura je put koji kroz sve čvorove grafa prolazi jednom i samo jednom i završava se u početnom čvoru:

$$hk = (i_1, i_2, i_3, \dots, i_{n-1}, i_n, i_1),\ i_p \neq i_k\ \forall\, p, k \in \{1, \dots, n\},\ p \neq k.$$

Nalaženje najkraće Hamiltonove konture liči na realni zadatak trgovačkog putnika koji planira obilazak više gradova, polazeći iz jednog grada i vraćajući se u njega.

Neki grafovi imaju jednu ili više Hamiltonovih kontura, a neki nijednu. Graf koji sadrži Hamiltonovu konturu, naziva se *Hamiltonov graf*.

Problem trgovačkog putnika se može formulisati kao zadatak celobrojnog programiranja na sledeći način:

$$(\min) f(x) = \sum_{(i,j)\in L} c_{ij} x_{ij}$$

p.o.

$$\sum_{\substack{i=1 \\ i\neq j}}^{n} x_{ij} = 1 \qquad j \in N \qquad (1)$$

$$\sum_{\substack{j=1 \\ j\neq i}}^{n} x_{ij} = 1\ , \qquad i \in N \qquad (2)$$

$$u_i - u_j + n\,x_{ij} \leq n - 1\ , \qquad i,j \in N,\ i \neq j,\ i,j \neq i_1 \qquad (3)$$

$$x_{ij} \in \{0,\ 1\}\ , \qquad i,j \in N$$

$$u_i \geq 0\ , \qquad i = 1,\dots,n,\ i \neq i_1$$

Ograničenja (1) i (2) obezbeđuju da se u svaki čvor uđe i izađe samo po jednom. Ograničenje (3) obezbeđuje da se rešenje ne sastoji od većeg broja odvojenih kontura. Promenljive x_{ij}, $i, j \in N$, su binarne promenljive. Kada je $x_{ij} = 0$, grana (i, j) ne pripada hk, a kada je $x_{ij} = 1$, grana (i, j) pripada hk.

Promenljive u_i, $i = 1, \ldots, n$ su pomoćne promenljive koje nemaju značaja kada se rešenje dobije. U slučaju da graf ne sadrži Hamiltonovu konturu, ovaj problem neće imati dopustivo rešenje.

Postoje dve osnovne grupe algoritama [12; 28] za rešavanje zadatka TSP:

1. *Egzaktni algoritmi* čija primena garantuje nalaženje optimalnog rešenja.
2. *Heuristički algoritmi* koji daju „dovoljno dobro" rešenje, tj. rešenje koje je blizu optimalnom, a često se dešava i da je upravo optimalno.

Broj mogućih Hamiltonovih kontura u potpuno povezanom grafu sa n čvorova je jednak $(n-1)!$. Računarski resursi potrebni za sistematsko pretraživanje dopustivog skupa, tj. skupa svih Hamiltonovih kontura, rastu eksponencijalno sa dimenzijom problema, odnosno sa brojem čvorova grafa. Za rešavanje zadatka TSP ne postoje egzaktni algoritmi polinomijalne složenosti. Algoritmi polinomijalne složenosti ne postoje ni za dobijanje odgovora na pitanje da li je neki graf Hamiltonov (osim u posebnim slučajevima) [12].

Od egzaktnih algoritama najviše se koriste različite implementacije metode grananja i ograničavanja, odnosno grananja i sečenja. Neke od njih su uspešno korišćene za rešavanje realnih problema na mrežama sa nekoliko hiljada čvorova. Ovde ćemo prikazati dva egzaktna algoritma koja se zasnivaju na sistematskom pretraživanju dopustivog skupa. To su *metoda vraćanja po tragu* (*backtracking*) za pronalaženje *hk* i *dinamičko programiranje* za određivanje najkraće *hk*.

Metoda vraćanja po tragu

Konačan skup dopustivih rešenja može se pretraživati na dva načina.

Potpuno pretraživanje ili *eksplicitna enumeracija* je metoda „grube sile" kojom se ispituje svako dopustivo rešenje. Ona se može svrstati u metode čija je primenljivost jako ograničena brojem dopustivih rešenja.

Posredno pretraživanje ili *implicitna enumeracija* su grupa metoda kojima je zajedničko da se ne ispituje svako od dopustivih rešenja ali se ipak sigurno pronalazi tačno rešenje. Pritom se koriste različita pravila za smanjivanje broja rešenja koja se ispituju.

Vraćanje po tragu pripada grupi metoda implicitne enumeracije.

Ideja vraćanja po tragu se često objašnjava primerom nalaženja puta kroz lavirint koji se može formulisati kao zadatak nalaženja puta između dva čvora u mreži. Polazi se od početnog čvora, a zatim se na bilo kom čvoru (raskrsnici lavirinta) bira grana kojom se nastavlja put. Pošto unapred nije poznata grana (pravac) koja vodi ka traženom putu (Hamiltonovoj konturi, izlazu iz lavirinta), jedno moguće rešenje je da se izabere bilo koja i o tome ostavi obeležje. Ako se ne pronađe izlaz, npr. došlo se u ćorsokak tj. do čvora iz koga se dalje ne može, treba se po tragu vratiti na prethodni čvor i pokušati s novom granom. Nekada

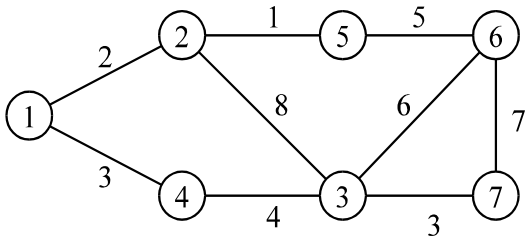
je potrebno vratiti se i nekoliko čvorova unazad, ali sistematskim, sukcesivnim vraćanjem po tragu i biranjem novih grana, na kraju se dolazi do rešenja. Ovo sve predstavlja seriju pokušaja i ispravljanja grešaka, ukoliko se one naprave. Ova metoda se zato zove metoda pokušaja i grešaka, a sam proces vraćanja unazad, metoda vraćanja po tragu.

Jedna od varijanti opšteg algoritma vraćanja po tragu, neznatno modifikovana za rešavanje postavljenog zadatka određivanja Hamiltonove konture, obuhvata sledeća pravila:

1. Na svakom koraku, odnosno kod svakog čvora, bolje je ići napred, ali kada to nije moguće, treba se vratiti po tragu.
2. Mogućnost koraka se proverava na osnovu toga da li postoji grana kojom iz tog čvora nije već pokušano nalaženje tražene konture.
3. Kretanje napred je uvek od prve slobodne grane koja polazi od posmatranog čvora (čvora na kome se nalazimo).
4. Redosled raspoloživih čvorova je utvrđen, npr. indeksom susednog čvora u rastućem poretku.
5. Grana koja je ranije korišćena se obeležava, jer algoritam treba da proba sve mogućnosti od određenog čvora ali ne sme da formira konturu.
6. Vraćanje po tragu se dešava kada više nema raspoloživih grana, odnosno kada nije moguće napraviti nijedan novi korak.
7. Vraćanje po tragu se sastoji od povratka na čvor iz koga se pošlo u prethodnom koraku i u pokušaju novog koraka sa sledećom raspoloživom granom.
8. Da bi se učinio korak nazad, potrebno je znati koji je bio prethodni korak. U tu svrhu se mora čuvati obeležje poslednje izabrane grane na svakom koraku. Ako se vraćanjem jednog koraka nazad dolazi u situaciju u kojoj su prethodno iskorišćene sve mogućnosti, trebalo bi napraviti još jedan korak nazad, i tako dalje, dok se ne otvori mogućnost novog koraka. ■

Iako se metoda vraćanja po tragu može modifikovati za rešavanje optimizacionih zadataka [40; 42], njena primena se u ovom kontekstu preporučuje samo za ispitivanje da li je graf Hamiltonov ili za pronalaženje optimalnog rešenja kada broj Hamiltonovih kontura nije veliki.

Primer 1.10. Odrediti Hamiltonovu konturu grafa prikazanog na slici 1.18.



Slika 1.18.

Rešenje: Počinje se od čvora 1 i uvek prvo pokušava ići na susedni čvor sa manjim indeksom. Tako se formira put:
(1, 2, 3, 4).
Iz čvora 4 se ne može dalje osim na čvor 1, čime se formira kontura koja nije Hamiltonova. Vraća se na čvor 3 i pravi put sa sledećom raspoloživom granom
(1, 2, 3, 6, 5).
Iz čvora 5 se dalje ne može i treba se vratiti. Postupak se dalje odvija na sledeći način. Povratak na čvor 6,
(1, 2, 3, 6, 7).
Povratak na čvor 6, a onda dalje na čvor 3
(1, 2, 3, 7, 6, 5).
Povratak na čvor 6, a onda na 7, zatim na 3, pa na 2.
(1, 2, 5, 6, 3, 4).
Povratak na čvor 3, a onda na 6
(1, 2, 5, 6, 7, 3, 4, 1).
Na ovaj način pronađena je Hamiltonova kontura dužine 25.
Dalji redosled puteva koji bi bili pretraživani je:
(1, 4, 3, 2, 5, 6, 7),
(1, 4, 3, 6, 5, 2),
(1, 4, 3, 6, 7),
(1, 4, 3, 7, 6, 5, 2, 1).
Ovo je, u stvari, ranije dobijena Hamiltonova kontura, ali sa obrnutim redosledom obilaska čvorova.

Prema tome, može se zaključiti da u mreži postoji samo jedna Hamiltonova kontura dužine 25. ■

Primena dinamičkog programiranja na rešavanje TSP

Data je mreža $G = (N, L)$ sa dužinama grana c_{ij}, za svako $(i, j) \in L$, dok su $c_{ij} = \infty$, za $(i, j) \notin L$. Usvojimo da je čvor 1 polazni i završni čvor. Definisaćemo skup $Q \subseteq N \setminus \{1\}$, kao bilo koji podskup čvorova grafa koji ne sadrži čvor 1. Označićemo sa $f(Q, j), j \notin Q$ dužinu najkraćeg puta koji polazi od čvora 1, prolazi kroz sve čvorove iz skupa Q i završava u čvoru j. Po definiciji, najkraća Hamiltonova kontura imaće dužinu $f(N \setminus \{1\}, 1)$ tj. Biće jednaka dužini najkraćeg puta koji polazi od prvog čvora, prolazi kroz sve čvorove osim prvog i završava se u prvom čvoru.

Na osnovu ovoga se može definisati sledeća rekurentna formula dinamičkog programiranja [35]:

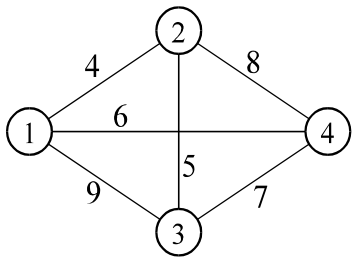
$$f(Q, j) = \min_{i \in Q} \{ f(Q \setminus \{i\}, i) + c_{ij} \}, \quad \forall\, Q \subseteq N \setminus \{1\}, \forall\, j \in N \setminus Q$$

$f(\varnothing, j) = c_{1j}$.

Optimalno rešenje dobija se za

$$f(N \setminus \{1\}, 1) = \min_{i \in N \setminus \{1\}} \{ f(N \setminus \{i, 1\}, i) + c_{i1} \}$$

Primer 1.11. Odrediti najkraću Hamiltonovu konturu za zadatu mrežu.



Slika 1.19.

Rešenje: Pošto je nulta etapa trivijalna, počećemo od prve:

$f(\{2\}, 3) = f(\varnothing, 2) + c_{23} = \underline{c}_{12} + \underline{c}_{23} = \underline{4 + 5} = 9$
$f(\{3\}, 2) = f(\varnothing, 3) + c_{32} = c_{13} + c_{32} = 9 + 5 = 14$
$f(\{2\}, 4) = f(\varnothing, 2) + c_{24} = c_{12} + c_{24} = 4 + 8 = 12$
$f(\{4\}, 2) = f(\varnothing, 4) + c_{42} = c_{14} + c_{42} = 6 + 8 = 14$
$f(\{3\}, 4) = f(\varnothing, 3) + c_{34} = c_{13} + c_{34} = 9 + 7 = 16$
$f(\{4\}, 3) = f(\varnothing, 4) + c_{43} = \underline{c}_{14} + \underline{c}_{43} = \underline{6 + 7} = 13$

Druga etapa:

$$f(\{2,3\},4) = \min_{i\in\{2,3\}} \begin{Bmatrix} f(\{3\},2) + c_{24} \\ f(\{2\},3) + \underline{c_{34}} \end{Bmatrix} = \min \begin{Bmatrix} 14 + 8 \\ \underline{9 + 7} \end{Bmatrix} = 16$$

$$f(\{2,4\},3) = \min_{i\in\{2,4\}} \begin{Bmatrix} f(\{4\},2) + c_{23} \\ f(\{2\},4) + c_{43} \end{Bmatrix} = \min \begin{Bmatrix} 14 + 5 \\ 12 + 7 \end{Bmatrix} = 19$$

$$f(\{3,4\},2) = \min_{i\in\{3,4\}} \begin{Bmatrix} f(\{4\},3) + \underline{c_{32}} \\ f(\{3\},4) + c_{42} \end{Bmatrix} = \min \begin{Bmatrix} \underline{13 + 5} \\ 16 + 8 \end{Bmatrix} = 18$$

I na kraju treća, završna etapa u kojoj dobijamo konačno rešenje:

$$f^* = f(\{2,3,4\},1) = \min_{i\in\{2,3,4\}} \left\{ f(N\setminus\{1,i\}, i) + c_{i1} \right\} =$$

$$= \min \begin{Bmatrix} f(\{3,4\},2) + \underline{c_{21}} \\ f(\{2,4\},3) + c_{31} \\ f(\{2,3\},4) + \underline{c_{41}} \end{Bmatrix} = \min \begin{Bmatrix} \underline{18 + 4} \\ 19 + 9 \\ \underline{16 + 6} \end{Bmatrix} = \underline{22}$$

Najkraća Hamiltonova kontura zadate mreže ima dužinu 22. Redosled čvorova se dobija prateći unazad dobijene minimalne vrednosti (koje su podvučene). U zadatku imamo dva najkraća puta i to: (1, 2, 3, 4, 1) i (1, 4, 3, 2, 1).

Napomena: U simetričnim mrežama se uvek dobija paran broj najkraćih Hamiltonovih kontura. Razlog tome je jasan: jednu konturu je moguće obići u oba smera. ■

Osim problema nalaženja najkraće Hamiltonove konture, isti postupak se može primeniti i na nalaženje najkraćeg Hamiltonovog puta kroz mrežu. *Hamiltonov put* se definiše kao put koji polazi od početnog, prolazi kroz ostale čvorove samo jedanput, i završava se u neki završni čvor. Postoje različite varijante ovog problema, npr. varijanta kod koje su zadati početni i završni čvor, ili samo jedan od njih. One se mogu rešiti opisanim postupkom dinamičkog programiranja uz male modifikacije.

Heuristički algoritmi za rešavanje TSP

Heurističkim algoritmima se nastoji relativno jednostavno i računarski efikasno pronaći rešenje, za koje se može tvrditi da je „dovoljno dobro“. Da je ono stvarno takvo, dokazuje se na primerima, za koje su egzaktnim metodama nađena optimalna rešenja ili na primerima za koje se može dati procena da li je rešenje dobro.

U velikom broju heurističkih algoritama koristi se pristup koji se sastoji od sledeća dva koraka:

1. Nađe se početno rešenje.
2. Postojeće rešenje se poboljšava pomoću određenog broja pravila.

Ako je potrebno, koraci 1. i 2. se ponavljaju više puta. Pravila koja se koriste za poboljšanje postojećeg rešenja formiraju se na osnovu opštih znanja iz optimizacije, znanja o konkretnom problemu i njegovim osobinama, iskustva sa sličnim zadacima i sl.

Nalaženje početnog rešenja

Postoji više algoritama za nalaženje početnog rešenja TSP kada je graf potpun. Najjednostavniji je slučajno generisanje konture, tj. permutacije od n elemenata. Međutim, pored zahteva da se početno rešenje generiše efikasno, obično se traži i da ono bude relativno dobro. Jedan od algoritama koji zadovoljava ove zahteve je *algoritam najbližeg suseda*: od početnog čvora ide se ka najbližem susedu (minimalno $c_{1j}, j = 2, \ldots, n$), a od njega dalje nastavlja po istom principu dok se ne obiđu svi čvorovi.

Koriste se i složenije metode u kojima se relaksiraju ili izostave neka od ograničenja (1) - (3). Na primer, može se u prvom koraku naći minimalno razapinjuće stablo. Zatim od stabla treba generisati konturu iterativnom eliminacijom i zamenom grana koje čine da je stepen nekog čvora (broj čvorova susednih tom čvoru) veći od dva. Dodavanje novih grana vrši se sa ciljem da se obezbedi povezanost i formira kontura.

Još jedan mogući postupak za dobijanje početnog rešenja je izostavljanje ograničenja (3) za eliminaciju odvojenih kontura i rešavanje zadatka linearnog celobrojnog programiranja bez ovih ograničenja. Originalni zadatak TSP se, u stvari, zamenjuje jednostavnijim zadatkom linearnog celobrojnog programiranja koji je poznat pod nazivom problem asignacije. Ako se kao rešenje dobije jedna kontura, onda je to Hamiltonova kontura i ona predstavlja optimalno rešenje originalnog zadatka. Ako se kao rešenje dobije nekoliko odvojenih kontura, što je češći slučaj, onda od tih kontura treba nekim pogodnim pravilima generisati Hamiltonovu konturu. Minimalna Hamiltonova kontura ne može biti kraća od optimalne vrednosti funkcije cilja dobijene rešavanjem problema signacije.

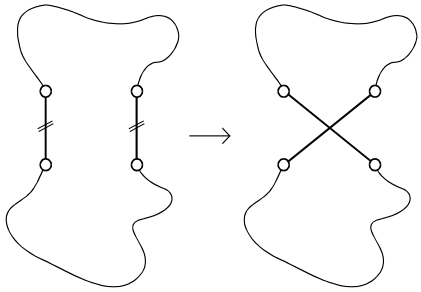
Metode za poboljšanje rešenja

Postoji veliki broj heurističkih metoda za poboljšanje postojećeg rešenja TSP koje se, prema principima na kojima se zasnivaju, mogu podeliti na: a) k-optimalne heuristike, b) tabu pretraživanje, c) simulirano kaljenje i d) genetski algoritmi. Svima je zajednička osobina da na neki način pretražuju okolinu postojećeg rešenja. Poslednje tri grupe su tzv. savremene metaheuristike na

kojima se danas veoma mnogo radi [12]. Ovde ćemo prikazati algoritam [22; 31] koji pripada metodama grupe a). U njima se koristi sledeća osnovna ideja pretraživanja okoline postojećeg rešenja.

Neka je postojeće rešenje nekog optimizacionog zadatka predstavljeno vektorom x dimenzije n, tj. $x = (x_1, \ldots, x_n)$. Treba ispitati da li se promenom k komponenata ovog vektora, $k < n$, u okvirima dopustivog skupa može dobiti bolje rešenje od postojećeg. Ako je odgovor negativan, postojeće rešenje je lokalni minimum i naziva se k-optimalno rešenje. U suprotnom, potrebno je postojeće rešenje zameniti novim i nastaviti lokalno pretraživanje njegove okoline. Prema tome, k-opt algoritam za rešavanje TSP ima sledeća dva osnovna koraka:

1. Formira se početna kontura;
2. Ukloni se k nesusednih grana iz konture i tako dobije k nepovezanih puteva. Tako dobijeni putevi se privremeno povezuju na sve moguće načine kojima se ostvaruje kontura. Ako neko povezivanje daje kraću konturu, ta kontura zamenjuje početnu i ovaj korak se ponavlja za svaku kombinaciju k grana. Postupak se zaustavlja kada se nijednom kombinacijom uklanjanja k grana iz konture ne može postići poboljšanje.



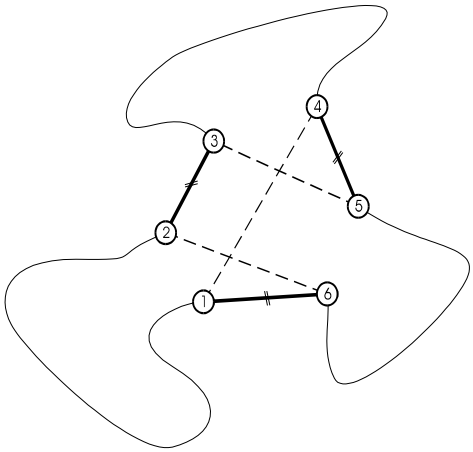
Slika 1.20.

Za rešavanje TSP u praksi se primenjuju 2-opt i 3-opt algoritmi. Eksperimenti pokazuju da se daljim povećanjem k retko dobija bolje rešenje.

U slučaju primene 2-opt algoritma ponovno povezivanje konture moguće je samo na jedan način (slika 1.20). U slučaju 3-opt algoritma, ako se uklone nesusedne grane (2, 3), (4, 5) i (1, 6) kao što je prikazano na slici 1.21, moguće je na sedam novih načina ponovo uspostaviti konturu i to:

1. (1, 3), (2, 5), (4, 6);
2. (1, 3), (2, 6), (4, 5);
3. (1, 4), (2, 5), (3, 6);
4. (1, 4), (2, 6), (3, 5);
5. (1, 5), (2, 3), (4, 6);
6. (1, 5), (2, 4), (3, 6);
7. (1, 6), (2, 4), (3, 5).

Jedan od nabrojanih načina skiciran na slici 1.21.

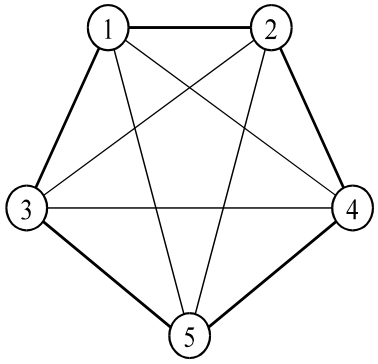


Slika 1.21.

Primer 1.12. Naći što kraću Hamiltonovu konturu za neorijentisan graf dat matricom:

i *j*:	1	2	3	4	5
1	0	4	10	7	6
2	4	0	13	11	15
3	10	13	0	12	8
4	7	11	12	0	5
5	6	15	8	5	0

Rešenje: Pošto se traži „što kraća“ Hamiltonova kontura, možemo da koristimo neki heuristički algoritam. Za nalaženje početnog rešenja koristićemo algoritam najbližeg suseda.
Usvojićemo da nam je početni čvor $s = 1$. Najbliži čvoru s je čvor kome odgovara najmanji element u prvom redu date matrice; to je čvor 2. Najbliži čvoru 2 je čvor 4 jer smo čvor 1 već uključili u put. Najbliži sused čvoru 4 je 5, a ovome čvor 3. Na taj način je određen Hamiltonov put. Spajanjem prvog i poslednjeg čvora (1 i 3) dobija se Hamiltonova kontura (1, 2, 4, 5, 3, 1) (slika 1.22.).



Slika 1.22.

Pokušaćemo da poboljšamo početno rešenje primenom 2-opt algoritma. Privremeno ćemo izbacivati po dve grane i unakrsno spajati čvorove. Tako ćemo dobijati alternativne Hamiltonove konture:

Rb	Isključene grane	Generisane konture	Dužina
0	Početno rešenje	(1, 2, 4, 5, 3, 1)	4+11+5+8+10=38
1	(1, 2), (4, 5)	(1, 4, 2, 5, 3, 1)	7+11+15+8+10=51
2	(1, 2), (3, 5)	(1, 5, 4, 2, 3, 1)	6+5+11+13+10=42
3	(1, 3), (2, 4)	(1, 4, 5, 3, 2, 1)	7+5+8+13+4=37
4	(1, 3), (4, 5)	(1, 5, 3, 4, 2, 1)	6+8+12+11+4=41
5	(2, 4), (3, 5)	(1, 2, 5, 4, 3, 1)	4+15+5+12+10=46

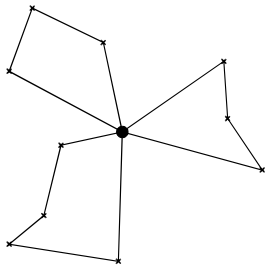
Vidimo da je najbolje rešenje 3, pa ćemo njega usvojiti kao početno rešenje za sledeću iteraciju.

Rb	Isključene grane	Generisane konture	Dužina
0	početno rešenje	(1, 4, 5, 3, 2, 1)	7+5+8+13+4=37
1	(1, 2), (4, 5)	(1, 5, 3, 2, 4, 1)	6+8+13+11+7=45
2	(1, 2), (3, 5)	(1, 3, 2, 5, 4, 1)	10+13+15+5+7=50
3	(1, 4), (2, 3)	(1, 3, 5, 4, 2, 1)	10+8+5+11+4=38
4	(1, 4), (3, 5)	(1, 5, 4, 3, 2, 1)	6+5+12+13+4=40
5	(2, 3), (4, 5)	(1, 2, 5, 3, 4, 1)	4+15+8+12+7=46

Pošto nismo uspeli da poboljšamo početno rešenje, ono se usvaja kao dovoljno dobro rešenje postavljenog zadatka. ■

1.7. Problem rutiranja vozila (VRP)

Zadatak trgovačkog putnika može da se uopšti na zadatak sa *m* trgovačkih putnika. Pritom se postavljaju dodatna ograničenja kao što su: svaki trgovački putnik treba da obiđe tačan broj gradova; broj putnika je ograničen; ograničen je broj gradova koje trgovački putnik može da obiđe itd. Posebnu grupu zadataka karakteriše zahtev da svi trgovački putnici krenu iz istog grada i vrate se u njega, a da pri tome svaki od njih obiđe ograničen broj gradova. Ova grupa zadataka nosi zajednički naziv *problem rutiranja vozila* VRP (*Vehicle Routing Problem*). Za ovaj problem se u literaturi mogu naći i nazivi planiranje maršrute vozila i raspoređivanje (upućivanje) vozila [12]. Analogija sa raspoređivanjem vozila je očigledna: iz jednog (centralnog) skladišta treba transportovati robu do (potrošačkih) centara koristeći više vozila (kamiona) ili jedno vozilo više puta. Dodatno se postavlja ograničenje na kapacitet vozila.



Slika 1.23.

Na slici 1.23. centralno skladište je obeleženo tačkom, a mesta isporuke sa „×“, dok linije predstavljaju putanje vozila kojim se roba razvozi.

Razmatraćemo osnovni zadatak raspoređivanja vozila u kome je potrebno vozilima istih kapaciteta (ili jednim vozilom sa više polazaka) snabdeti određenom količinom robe mesta izporuke. Koristićemo sledeće oznake:
$i = 1$ - oznaka čvora mreže koji predstavlja magacin;
$i = 2, \ldots, n$ - oznake čvorova u koje treba isporučiti robu (potrošači);
c_{ij} - rastojanje ili troškovi transporta između svaka dva čvora mreže;

$$x_{ij} = \begin{cases} 0 & - \text{ vozilo ide granom } i-j \\ 1 & - \text{ vozilo ne ide granom } i-j \end{cases}$$

q_i - potreba i-tog potrošača (težina i-tog čvora);
Q - kapacitet vozila.

Ponovo ćemo razmatrati mrežu datu grafom $G = (N, L)$ i matricom rastojanja $C = (c_{ij})$. Indeks čvora $i = 1$ rezervisan je za skladište. Zadatak VRP se sastoji u određivanju putanja (ruta) vozila tako da se minimizira njihova ukupna dužina (troškovi), a koje su takve da:

1. Svaki grad u $N\backslash\{1\}$ se posećuje tačno jedanput samo jednim vozilom.
2. Sve rute počinju i završavaju u skladištu.
3. Zadovoljena su neka dodatna ograničenja.

Najčešća dodatna ograničenja su [7; 8; 29]:

1. Ograničenja kapaciteta: svakom čvoru (gradu) $i > 1$ pridružuju se nenegativna težina q_i, a zbir težina na bilo kojoj ruti ne sme da bude veći od kapaciteta vozila Q.
2. Broj gradova na proizvoljnoj ruti je ograničen sa P (u ovom slučaju je $q_i = 1$ za $i > 1$ i $Q = P$).
3. Ograničenje na ukupnu dužinu rute (vreme putovanja): ukupna dužina rute je ograničena sa Λ. Ova dužina se dobija na osnovu vremena putovanja između gradova i vremena zadržavanja u gradovima.

4. Vremenski prozori: grad i treba da bude posećen u vremenskom intervalu $[a_i, b_i]$, a postoje i zadržavanja (čekanja) u gradu i.
5. Odnosi prethođenja između parova čvorova; grad i ne može biti posećen pre grada j.

Kao i mnogi drugi zadaci optimizacije na mrežama, VRP može da se formuliše kao zadatak linearnog celobrojnog programiranja. Postoji nekoliko takvih standarnih formulacija. Svaka od njih je pogodna za primenu određene optimizacione metode.

Jedan postupak je relaksacija problema VRP na problem m trgovačkih putnika [29]. Polazi se od pretpostavke da je poznata gornja granica m_U za m. Problem m trgovačkih putnika se najpre transformiše u problem jednog trgovačkog putnika na sledeći način:

1. Poveća se broj čvorova uvođenjem m_U - 1 veštačkih skladišta; neka je $n' = n + m_U - 1$, $N' = \{1, \dots, n'\}$ i $L' = L \cup \{(i,j) \mid j \in N', i \neq j, i \in N' \backslash N \vee j \in N' \backslash N\}$;
2. Definiše se proširena matrica rastojanja $C' = (c'_{ij})$ pridružena N'

$$c'_{ij} = \begin{cases} c_{ij}, & i, j \in N \\ c_{i1}, & i \in N \backslash \{1\}, j \in N' \backslash N \\ c_{1j}, & i \in N' \backslash N, j \in N \backslash \{1\} \\ \gamma, & i, j \in (N' \backslash N) \cup \{1\} \end{cases}$$

gde vrednosti za γ zavise od varijante razmatranog zadataka i iznose:
γ = ∞ - kada se traži minimalna dužina za m_U vozila;
γ = 0 - kada se traži minimalna dužina za najviše m_U vozila;
γ = -∞ - kada se traži minimalna dužinma za najmanji broj vozila.

VRP se sada može formulisati na sledeći način:
Neka je x_{ij} ($i \neq j$) binarna promenljiva jednaka 1 ako i samo ako se grana (i,j) iz L' pojavljuje u optimalnom rešenju. Zadatak je:

$$(\min) \sum_{i \neq j} c_{ij} x_{ij} \tag{1}$$

p.o.

$$\sum_{j=1}^{n'} x_{ij} = 1, \quad i = 1, \dots, n' \tag{2}$$

$$\sum_{i=1}^{n'} x_{ij} = 1, \quad j = 1, \dots, n' \tag{3}$$

$$\sum_{i,j\in S} \leq |S| - v(S)\ ,\quad S \subset N'\backslash\{1\};\ |S| \geq 2 \tag{4}$$

$$x_{ij} \in \{0, 1\}\ ,\ i,j = 1, \ldots, n',\ i \neq j \tag{5}$$

U ovoj formulaciji (1), (2), (3) i (5) definišu modifikovani problem dodeljivanja (asignacije), u kome je na glavnoj dijagonali zabranjeno dodeljivanje. Ograničenja (4) su ograničenja za eliminaciju podkontura; $v(S)$ je pogodno određena donja granica broja vozila potrebnih da se u optimalnom rešenju posete svi čvorovi u S. Ova ograničenja se dobijaju sledećim razmatranjem:

Za bilo koje $S \subset N'\backslash\{1\}$, $|S| \geq 2$, $\overline{S} = N'\backslash N$ mora biti

$$\sum_{i\in S}\sum_{j\in\overline{S}} x_{ij} \geq v(S).$$

Pored toga, uvek važi sledeća jednakost:

$$|S| = \sum_{i,j\in S} x_{ij} + \sum_{i\in S}\sum_{j\in\overline{S}} x_{ij}.$$

Vrednost $v(S)$ zavisi od tipa problema VRP koji se razmatra. U slučaju da su data ograničenja kapaciteta treba uzeti:

$$v(S) = \left\lceil \frac{\sum_{i\in S} q_i}{Q} \right\rceil$$

gde $\lceil x \rceil$ označavazaokruživanje broja na veći ceo broj.

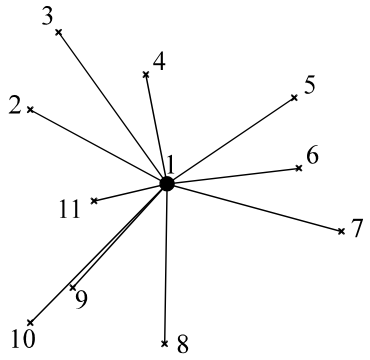
Rešavanje zadatka VRP primenom algoritma ušteda

Ovaj algoritam [10] je jedan od najviše primenjivanih heurističkih pristupa rešavanju zadatka VRP. Algoritam polazi od početnog rešenja u kojem ima n-1 ruta koje se formiraju tako što se po jedno vozilo upućuje iz skladišta do jednog potrošača i vraća u skladište (slika 1.24.). Ovo rešenje se matematički može zapisati:

$x_{1i} = 1,\ \forall\, i \in N \setminus \{1\},$

$x_{ij} = 0,\ \forall\, i \neq 1 \wedge j \neq 1$

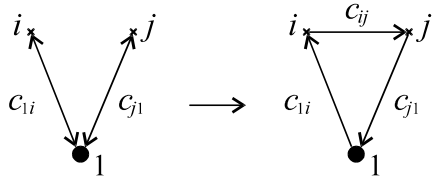
Zatim se interaktivno, iz koraka u korak spajaju po dve putanje koje donose najveću uštedu, a zadovoljavaju ograničenja zadatka.



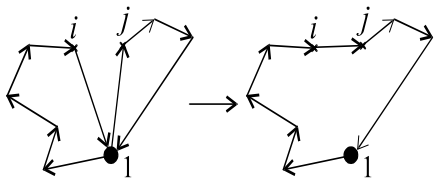
Slika 1.24.

Pojam uštede objasnićemo na sledećem jednostavnom primeru prikazanom na slici 1.25. U slučaju pod (a), vozilo kreće iz skladišta do čvora i, zatim se vraća u skladište, ide do čvora j i ponovo vraća. Pri tom pravi ukupne troškove: $C_a = c_{1i} + c_{i1} + c_{1j} + c_{j1}$. U slučaju pod (b), vozilo kreće iz skladišta, ide do čvora i, zatim iz i ide do čvora j i potom vraća u skladište. Pri tom pravi ukupne troškove: $C_b = c_{1i} + c_{ij} + c_{j1}$. Ako se umesto plana pod (a) prikazanog na slici 1.25. primeni plan (b) ostvariće se ušteda S_{ij}.

$$S_{ij} = C_a - C_b = c_{1i} + c_{i1} + c_{1j} + c_{j1} - c_{1i} - c_{ij} - c_{j1} = c_{i1} + c_{1j} - c_{ij}$$



(a) (b)



(c) (d)

Slika 1.25.

Pojam uštede se može koristiti i pri spajanju dve rute u jednu (slike 1.25. pod (c) i (d)). Treba primetiti da ušteda S_{ij} može biti veća, manja ili jednaka nuli. Ima smisla spajati dve rute samo ako je ušteda veća od nule i ako su zadovoljena

ostala ograničenja zadatka (kapacitet vozila).

Algoritam

1° Kreirati n - 1 ruta (1, i, 1), i = 2, ... , n, tj. odrediti početno rešenje na sledeći način:

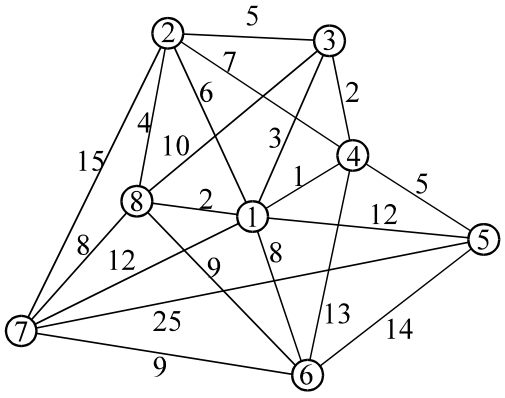
$$x_{ij} = \begin{cases} 1 \text{ za } (i = 1,\ j \in N \setminus \{1\}) \vee (i \in N \setminus \{1\},\ j = 1) \\ 0 \text{ za } \forall\ i \neq 1 \wedge j \neq 1 \end{cases}$$

2° Izračunati moguće uštede $S_{ij} = c_{i1} - c_{ij} + c_{1j}$, $\forall\ (i, j) \in L$.

3° Sve pozitivne uštede sortirati u nerastući niz.

4° Idući od prve ka poslednjoj uštedi iz niza, pokušati napraviti moguće uštede na sledeći način: posmatrati dve rute koje sadrže grane (i, 1) i (1, j). Privremeno spojiti ove dve rute uvodeći granu (i, j) i brišući grane (i, 1) i (1, j). Ako je dobijena ruta dopustiva (zadovoljava ograničenje kapaciteta), iskoristiti granu (i, j), formirati rutu i pokušati sa sledećom mogućom uštedom iz niza. Ako privremena ruta nije dopustiva, odbaciti granu (i, j), ne formirati novu rutu, skinuti S_{ij} iz niza i preći na sledeću potencijalnu uštedu. ■

Primer 1.13. Rešiti zadatak VRP sa slike 1.26. ako su date tražnje potrošača: q_2 = 40, q_3 = 20, q_4 = 25, q_5 = 70, q_6 = 45, q_7 = 50, q_8 = 100 i kapacitet vozila Q = 200.



Slika 1.26.

Rešenje: Početno rešenje ima oblik:

$$X = (x_{ij}) = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Moguće uštede su:

$S_{23} = 4$ $S_{24} = 0$ $S_{27} = -3$ $S_{28} = 4$

$S_{34} = 2$ $S_{38} = -5$ $S_{45} = 8$ $S_{46} = -4$

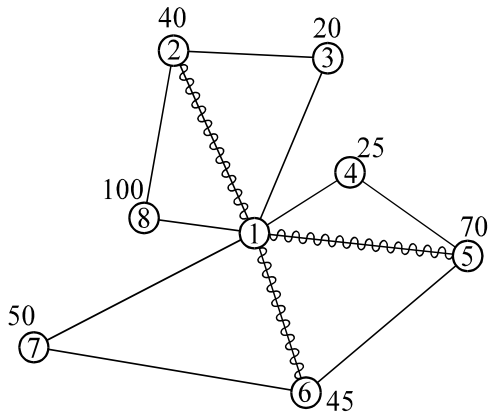
$S_{56} = 6$ $S_{57} = -1$ $S_{67} = 11$ $S_{68} = 1$

$S_{78} = 6$

Kada sve pozitivne uštede sortiramo u nerastući niz, dobijamo sledeću listu:

$S_{67} = 11$
$S_{45} = 8$
$S_{56} = 6$
$S_{78} = 6$
$S_{23} = 4$
$S_{28} = 4$
$S_{34} = 2$
$S_{68} = 1$

Svakoj od ovih ušteda odgovara spajanje ruta koje je moguće uraditi samo ako su zadovoljena ograničenja kapaciteta.



Slika 1.27.

1. (6-7):
   $q_6 + q_7 = 45 + 50 = 95 < 200 \Rightarrow x_{61} = 0, x_{17} = 0, x_{67} = 1$
2. (4-5):
   $q_4 + q_5 = 70 + 25 = 95 < 200 \Rightarrow x_{41} = 0, x_{15} = 0, x_{45} = 1$
3. (5, 6):
   ovom vezom spajaju se ruta (1, 4, 5) i ruta (1, 6, 7):
   $q_4 + q_5 + q_6 + q_7 = 190 < 200 \Rightarrow x_{51} = 0, x_{16} = 0, x_{56} = 1$
   Pošto su i x_{15} i x_{51} jednaki nuli, granu (1, 5) možemo da uklonimo (precrtamo na mreži). Ovo važi i za granu (1, 6).
4. (7, 8):
   $190 + 100 = 390 > 200 \Rightarrow$ nedopustivo povezivanje;, ušteda S_{78} se ne može iskoristiti.
5. (2, 3):
   $40 + 20 = 60 < 200 \Rightarrow x_{21} = 0, x_{13} = 0, x_{23} = 1$
6. (2, 8):
   $60 + 100 = 160 < 200 \Rightarrow x_{81} = 0, x_{12} = 0, x_{82} = 1$
   Kao i u trećem koraku, sada možemo da uklonimo (1, 2).
7. (3, 4):
   $160 + 190 = 350 > 200 \Rightarrow$ nedopustivo.
8. (6, 8):
   Pošto je $x_{16} = x_{61} = 0$, ovu uštedu ne možemo da ostvarimo.

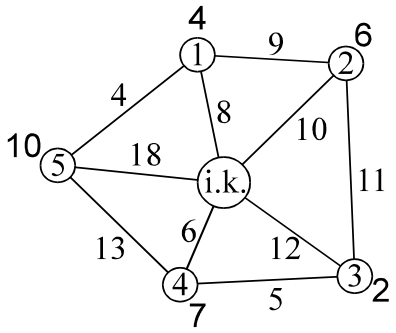
Pošto smo dobili konačno rešenje (slika 1.27.) potrebno je da

izračunamo dužinu puta koju vozilo treba da pređe:

$$f^* = 2\,c_{12} + 2\,c_{13} + 2\,c_{14} + 2\,c_{15} + 2\,c_{16} + 2\,c_{17} + 2\,c_{18} - S_{67} - S_{45} - S_{56} - S_{23} - S_{28}$$

$$f^* = 88 - 33 = 55$$ ■

Primer 1.14. Raznosač novina mora da isporuči dnevnu štampu od izdavačke kuće do 5 trafika. Prevozno sredstvo kojim raspolaže omogućava mu da odjednom ponese najviše 12 paketa novina. Na slici 1.28. je data skica rasporeda izdavačke kuće i trafika (1-5), njihova udaljenost i količina novina koju potražuje svaka od njih. Primenom algoritma ušteda odrediti optimalnu rutu raznosača.

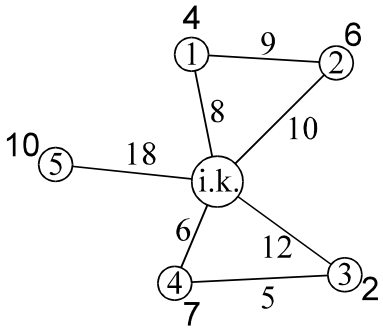


Slika 1.28.

Rešenje:

$S_{12} = 9$		$S_{51} = 22$	$10 + 4 > 12$ => nedopustivo
$S_{23} = 11$	sort.	$S_{34} = 13$	$2 + 7 < 12$ => OK
$S_{34} = 13$	=>	$S_{45} = 11$	$10 + 7 > 12$ => nedopustivo
$S_{45} = 11$		$S_{23} = 11$	$6 + 9 > 12$ => nedopustivo
$S_{51} = 22$		$S_{12} = 9$	$4 + 6 < 12$ => OK

Konačno rešenje:

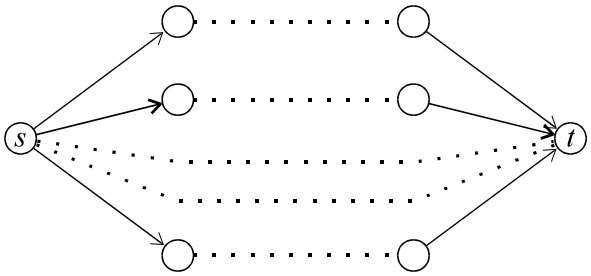


Slika 1.29.

Ukupan iznos uštede $S = S_{34} + S_{12} = 22$. ■

1.8. Maksimalni protok kroz mrežu

Posmatramo mrežu koja je definisana grafom $G = (N, L)$, *kapacitetima* (maksimalno mogućim protocima) grana c_{ij}, $(i, j) \in L$ i čvorovima s i t takvim da je $\Gamma^{-1}(s) = \Gamma(t) = \varnothing$ (slika 1.30). Čvor s je početni čvor ili izvor, a t krajnji čvor ili ušće. Potrebno je odrediti *kapacitet mreže*, odnosno *maksimalni protok* koji se može ostvariti kroz mrežu od izvora do ušća. Zadaci ovog tipa sreću se u drumskom i rečnom saobraćaju, u telekomunikacijama, u mrežama gasovoda i naftovoda, pri projektovanju proizvodnih linija itd.



Slika 1.30.

Koristićemo sledeće oznake:
s - početni čvor ili izvor
t - krajnji čvor ili ušće
f_{ij} - protok kroz granu (i, j)
c_{ij} - kapacitet grane (i, j), važi da je $0 \le f_{ij} \le c_{ij} \ \forall (i, j) \in L$
v - protok kroz mrežu.

Ideja za rešavanje postavljenog zadatka počiva na pojmu *preseka mreže*. Presek mreže je skup grana čijim bi se ukljanjanjem presekli svi putevi između početnog i kranjeg čvora. Svakom preseku odgovara *kapacitet* ili *propusna sposobnost preseka* koji je jednak zbiru kapaciteta grana koje mu pripadaju. Od svih preseka, onaj koji ima najmanji kapacitet naziva se *minimalni presek*. Jasno je, da je maksimalni protok kroz mrežu jednak kapacitetu minimalnog preseka. Može se reći da je minimalni presek usko grlo mreže.

Postoji više algoritamskih realizacija ove ideje. Algoritam koji sledi je iterativni postupak u kome se u svakoj iteraciji pronalaze, ako postoje, novi putevi kojima je moguće ostvariti dodatni protok [18].

Najpre se nađe bilo koji put između s i t i izračuna njegov kapacitet. On je jednak kapacitetu one grane na putu koja ima najmanji kapacitet. Zapamti se ovaj kapacitet koji se naziva *dodatni protok* i za njegov iznos se smanje kapaciteti grana koje pripadaju utvrđenom putu. Kapacitet bar jedne grane tako će postati jednak nuli. Postupak se nastavlja sve dotle, dok možemo naći neki put od s do t, sa protokom većim od nule. Maksimalni protok jednak je zbiru dodatnih protoka, a grane čiji su kapaciteti na kraju jednaki nuli predstavljaju minimalni presek.

U implementaciji algoritma koristi se koncept obeležavanja čvorova. Obeležje čvora j sastoji se od dva dela: $[i_j, F_j]$. Prvi deo označava čvor iz kojeg dolazi dodatni tok, a drugi njegovu vrednost. Na početku svake iteracije, svi čvorovi osim čvora s su neobeleženi.

Skup obeleženih čvorova označićemo sa M a element ovog skupa sa i. U skupu neobeleženih čvorova $N \setminus M$ treba pronaći čvor j koji sledi neki obeleženi čvor i i do koga bi granom (i, j) bilo moguće ostvariti dodatni protok $c_{ij} - f_{ij} > 0$. Obeležimo sa Q_i skup svih takvih čvorova, tj:

$$Q_i = \{ j \mid j \in \Gamma(i) \cap N \setminus M, c_{ij} - f_{ij} > 0 \}.$$

Ako postoji $i \in M$ takav da je $Q_i \neq \varnothing$, treba nastaviti sa obeležavanjem čvorova dok se ne stigne do čvora t. Ako je $Q_i = \varnothing$, za svako $i \in M$, nije moguće dalje ostvarivati dodatne protoke, odnosno, pronađen je minimalni presek koji sačinjavaju grane za koje je $c_{ij} - f_{ij}$.

Određivanje ekstremnih protoka primenom Ford-Falkersonovog algoritma

Algoritam

1° Inicijalizacija. Staviti: protok kroz mrežu $v = 0$, protoci kroz grane $f_{ij} = 0$, $\forall (i, j) \in L$, i skup $M = \{s\}$.

2° Pronaći $i \in M$ za koje važi $Q_i \neq \varnothing$. Ako je $Q_i = \varnothing$ za svako $i \in M$, preći na korak 6°.

3° Svakom čvoru $j \in Q_i$, dodeliti obeležje $[i, F_j]$, gde je $F_j = \min\{F_i, c_{ij} - f_{ij}\}$ i proširiti skup M elementima skupa Q_i, tj. $M = M \cup Q_i$.

4º Proveriti da li $t \in M$. Ako jeste, uvećati ukupan protok kroz mrežu za dodatni protok F_t, tj. $v = v + F_t$ i promeniti protoke f_{ij} onih grana koje se nalaze na putu kojim je dodatni protok F_t stigao od s do t. U suprotnom ići na sledeći korak.

5º Staviti $M = \{s\}$ i vratiti se na korak 2º.

6º Postupak je završen. Dobijeni protok v je maksimalan protok kroz mrežu, a grane (i, j) za koje važi: $i \in M$ i $j \in N \setminus M$, pripadaju minimalnom preseku mreže. KRAJ. ■

Napomene:

1) Protoci f_{ij} predstavljaju protoke kroz grane mreže, a razlika $c_{ij} - f_{ij}$ je neiskorišćen kapacitet pojedinih grana. Grane za koje važi $f_{ij} = c_{ij}$ pripadaju minimalnom preseku i predstavljaju usko grlo u mreži. Ukupan protok kroz mrežu v jednak je zbiru protoka grana koje se nalaze na minimalnom preseku.

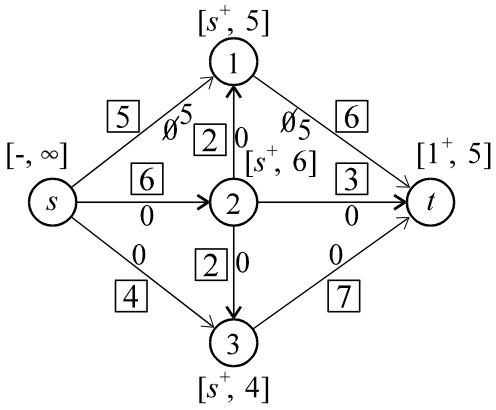
2) Isti algoritam se može primeniti i na razne modifikacije posmatranog problema. Na primer, problem ekstremizacije protoka kroz mrežu sa više izvorišta $(s_1, \ldots, s_p)$ i više ušća $(t_1, \ldots, t_q)$ se može svesti na problem sa jednim izvorištem i jednim ušćem tako što se uvedu novi čvorovi: s - fiktivno izvorište koje se poveže sa postojećim izvorištima granama $(s, s_1), \ldots, (s, s_p)$ i t - fiktivno ušće koje se poveže sa postojećim ušćima granama $(t_1, t), \ldots, (t_q, t)$. Ovim granama se dodeli dovoljno veliki kapacitet tako da one ne mogu da pripadaju minimalanom preseku. Na ovako modifikovanu mrežu se bez ikakvih izmena može primeniti navedeni algoritam. Takođe, sa malim modifikacijama algoritma, mogu se rešavati problemi sa ograničenim protokom čvorova ili sa zadatim minimalnim kapacitetima grana.

Primer 1.15. Data je mreža sa zadatim kapacitetima grana (brojevi u kvadratićima).

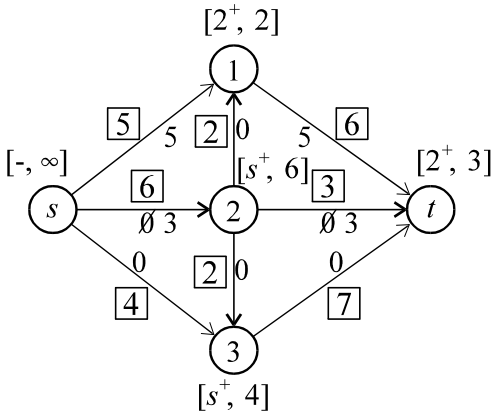
a) Odrediti maksimalan protok kroz mrežu od izvora s do ušća t i minimalan presek mreže.

b) Ako je cena povećanja protoka u svim granama jednaka, šta je potrebno uraditi da bi se protok cele mreže najracionalnije povećao?

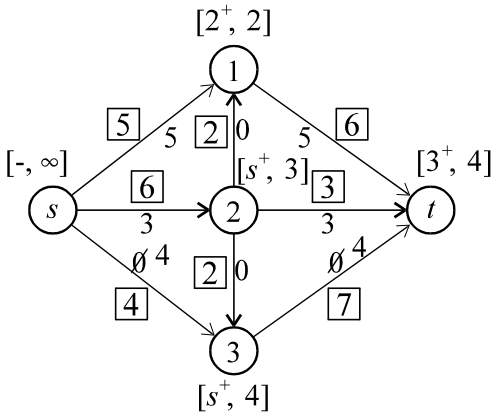
Rešenje: a)



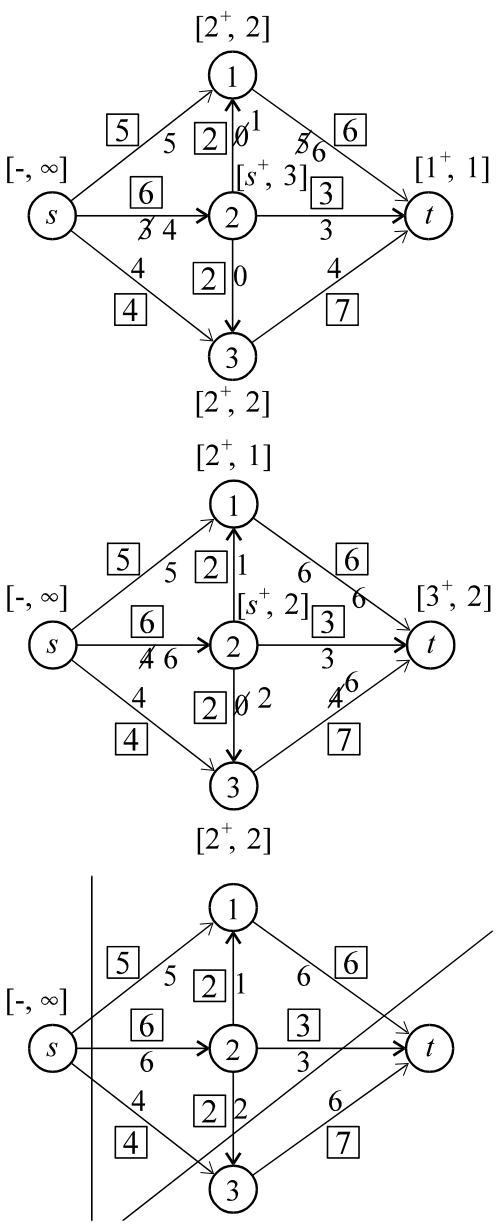
v=5



v=5+3=8



v=8+4=12

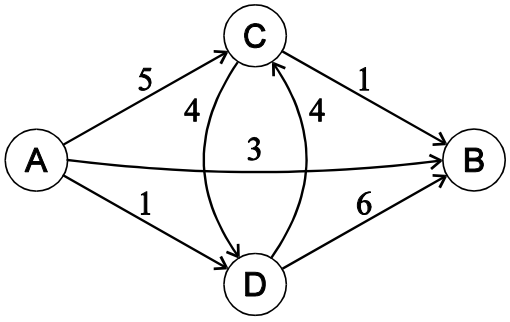


Slike 1.31. - 1.36.

Dalje obeležavanje je nemoguće. Jedan minimalan presek je $\{s\}/\{1, 2, 3, t\} = \{(s, 1), (s, 2), (s, 3)\}$, ali ima i drugi $\{s, 1, 2\}/\{3, t\} = \{(s, 3), (2, 3), (2, t), (1, t)\}$.

b) Da bi se povećao protok kroz mrežu, potrebno je povećati kapacitet minimalnog preseka. Pošto imamo dva minimalna preseka, moramo povećati kapacitet oba. Pošto grana (s, 3) pripada i jednom i drugom minimalnom preseku, povećanjem njenog kapaciteta će se povećati kapacitet oba minimalna preseka, a samim tim i maksimalan protok kroz mrežu. ■

Primer 1.16. U grad **B** koji je u ratnom okruženju, potrebno je dostaviti humanitarnu pomoć iz grada **A** koja se može dopremati samo avionima. Pored direktnih letova **AB**, pomoć je moguće isporučivati i preko gradova **C** i **D**. Broj letova između svaka dva grada je ograničen. Šematski prikaz avionskih linija i dozvoljeni broj letova na svakoj liniji je prikazan na slici 1.37:



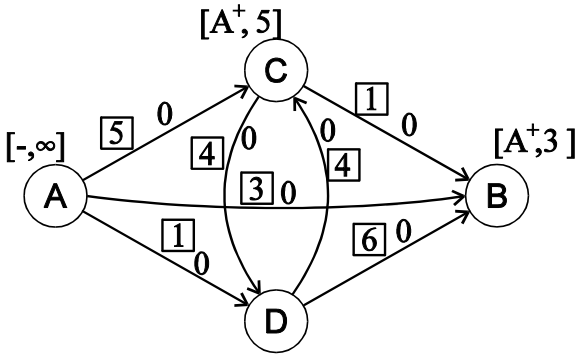
Slika 1.37.

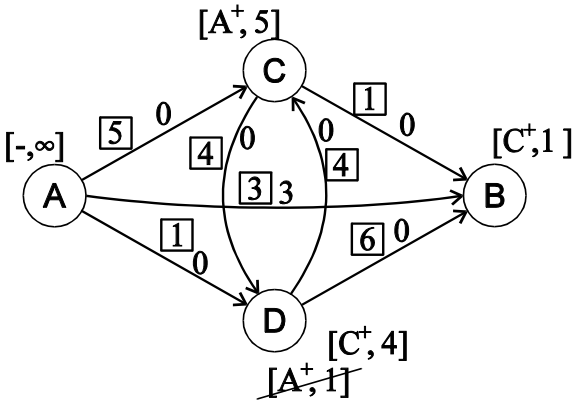
Potrebno je:

a) odrediti koliko je isporuka humanitarne pomoći najviše moguće dopremiti u grad **B**;

b) na kojoj liniji je najpotrebnije uvesti nove letove da bi se količina isporučene humanitarne pomoći najefikasnije povećala.

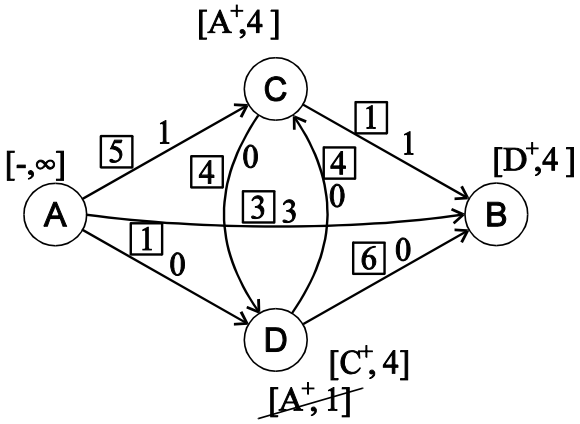
Rešenje: a) $v = 0, f_{ij} = 0, \forall\ (i,j) \in L$



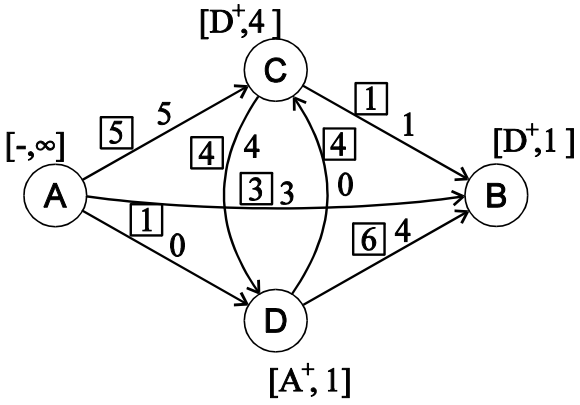


$v = 3, f_{AB} = 3$

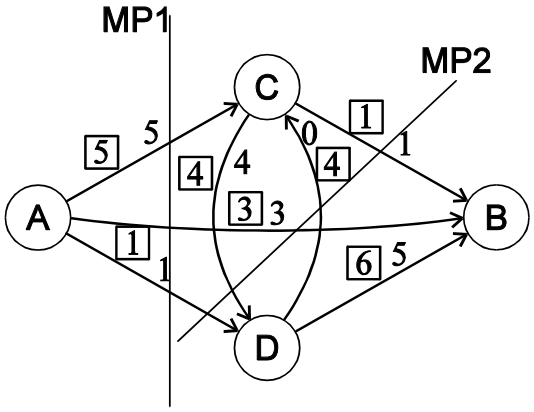
$v = 4, f_{CB} = 1, f_{AC} = 1$



$v = 4+4 = 8, f_{DB} = 4, f_{CD} = 4, f_{AC} = 1+4 = 5$



$v = 8+1 = 9, f_{DB} = 4+1 = 5, f_{AD} = 1$



Slike 1.38. - 1.42.

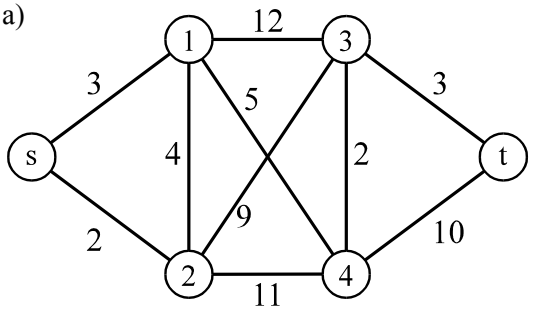
Odmah se uočava da dalje obeležavanje nije moguće. Minimalni preseci i protoci kroz grane su prikazani na slici 1.42, a ukupan protok kroz mrežu, tj. maksimalan broj isporuka je 9 i to: 3 direktna leta, jedan preko **C**, 4 preko **C** i **D**, i jedan samo preko **D**.

b) Da bi se povećao broj letova do grada **B**, potrebno je povećati broj letova na nekoj od linija koje se nalaze na minimalnom preseku. Pošto postoje dva minimalna preseka {**AC**, **AB**, **AD**} i {**AB**, **AB**, **CB**, **CD**}, potrebno je povećati broj letova na linijama koje su na oba preseka. Ovo će najefikasnije biti urađeno ako povećamo protok kroz linije koje su na oba preseka istovremeno, a to su: direktna linija **AB** i linija **AD** (na liniji **DB** ima mesta za još jedan let). ■

1.10. Rešeni zadaci

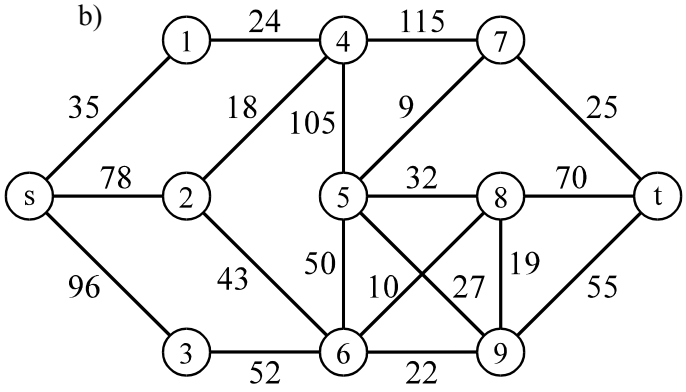
Zadatak 1.1. Koristeći Dijkstrin algoritam, odrediti najkraće puteve od čvora *s* do čvora *t*, na sledećim mrežama (slike 1.43, 1.44. i 1.45.):

a)

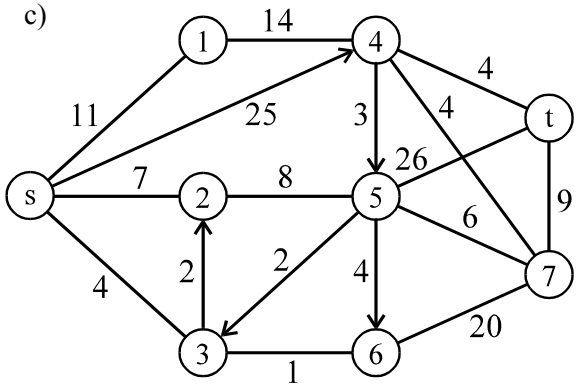


Slika 1.43.

b)



Slika 1.44.



Slika 1.45.

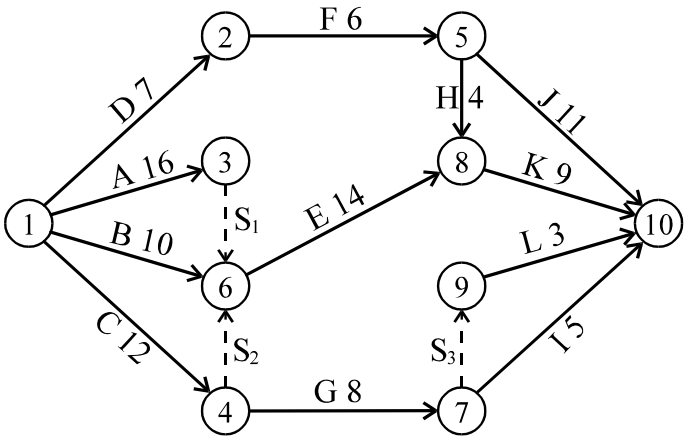
Rešenje: a) p : (s, 1, 4, 3, t); $d_p = 13$
b) p : (s, 1, 4, 2, 6, 8, 5, 7, t); $d_p = 196$
c) p : (s, 3, 2, 5, 7, 4, t); $d_p = 28$ ■

Zadatak 1.2. Projekat je definisan listom aktivnosti, njihovim zavisnostima i trajanjem:

Aktivnost	Zavisi od	$t_{(i-j)}$
A	-	16
B	-	10
C	-	12
D	-	7
E	A, B, C	14
F	D	6
G	C	8
H	F	4
I	G	5
J	F	11
K	E, H	9
L	G	3

Potrebno je nacrtati i numerisati mrežni dijagram ovog projekta i naći kritičan put koristeći Belmanov algoritam.

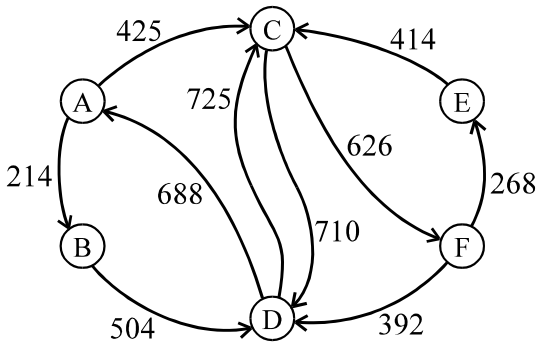
Rešenje:



Slika 1.46.

Nalaženje kritičnog puta se svodi na nalaženje najdužeg puta u mreži. Belmanov algoritam se može na više načina primeniti na ovaj problem: moguće je modifikovati sam algoritam tako da nalazi maksimum umesto minimuma, konvertovati trajanja aktivnosti u inverznu vrednost ($\frac{1}{t_{(i-j)}}$) ili (što je najjednostavnije) pomnožiti sva trajanja aktivnosti sa -1. Na ovaj način se dobija da kritični put prolazi kroz čvorove: (1, 3, 6, 8, 10), odnosno sastoji se od aktivnosti: A, E i K, a projekat se može završiti za $T = 39$ vremenskih jedinica. ■

Zadatak 1.3. Avionska kompanija treba da formira cenovnik letova vazdušnog saobraćaja između 6 gradova jednog područja. Skica gradova i vazdušnih koridora (kuda je jedino moguće leteti), data je na slici 1.47. Potrebno je odrediti najmanje udaljenosti u kilometrima između svaka dva grada tog područja.



Slika 1.47.

Rešenje: Ovaj problem je najlakše rešiti Flojdovim algoritmom. Rešenje je:

od\do	A	B	C	D	E	F
A	0	214	425	718	1319	1051
B	1192	0	1229	504	2123	1855
C	1398	1612	0	710	894	626
D	688	902	725	0	1619	1351
E	1812	2026	414	1124	0	1040
F	1080	1294	682	392	268	0

■

Zadatak 1.4. Koristeći Jenov algoritam rešiti sledeći zadatak nelinearnog celobrojnog programiranja:

$$(\min) f(x) = 6x_1^2 + \sqrt{x_2} + \frac{1}{2x_3}$$

$$p.o.$$

$$g(x) = x_1 + x_2 + x_3 \geq 5$$

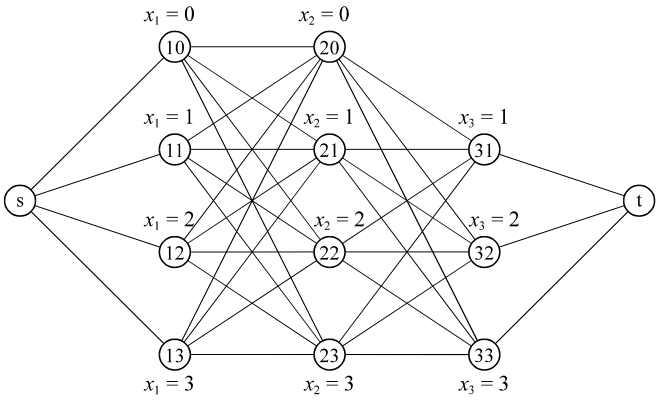
$$x_1, x_2, x_3 \in \{0, 1, 2, 3\}$$

$$x_3 \neq 0$$

Rešenje: Da bi zadatak (separabilnog) celobrojnog programiranja mogao da se rešava Jenovim algoritmom, potrebno je prvo napraviti njegovu grafovsku interpretaciju. To se radi tako, što se osim početnog i

krajnjeg čvora grafa, za svaku dopuštenu vrednost svake promenljive uvede po jedan čvor. Svi čvorovi koji predstavljaju jednu promenljivu se povežu granama sa svakim čvorom prethodne i sledeće promenljive, a čvorovi prve i poslednje promenljive se povežu za izvorište, odnosno ušće. Dobija se graf čije su grane uvek usmerene od izvora ka ušću. Svakoj od grana grafa se dodeli dužina koja odgovara vrednosti koju ima onaj deo funkcije cilja, u kom se promenljiva u čiji se čvor uliva grana nalazi, kada ta promenljiva ima vrednost koju taj čvor predstavlja. Grane koje se ulivaju u ušće imaju dužinu 0. Za zadati matematički model dobija se sledeća grafovska interpretacija:

Zbog preglednosti slike 1.48. grane su crtane bez strelica, ali se podrazumeva da su one usmerena s leva na desno. Dužine pojedinih grana su sledeće:



Slika 1.48.

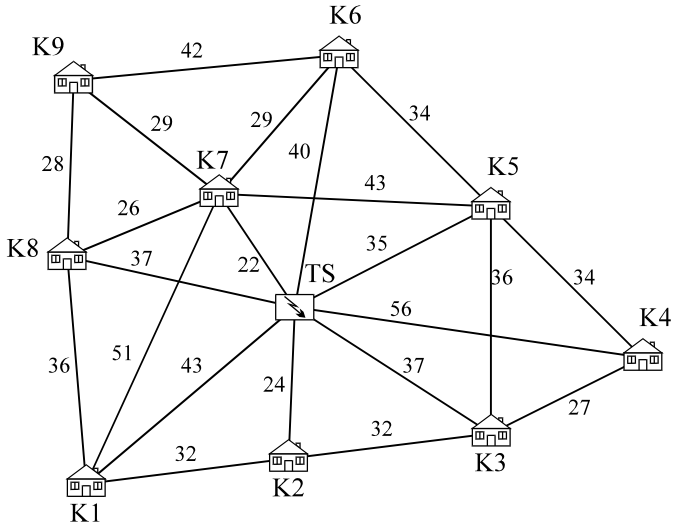
$c_{s10} = 0$ $c_{i20} = 0$
$c_{s11} = 6$ $c_{i21} = 1$ $c_{i31} = 0{,}5$ $c_{it} = 0$
$c_{s12} = 24$ $c_{i22} = 1{,}41$ $c_{i32} = 0{,}25$
$c_{s13} = 54$ $c_{i23} = 1{,}73$ $c_{i33} = 0{,}17$

gde i predstavlja sve čvorove koji prethode datom čvoru.
Kod primene Jenovog algoritma treba tražiti najkraće puteve sve dok ne bude ispunjen uslov $x \in X$. Za zadati primer, Jenov algoritam daje sledeće rezultate:

Rb	Dužina puta	Puta	Rešenje x	$g(x)$
1	0,17	(s, 10, 20, 33, t)	(0, 0, 3)	3
2	0,25	(s, 10, 20, 32, t)	(0, 0, 2)	2
3	0,50	(s, 10, 20, 31, t)	(0, 0, 1)	1
4	1,17	(s, 10, 21, 33, t)	(0, 1, 3)	4
5	1,25	(s, 10, 21, 32, t)	(0, 1, 2)	3
6	1,50	(s, 10, 21, 31, t)	(0, 1, 1)	2
7	1,58	(s, 10, 22, 33, t)	(0, 2, 3)	5

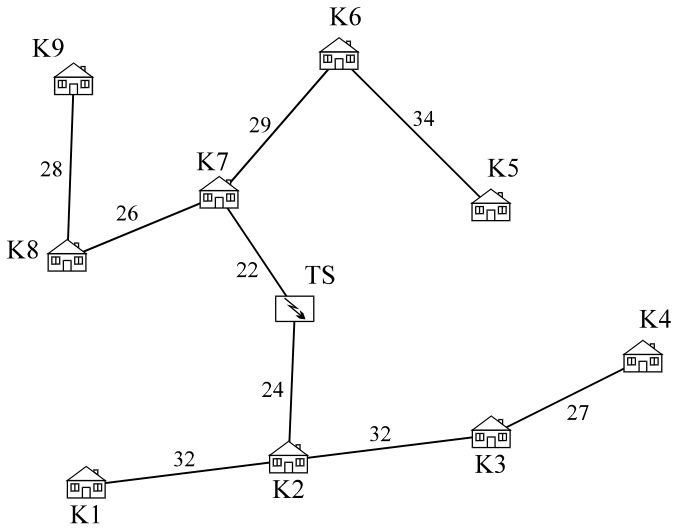
Vidi se, da tek sedmi najkraći put zadovoljava uslov modela. Konačno rešenje modela glasi: $x_1^* = 0, x_2^* = 2, x_3^* = 3, f^* = 1{,}58$.■

Zadatak 1.5. Elektrodistribucija treba da poveže 9 domaćinstava jednog naselja na lokalnu trafostanicu. Mapa naselja i udaljenost između pojedinih kuća i trafostanice dati su na slici 1.49. Treba napraviti plan povezivanja trafstanice i domaćinstava, a da dužina električnih vodova bude što kraća.



Slika 1.49.

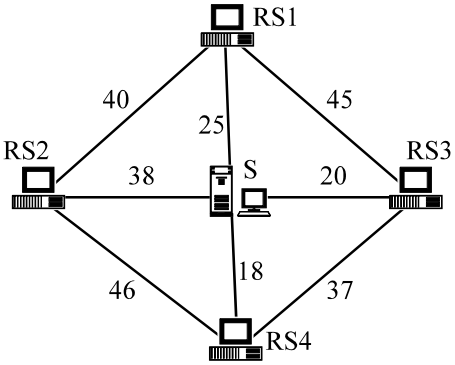
Rešenje: Ovaj zadatak se može posmatrati kao problem najkraćeg razapinjućeg stabla:



Slika 1.50.

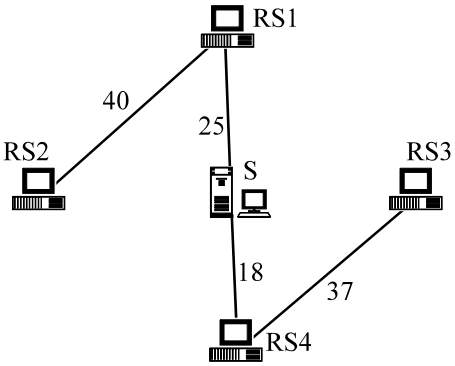
Najkraća dužina vodova iznosi 254. ■

Zadatak 1.6. U upravnoj zgradi preduzeća je potrebno postaviti računarsku mrežu koja se sastoji od jednog servera i 4 radne stanice. Raspored računara i potrebna dužina koaksijalnog kabla potrebnog za povezivanje (u metrima) data je na slici 1.51. Odrediti na koji je način najjeftinije umrežiti računare ako se zna da je cena jednog metra kabla 7 din, cena dodatne mrežne kartice 150 din, a jedan terminator košta 10 din.
(Umrežavanje se izvodi tako što do svakog računara mora da dođe kabal koji se priključuje na mrežnu karticu računara. Svaka mrežna kartica ima dva priključka, tako da se svaki računar pomoću jedne kartice može direktno povezati sa dva druga računara. Ako je jedan od priključaka slobodan, na njega se mora staviti terminator. Ako se računar želi direktno povezati sa više od dva računara, mora se dodati još jedna mrežna kartica. Računari se najčešće povezuju redno, tako da svaki ima po jednu karticu, a na prvi i poslednji se stavi terminator. Druga mogućnost je da se svaki računar poveže sa najbližim.)

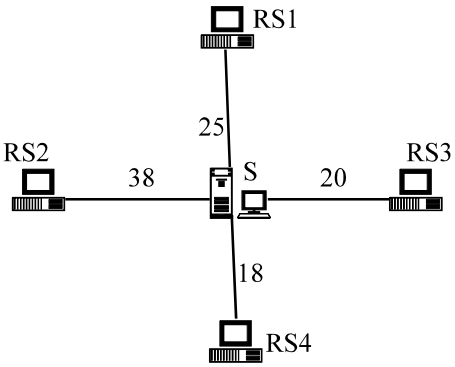


Slika 1.51.

Rešenje: Da bi se izračunala dužina potrebnih kablova za rednu vezu računara, potrebno je pronaći najkraći Hamiltonov put, s tim što nije bitno koji je računar prvi, a koji poslednji. Najkraći Hamiltonov put kroz zadatu mrežu (dobijen korišćenjem DP) je hp = RS3 - RS4 - S - RS1 - RS2, dužine d_{hp} = 120 m (slika 1.52.).



Slika 1.52.



Slika 1.53.

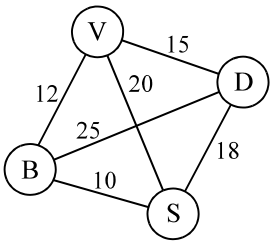
Najmanja dužina kablova potrebna za umrežavanje datih računara se može dobiti ako se nađe najkraće razapinjuće stablo grafa. Struktura ovog stabla je data na slici 1.53, a ukupna potrebna dužina kablova je $d_{SST} = 101$ m, dakle, 19 m kraća nego kod redne veze. Da bi ova mreža funkcionisala, potrebno je dodati još jednu mrežnu karticu (na serveru) i dva terminatora (na RS1 i RS4).
Ako uporedimo finansijske efekte ove dve varijante, dobijamo: $150 + 2 \times 10 - 19 \times 7 = 37$, tj. iako je kod redne veze potrebno $19m$ kablova više, ova varijanta je ipak za 37 dinara jeftinija. ■

Zadatak 1.7. Raznosač pica mora da isporuči 3 pice i da se opet vrati u piceriju po novu turu. Picerija se nalazi na Voždovcu, a mesta isporuke su na: Slaviji, Dedinju i Braće Jerković. Raznosač za prevoz koristi motorcikl, a iz iskustva zna da mu je za prevoz potrebno sledeće vreme (u minutima):

od \ do	Voždovca	Slavije	Dedinja	B. Jerković
Voždovca	0	20	15	12
Slavije	20	0	18	10
Dedinja	15	18	0	25
B. Jerković	12	10	25	0

Primenom DP odrediti kojim putem treba da prođe raznosač da bi se što pre vratio po sledeću turu.

Rešenje:



Slika 1.54.

$f(\{D\}, S) = \underline{c_{VD}} + \underline{c_{DS}} = \underline{15 + 18} = 33$
$f(\{D\}, B) = c_{VD} + c_{DB} = 15 + 25 = 40$
$f(\{B\}, D) = c_{VB} + c_{BD} = 12 + 25 = 37$
$f(\{B\}, S) = \underline{c_{VB}} + \underline{c_{BS}} = \underline{12 + 10} = 22$
$f(\{S\}, D) = c_{VS} + c_{SD} = 20 + 18 = 38$
$f(\{S\}, B) = c_{VS} + c_{SB} = 20 + 10 = 30$

Druga etapa:
$f(\{D, B\}, S) =$

$$= \min_{i\in\{B,D\}} \begin{Bmatrix} f(\{D\},\ B) + c_{BS} \\ f(\{B\},\ D) + c_{DS} \end{Bmatrix} = \min \begin{Bmatrix} 40 + 10 \\ 37 + 18 \end{Bmatrix} = 50$$

$f(\{D, S\}, B) =$

$$= \min_{i\in\{D,S\}} \begin{Bmatrix} f(\{D\},\ S) + \underline{c_{SB}} \\ f(\{S\},\ D) + c_{DB} \end{Bmatrix} = \min \begin{Bmatrix} \underline{33 + 10} \\ 38 + 25 \end{Bmatrix} = 43$$

$f(\{B, S\}, D) =$

$$= \min_{i\in\{B,S\}} \begin{Bmatrix} f(\{B\},\ S) + \underline{c_{SD}} \\ f(\{S\},\ B) + c_{BD} \end{Bmatrix} = \min \begin{Bmatrix} \underline{22 + 18} \\ 30 + 25 \end{Bmatrix} = 40$$

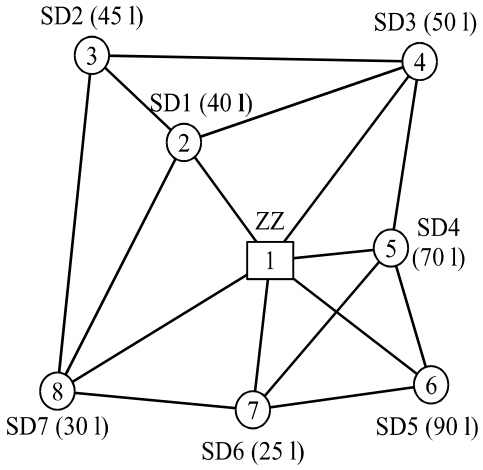
I na kraju treća, završna etapa u kojoj dobijamo konačno rešenje:
$f^* = f(\{D, B, S\}, V) =$

$$= \min_{i\in\{D,B,S\}} \begin{Bmatrix} f(\{D,B\}, S) + c_{SV} \\ f(\{D,S\}, B) + \underline{c_{BV}} \\ f(\{B,S\}, D) + \underline{c_{DV}} \end{Bmatrix} = \min \begin{Bmatrix} 50 + 20 \\ \underline{43 + 12} \\ \underline{40 + 15} \end{Bmatrix} = 55$$

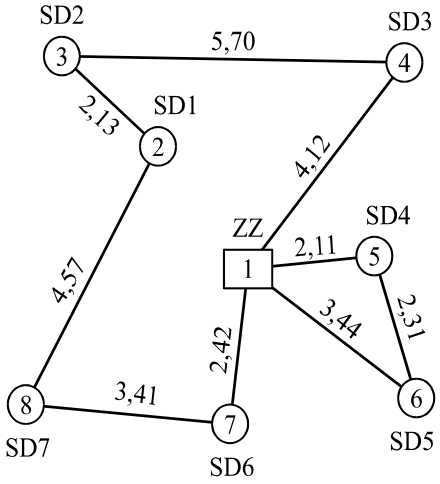
Raznosač će da napravi sledeću turu: Voždovac - Dedinje -

- Slavija - B. Jerković - Voždovac (ili isti put, ali u suprotnom smeru), a vratiće se za 55 minuta. ■

Zadatak 1.8. Zemljoradnička zadruga svakog dana sakuplja mleko proizvedeno u sedam obližnjih seoskih domaćinstava. Prikupljanje se vrši u cisternu zapremine 200 l, a raspored seoskih domaćinstava i zemljoradničke zadruge, postojeći putevi između njih, kao i očekivana količina mleka preuzeta od svakog domaćinstva su prikazani na slici 1.55. Udaljenosti između pojedinih tačaka su (u km): $c_{12} = 2{,}48$, $c_{14} = 4{,}12$, $c_{15} = 2{,}11$, $c_{16} = 3{,}44$, $c_{17} = 2{,}42$, $c_{18} = 4{,}25$, $c_{23} = 2{,}13$, $c_{24} = 4{,}30$, $c_{28} = 4{,}57$, $c_{34} = 5{,}70$, $c_{38} = 5{,}43$, $c_{45} = 3{,}04$, $c_{56} = 2{,}31$, $c_{57} = 3{,}54$, $c_{67} = 3{,}12$, $c_{78} = 3{,}41$. Odrediti koju najkraću rutu mora da pređe cisterna da bi pokupila mleko iz svih domaćinstava.



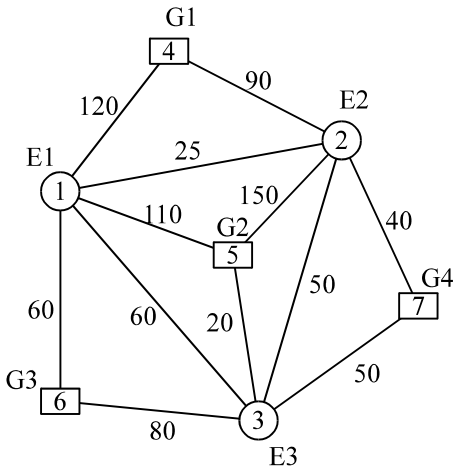
Slika 1.55.



Slika 1.56.

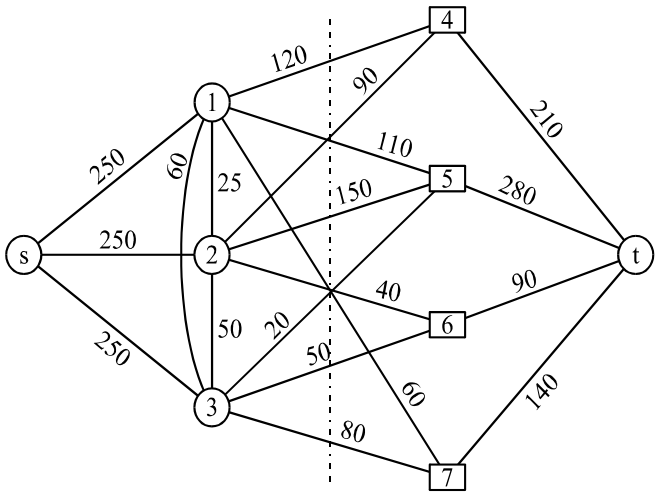
Rešenje: Moguće sortirane uštede (usvojeno da je $c_{13} = c_{12} + c_{23}$):
$S_{23} = 4{,}96$, $S_{78} = 3{,}26$, $S_{56} = 3{,}24$, $S_{45} = 3{,}19$, $S_{34} = 3{,}03$, $S_{67} = 2{,}74$, $S_{28} = 2{,}49$, $S_{24} = 2{,}30$, $S_{57} = 0{,}99$, $S_{38} = 0{,}95$.
Nakon primene algoritma ušteda, dobija se rešenje kao na slici 2,56. pri čemu je ukupna ostvarena ušteda u odnosu na početno rešenje $S = 16{,}98$. Ukupna dužina rute iznosi 30,21 km. ■

Zadatak 1.9. U jednom regionu se nalaze tri elektrane i četiri veća grada kojima se isporučuje struja. Raspored elektrana, gradova, postojećih linija dalekovoda i njihovi kapaciteti (u MWh) su prikazani na slici 2,57. Sve tri elektrane imaju kapacitet 250 MWh. Potrebno je odrediti, kojim će se linijama i u kojim količinama isporučivati električna energija da bi gradovi bili što bolje snabdevani.



Slika 1.57.

Rešenje: Pošto je ovo problem sa više izvorišta i više ušća, moramo ga prvo svesti na problem sa jednim izvorištem i jednim ušćem. Uvodimo fiktivno izvorište *s* i fiktivno ušće *t*. Kapacitete elektrana ćemo uključiti u algoritam tako što će grane od fiktivnog izvorišta do elektrana imati kapacitet jednak kapacitetu elektrana. Kapaciteti grana od svakog grada do ušća će biti jednaki zbiru svih grana koje u grad ulaze. Modifikovana mreža i stvarni protoci kroz grane su prikazani na slici 2.58. Maksimalan protok kroz mrežu iznosi $v = 720$ MWh, a minimalni presek čine sve grane koje idu od elektrana do gradova.



Slika 1.58.

■

Zadatak 1.10. Firma „A“ želi da kupi robu od firme „B“, ali pošto joj nisu namireni dugovi, nije u stanju da plati željenu robu, pa je odlučeno da se uplata izvrši putem prebijanja dugova. Poznato je da: „B“ duguje firmama „F“, „G“ i „H“ i to 26, 17 i 4 miliona respektivno; „F“ duguje firmi „G“ 15, a „C“ 5 miliona; „G“ duguje „C“, „D“ i „E“ po 3, 10 i 16 miliona (respektivno); „H“ duguje „C“ 8, a „E“ 11 miliona; „E“ duguje firmi „D“ 9 miliona; a i „C“ i „D“ i „E“ duguju firmi „A“ po 18, 24 i 13 miliona.
Odrediti koliku vrednost robe može „A“ da kupi od „B“ tako da uplatu izvrši isključivo prebijanjem dugova. Odrediti koliko duga svaka firma treba da oprosti, i kojoj firmi, da bi transakcija mogla da se obavi.

Rešenje: Ovaj problem se može predstaviti kao mreža, tako što se firme predstave kao čvorovi, a dugovanja kao orijentisane grane sa maksimalnim protocima koji imaju vrednost dugovanja. Rešenje se dobija tako što se nađe maksimalan protok kroz mrežu, a protoci kroz pojedine grane predstavljaju iznos duga koji firme treba da oproste jedna drugoj. Ovakav postupak daje sledeće rešenje:
„A“ može da kupi od „B“ robu u vrednosti od 38 miliona. Da bi

se ova uplata ostvarila, potrebno je da:

Firma:	A	A	A	C	C	C	D	D	E	F	G	G	H
Oprašta dug:	C	D	E	F	G	H	E	G	G	B	F	B	B
U iznosu:	12	13	13	5	3	4	3	10	16	17	12	17	4

■