

Problem traženja minimalnog povezujućeg stabla

Kerim Kadušić

Elektrotehnički fakultet

Univerzitet u Sarajevu

Sarajevo, Bosna i Hercegovina

kkadusic1@etf.unsa.ba

Lino Bevanda

Elektrotehnički fakultet

Univerzitet u Sarajevu

Sarajevo, Bosna i Hercegovina

lbevanda1@etf.unsa.ba

Sažetak—U ovom radu je razmatran problem traženja minimalnog povezujućeg stabla uz objašnjenje osnovnih pojmljova teorije grafova koji su relevantni za ovaj problem. Proučavane su metode i raspoloživi algoritmi, nakon čega se pristupilo detaljnoj analizi odabranog Primovog algoritma koja je bila potrebna za uspješnu implementaciju istog u programskom jeziku Python. Implementirani algoritam se potom primjenio na primjere problema praktične prirode što je omogućilo daljnju analizu principa traženja minimalnog povezujućeg stabla uz slikoviti prikaz svakog koraka algoritma. Na samom kraju testirane su performanse algoritma, te predložen način optimizacije istog.

Ključne riječi—Minimalno povezuće stablo, Primov algoritam, graf, matrica susjedstva

I. UVOD

S razvojem računarske nauke pojavljivali su se različiti problemi koji su se opisivali različitim strukturama podataka. Među strukturama podataka su se posebno isticali grafovi koji svoju primjenu nemaju samo u računarstvu i informatici, nego i u drugim domenima poput matematike i elektrotehnike. Snaga grafova se ogleda u mogućnosti objedinjavanja skupine objekata i njihovih relacija. Graf definisemo kao strukturu podataka koju čini skupina objekata (čvorovi) i koji su međusobno povezani relacijama ili vezama (grane grafa) [1].

Za grafove se vežu uobičajeni problemi koji se korištenjem različitih algoritama uspješno rješavaju. Neki od poznatijih problema su navedeni u nastavku:

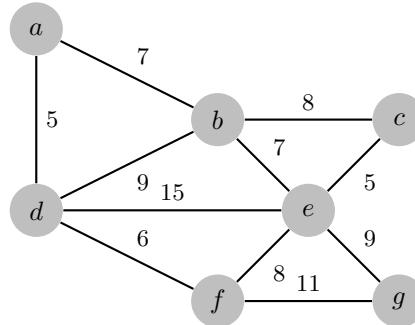
- Otkrivanje najdužeg ili najkraćeg puta između dva čvora
- Pretraživanje grafova
- Problem maksimalnog protoka
- Problem traženja minimalnog povezujućeg stabla
- Uparivanje u grafovima

U fokusu razmatranja ovog rada će biti problem traženja minimalnog povezujućeg stabla. Prije upoznavanja s ovim problemom potrebno je definisati pojам stabla. Stablo predstavlja neusmjereni povezani aciklični graf, pri čemu neusmjereni grafovi predstavljaju grafove kod kojih je svaka veza između dva čvora x i y obostrana. Pod acikličnim grafovima se podrazumjevaju oni grafovi koji ne posjeduju cikluse [2]. Da bi neusmjereni povezani aciklični graf bio stablo mora ispuniti nekoliko uslova:

- 1) Ako graf ima n čvorova, onda on mora posjedovati $n - 1$ grana

- 2) Graf koji predstavlja stablo gubi svojstvo povezanosti ukoliko mu se ukloni bilo koja grana, odnosno u tom slučaju graf više nije stablo
- 3) Ako se između svaka dva čvora grafa nalazi tačno jedan put koji ne prolazi više puta kroz isti čvor (elementarni put)

Za navedene grafove koji ne ispunjavaju ove uslove vrijedi da su interno sadržani od barem jednog podgraфа koji ispunjava uslove tj. svaki povezani aciklični neusmjereni graf posjeduje barem jedno stablo. Takva stabla se nazivaju povezujućim stablima ili kosturima grafa [1].



Slika 1: Primjer acikličnog neusmjerenog grafa

Na Slici 1. svakoj grani grafa je pridružena vrijednost. Ta vrijednost predstavlja težinu grane, te stoga prikazani graf predstavlja i težinski graf. Problem traženja minimalnog povezujućeg stabla se zasniva na pronalasku povezujućeg stabla tako da je zbir težina grana koje ga čine najmanji.

Ovaj problem se u praksi najčešće povezuje sa problemima povezivanja svih čvorova tako da se upotrijebi što manje resursa. Neka na primjer postoji ograničena dužina kablova koje je potrebno iskoristiti za povezivanje dalekovoda međusobno u elektrodistribucionu mrežu. Rješavanjem problema minimalnog povezujućeg stabla moguće saznati minimalnu potrebnu dužinu kablova koja se treba iskoristiti za povezivanje dalekovoda. Generalno, uspješnim rješavanjem ovog problema postiže se tražena ušteda resursa.

Za problem rješavanja minimalnog povezujućeg stabla su razvijeni efikasni algoritmi. Među njima se posebno ističu Kruskalov i Primov algoritam. U nastavku će biti riječi o nekim poznatijim algoritmima za rješavanje problema minimalnog povezujućeg stabla.

A. Borůvka algoritam

Borůvka algoritam je prvi algoritam korišten u oblasti nalaženja minimalnog povezujućeg stabla. Otkrio ga je 1926. godine češki matematičar Otakar Borůvka, tražeći efikasni način postavljanja električne mreže [3].

B. Kruskalov algoritam

Kruskalov algoritam se obično primjenjuje nad rijetkim grafovima. U prvoj iteraciji je potrebno uzeti granu koja ima najmanju težinu. Ukoliko postoji više grana sa istom težinom koja je ujedno i najmanja, svejedno je koja će biti odabrana. U narednim iteracijama je potrebno uzeti granu s najmanjom težinom od onih grana koje nisu do tada uzete, tako da ta grana ne obrazuje ciklus s do tada uzetim granama. Ovaj postupak se iterativno ponavlja sve dok se ne obrazuje povezujuće stablo [4].

II. KORIŠTENI ALGORITAM

Za potrebe ovog rada implementiran je Primov algoritam. Primov algoritam je razvio češki matematičar Vojtěch Jarník 1930. godine koji je svoju popularnost stekao tek 30-ak godina kasnije, kada su Robert C. Prim i Edsger W. Dijkstra uvidjeli mogućnost primjene algoritma u računarskoj nauci i odlučili republicirati algoritam. Primov algoritam dolazi u različitim verzijama s obzirom da je osnovna verzija bila reda vremenske kompleksnosti $\mathcal{O}(n^3)$ gdje n predstavlja broj čvorova grafa, što predstavlja lošiju vremensku kompleksnost od jednostavnijeg Kruskalovog algoritma reda $\mathcal{O}(m \log_2 m)$. Poboljšane verzije Primovog algoritma, osim bolje vremenske kompleksnosti, pogodnije su za primjenu kod gustih grafova u odnosu na druge algoritme [2].

Bitno je napomenuti da Primov algoritam, kao i Borůvka i Kruskalov algoritam, spada u porodicu tzv. *pohlepnih* ili *proždrljivih* (engl. *greedy*) algoritama, što znači da u svakom koraku vrši izbor one odluke koja u tom trenutku najviše doprinosi konačnom cilju, ne vodeći računa kako će se taj izbor odraziti na eventualne odluke koje će biti naknadno donesene. Generalno problem minimalno povezujućeg stabla ima izuzetno mnogo dopustivih rješenja koja ne moraju biti optimalna. Prema *Cayleyevoj* teoremi iz teorije grafova, graf sa n čvorova može imati čak do n^{n-2} povezujućih stabala [4].

Osnova ideja rada Primovog algoritma je sljedeća: Za početak je potrebno odabratи proizvoljan čvor grafa te u kostur dodati granu s najmanjom težinom koja izlazi iz tog čvora. U narednoj iteraciji u stablo se dodaje grana sa najmanjom težinom od grana koje spajaju čvorove koji se trenutno nalaze u stablu s njima susjednim čvorovima koji još nisu ušli u stablo. Ovaj postupak se ponavlja sve dok se ne formira povezujuće stablo (kostur), odnosno sve dok se ne izabere $n-1$ grana (pri čemu je n broj čvorova grafa) [1].

Dakle kod Primovog algoritma u svakoj iteraciji postoji stablo koje se postepeno proširuje kako izvođenje algoritma napreduje, za razliku od već opisanog Kruskalovog algoritma kod kojeg odabrane grane formiraju stablo tek na samom kraju algoritma [1].

Implementacija koju prati ovaj rad je fokusirana samo na osnovnu verziju Primovog algoritma s obzirom da se za predstavljanje grafova koristi matrica susjedstva, a osnovna verzija je najpogodnija za grafove predstavljene matricom susjedstva. U nastavku će biti opisan princip rada ove verzije Primovog algoritma:

Algoritam 1 Pseudokod osnovne verzije Primovog algoritma

```
1: procedure MST-PRIM (G, s)
2:    $U \leftarrow \{s\}$ 
3:    $E' \leftarrow \emptyset$ 
4:   while ( $U \neq V$ ) do
5:      $\text{pronaći}(u, v) \Rightarrow \min\{w(u, v) : (u \in U) \text{ and } (v \in (V - U))\}$ 
6:      $U \leftarrow U + \{v\}$ 
7:      $E' \leftarrow E' + \{(u, v)\}$ 
8:   end while
9:    $MST \leftarrow (U, E')$ 
10:  return MST
11: end procedure
```

Metoda MST-PRIM prima dva parametra: graf i početni čvor. Na početku se inicijaliziraju skupovi posjećenih čvorova U i skup grana koje obuhvata minimalno povezujuće stablo (linije 2-3). Nakon inicijalizacije slijedi glavni dio algoritma uutar while petlje (linije 4-8). Uslov zaustavljanje petlje je kada algoritam posjeti sve čvorove grafa (skup V). Osnovna ideja Primovog algoritma je pronaći granu (u, v) sa najmanjom težinom pri čemu čvor u pripada skupu posjećenih čvorova U , a čvor v priprada skupu neposjećenih čvorova $V - U$ (linija 5). Nakon što se takva grana pronađe, ona se dodaje skupu E' (linija 7), a čvor v skupu posjećenih čvorova U (linija 6). Ovaj se postupak ponavlja sve dok se ne ispunii uslov zaustavljanja, odnosno posjete svi čvorovi grafa. Po ispunjavanju uslova zaustavljanja slijedi objedinjavanje skupova U i E' u minimalno povezujuće stablo (MST) koje metoda MST-PRIM vraća kao rezultat.

S obzirom na jednostavnost algoritma nije bilo potrebno svoditi problem na verziju koja bi bila pogodnija za korišteni algoritam.

III. SIMULACIJSKI REZULTATI

Implementacija Primovog jezika je obavljena u programskom jeziku Python 3.0.

Program 1: MST_PRIM metoda u Python-u

```
1 def MST_PRIM (G, s):
2   V = *range(0, len(G), 1)
3   U = {s}
4   E_prim = set()
5   ukupna_tezina = 0
6   while U != V:
7     cvor, min_grana, tezina, postoji =
8       pronadji_min(G, U, V)
9     if not postoji:
10      raise Exception("Cvor nije
11      pronadjen")
12      U.add(cvor)
```

```

11     E_prim.add(min_grana)
12     # Crtaj u svakoj iteraciji
13     draw_graph([G, E_prim, 0])
14     ukupna_tezina += tezina
15
    return [G, E_prim, ukupna_tezina]

```

Sa listinga Program 1. se može vidjeti da je implementacija skoro identična priloženom pseudokodu uz minorne izmjene. U drugoj liniji kôda je inicijaliziran skup svih čvorova, a zatim skup posjećenih čvorova i grana u skladu sa prvom i drugom linijom pseudokoda. Uz ove skupove inicijalizirana je i varijabla `ukupna_tezina` koja se odnosi na težinu minimalnog povezujućeg stabla. U glavnoj petlji (linija 6) poziva se metoda `pronadji_min` koja obavlja posao pronalaska najmanje grane za koju vrijedi izraz u četvrtoj liniji pseudokoda. Slijedi prikaz implementacije metode `pronadji_min`:

Program 2: `pronadji_min` metoda u Python-u

```

1 def pronadji_min (G, U, V):
2     razlika = [item for item in V if item
3                 not in U]
4     min, u, v, postoji = inf, 0, 0, False
5
6     for c_u in U:
7         for c_v in V:
8             if G[c_u][c_v] != sys.maxsize
9                 and c_v in razlika:
10                tezina = G[c_u][c_v]
11                if tezina < min:
12                    min, u, v = tezina,
13                           c_u, c_v
14
15    if min != inf:
16        postoji = True
17
18    return v, (u, v), min, postoji

```

U drugoj liniji inicijalizira se skup koji predstavlja razliku skupa V i U ($V - U$), kao i pomoćne varijable: minimalna težina pronađene grane (`min`), posjećeni i neposjećeni čvorovi (`u` i `v`), te varijabla `postoji`. Varijabla `postoji` služi kao indikator da li uopće postoje grane koje je moguće razmatrati, što se vrlo lako može otkriti provjerom da li je finalna vrijednost varijable `min` jednaka beskonačnosti (linija 11). Ova varijabla također ukazuje da graf koji se razmatra nije u ispravnom formatu uslijed čega se baca izuzetak u `MST_PRIM` metodi.

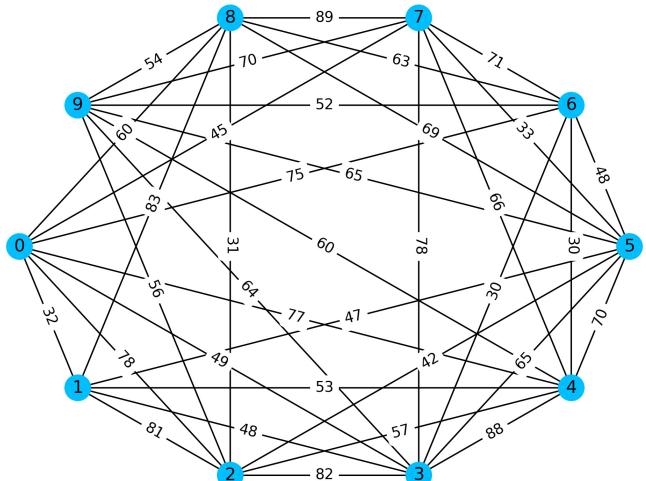
U linijama 5-10, prolazi se kroz skupove posjećenih i neposjećenih čvorova i u skladu sa već spomenutim izrazom iz pseudokoda (linija 5) traži najmanja grana. Na kraju, kao rezultat metode, vraća se neposjećeni čvor, minimalna grana i njena težina, te varijabla `postoji`. Vraćeni čvorovi i grane iz metode tako da mi je sad samo trebalo promijeni se dodaju u skupove U i E' respektivno u metodi `MST-PRIM`. `MST-PRIM` na kraju vraća razmatrani graf sa listom čvorova koji čine minimalno povezuće stablo kao i ukupnu težinu istog. U nastavku slijedi primjena implementiranog algoritma na praktičnim primjerima problema povezivanja dalekovoda u elektrodistribucionu mrežu i problema uvezivanja europskih gradova optičkim kablom.

A. Problem povezivanja dalekovoda

Prepostaviti će se da je kompanija A zadužena za postavljanje i povezivanje 10 dalekovoda. Međutim, uslijed nepredviđenih okolnosti su bili prinuđeni da drugačije rasporede dalekovode što je poremetilo planove povezivanja istih jer je potrebno ponovno utvrditi kolika je minimalna dužina kablova da bi se povezali svi dalekovodi. Ovaj problem će se modelirati pomoću grafova prikazanih sljedećom matricom susjedstva:

∞	32	78	49	77	∞	75	45	60	∞
32	∞	81	48	53	47	∞	∞	83	∞
78	81	∞	82	57	42	∞	∞	31	56
49	48	82	∞	88	65	30	78	∞	64
77	53	57	88	∞	70	30	66	∞	60
∞	47	42	65	70	∞	48	33	69	65
75	∞	∞	30	30	48	∞	71	63	52
45	∞	∞	78	66	33	71	∞	89	70
60	83	31	∞	∞	69	63	89	∞	54
∞	∞	56	64	60	65	52	70	54	∞

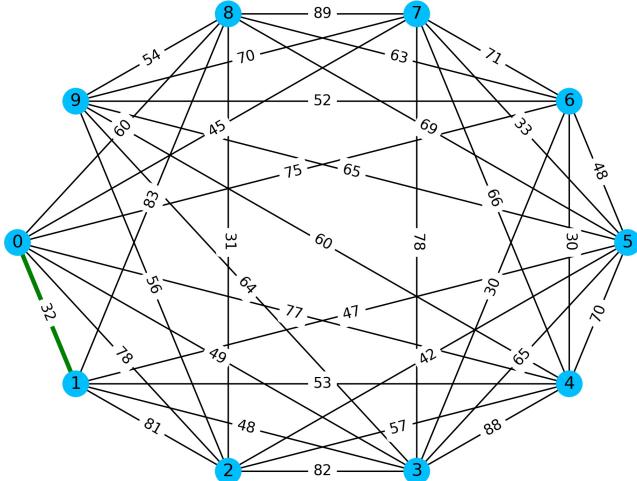
Svaki red ili kolona matrice predstavlja dužinu kabla potrebnog za povezivanje i-tog dalekovoda sa j-tim dalekovodom. Kako nije moguće povezati dalekovod sa samim sobom, na dijagonali matrice stoje vrijednosti beskonačnosti (∞). Također, vrijednost beskonačnosti označavaju nemogućnost povezivanja dva dalekovoda uslijed geografskih prepreka ili inicijalnih projektnih planova. Na osnovu postavke se formira početni graf kod kojeg svaki dalekovod predstavlja jedan čvor, a granama su prikazane potencijalne veze među dalekovodima:



Slika 2: Početni graf

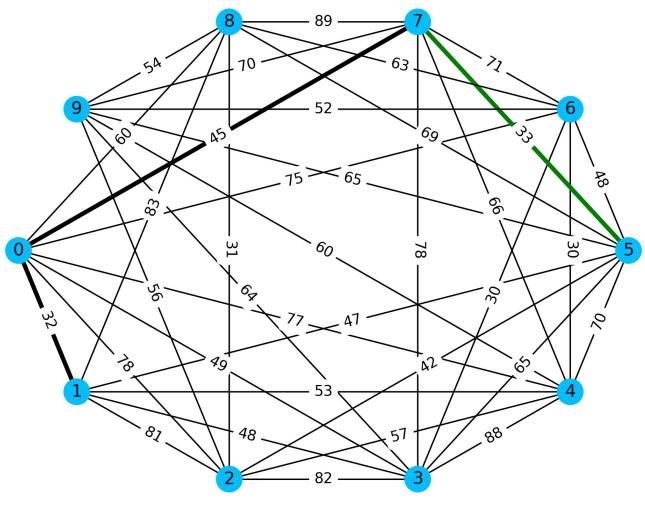
Prepostaviti će se da pretraga minimalnog povezujućeg stabla počinje od čvora 0. Na narednim figurama će biti prikazani svi koraci, odnosno sve iteracije potrebne za pronađak MST-a. Inicijalno u skupu posjećenih čvorova U se nalazi odabrani početni čvor 0 od kojeg pretraga kreće, skup grana stabla E' je prazan. U while petlji se poziva metoda `pronadji_min` i ona evaluira grane $\{0,1\}$, $\{0,2\}$, $\{0,3\}$, $\{0,4\}$, $\{0,6\}$, $\{0,7\}$ i $\{0,8\}$ tj. evaluiraju se sve grane koje nisu označene i čija je

težina nije jednaka beskonačnosti. Kako je iz priloženog grafa vidljivo, najmanju težinu ima grana $\{0, 1\}$ tako da se ta grana dodaje u skup E' , a čvor 1 u skup posjećenih čvorova U .

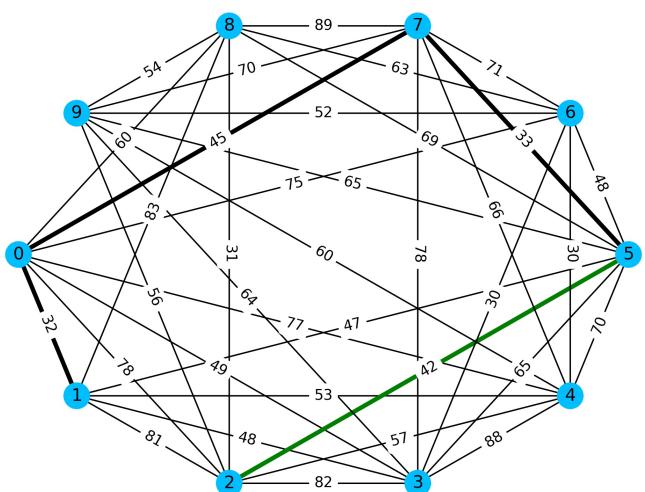


Slika 3: Prva iteracija

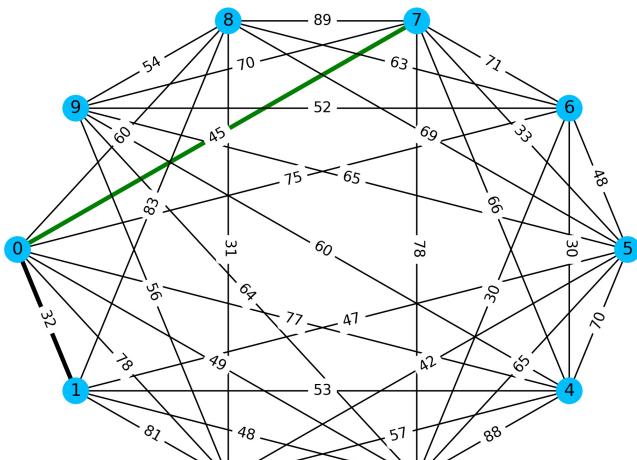
Nakon prvog koraka u skupu posjećenih čvorova se nalaze čvorovi 0 i 1, što znači da će vrijediti $V - U = \{2, 3, 4, 5, 6, 7, 8, 9\}$. Na osnovu izraza iz pseudokoda 1 (5 linija), algoritam će razmatrati sljedeći skup grana: $\{0, 2\}$, $\{0, 3\}$, $\{0, 4\}$, $\{0, 6\}$, $\{0, 7\}$, $\{0, 8\}$, $\{1, 2\}$, $\{1, 3\}$, $\{1, 4\}$, $\{1, 6\}$, $\{1, 7\}$ i $\{1, 8\}$. U navedenom skupu grana $\{0, 7\}$ ima najmanju težinu 45. Ova grana se također uvodi u skup E' , a čvor 7 u skup posjećenih čvorova. Kako ostatak algoritma teče na potpuno isti način, bez dodatnih objašnjenja će biti prikazan ostatak koraka (iteracija algoritma). U svakoj iteraciji je zelenom bojom označena grana koja je u toj iteraciji odabrana dok grane podebljane crnom bojom predstavljaju trenutne grane minimalno povezujućeg stabla.



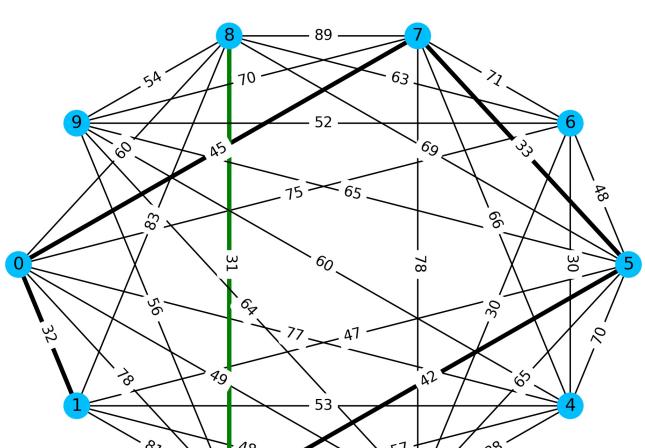
Slika 5: Treća iteracija



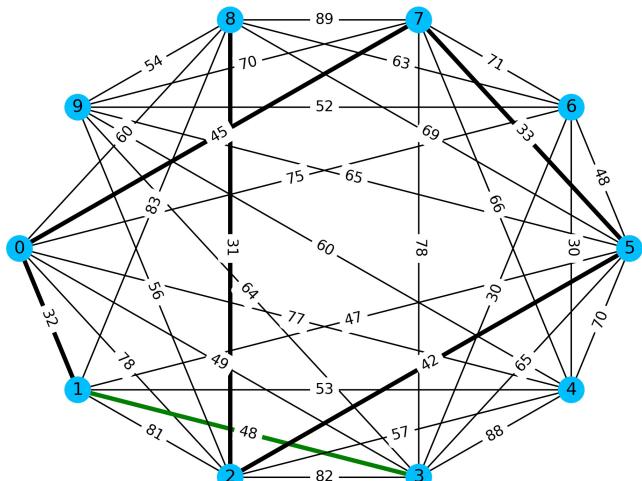
Slika 6: Četvrta iteracija



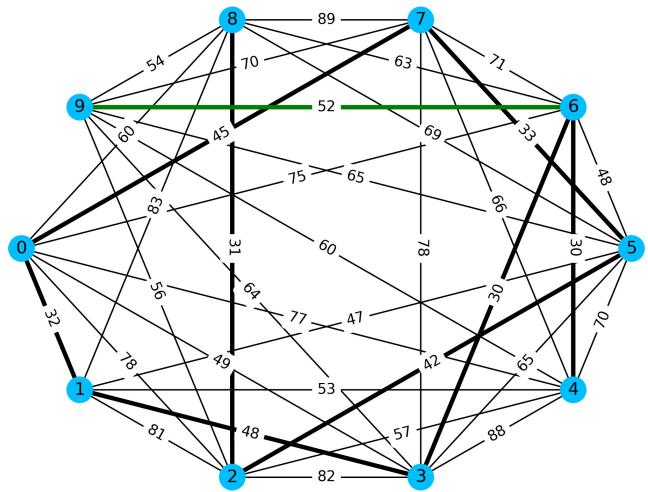
Slika 4: Druga iteracija



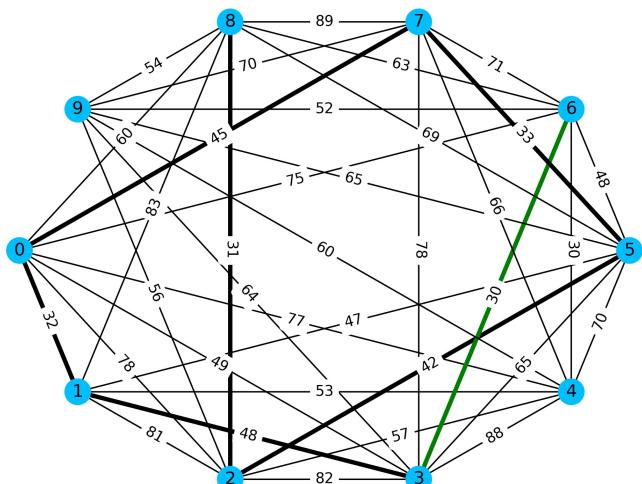
Slika 7: Peta iteracija



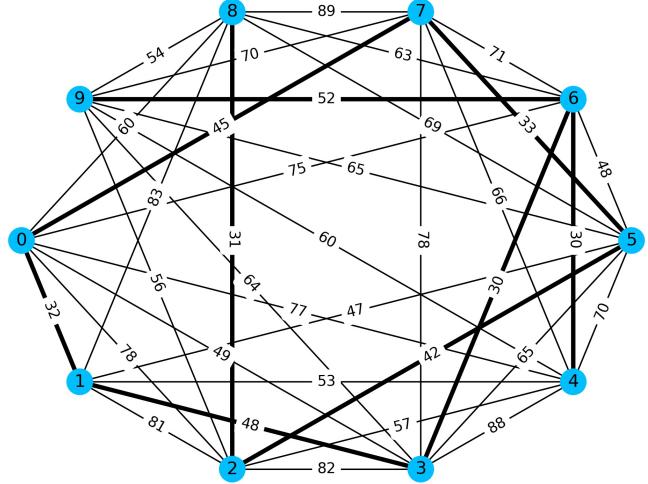
Slika 8: Šesta iteracija



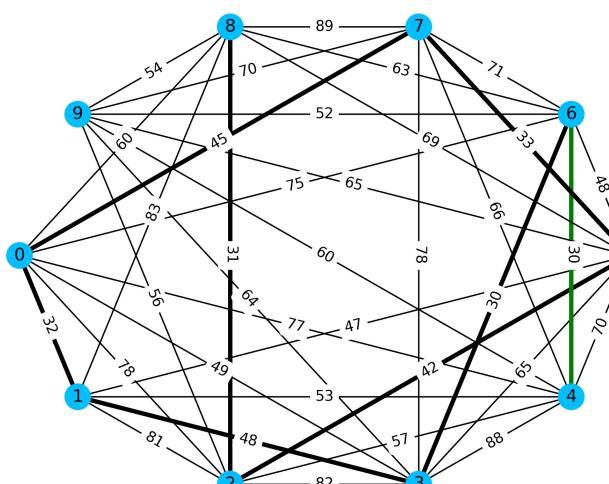
Slika 11: Deveta iteracija



Slika 9: Sedma iteracija



Slika 12: Konačno minimalno povezujuće stablo za problem povezivanja dalekovoda



Slika 10: Osma iteracija

Kako početni graf sadrži 10 čvorova, kostur je formiran nakon što je izabrano 9 grana: (0,1), (0,7), (7,5), (5,2), (2,8), (1,3), (6,3), (6,4) i (6,9). Traženo minimalno povezujuće stablo ima ukupnu težinu: $32 + 45 + 33 + 42 + 31 + 48 + 30 + 30 + 52 = 343$. Ono što je preostaje da se izračuna maksimalna vrijednost ušteda resursa. Da bi se to postiglo modificirat će se algoritam tako da u svakoj iteraciji uzima grane s najvećom težinom.

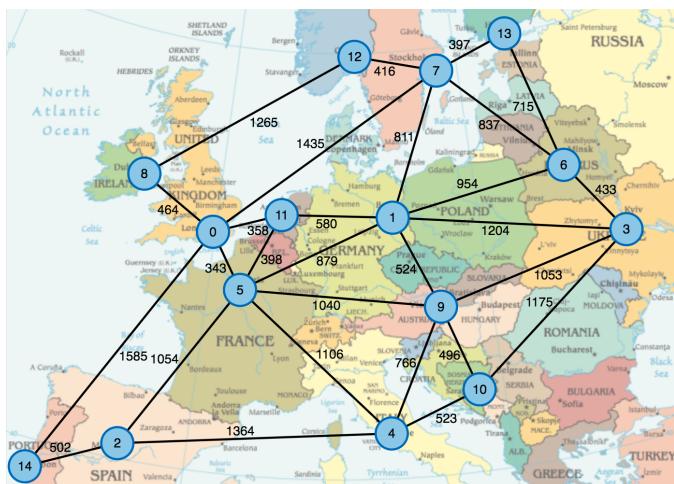
Odabrane grane su: (4, 5), (2, 1), (0, 6), (1, 8), (2, 3), (8, 7), (3, 4), (0, 2) i (7, 9), čije su sumirane težine jednake 716 metara. Iz ovoga slijedi da je postignuta značajna ušteda od 373 metra kablova što je oko 52% uštede.

B. Problem povezivanja gradova optičkim kablovima za Internet

Problem minimalnog povezujućeg stabla je pogodno primjeniti i na problem povezivanja gradova optičkim kablovima za Internet. Neka je potrebno povezati sljedeće europske

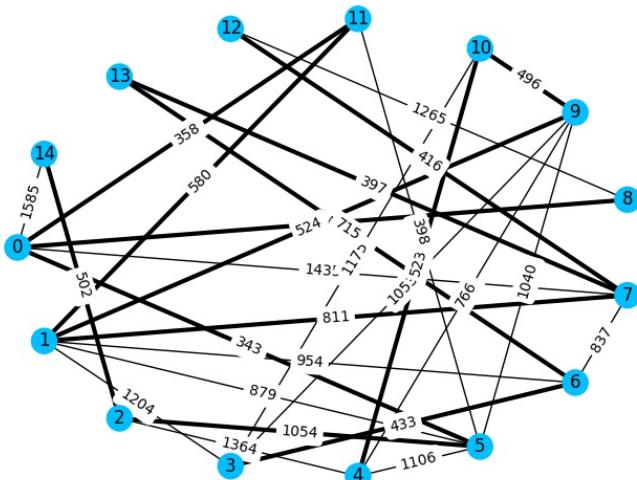
glavne gradove, gdje su težine grana stvarne zračne udaljenosti između gradova predstavljene u kilometrima, a broj u zagradi predstavlja redni broj čvora:

- London (0)
- Berlin (1)
- Madrid (2)
- Kiev (3)
- Rim (4)
- Pariz (5)
- Minsk (6)
- Štokholm (7)
- Dablin (8)
- Beč (9)
- Sarajevo (10)
- Amsterdan (11)
- Oslo (12)
- Helsinki (13)
- Lisabon (14)

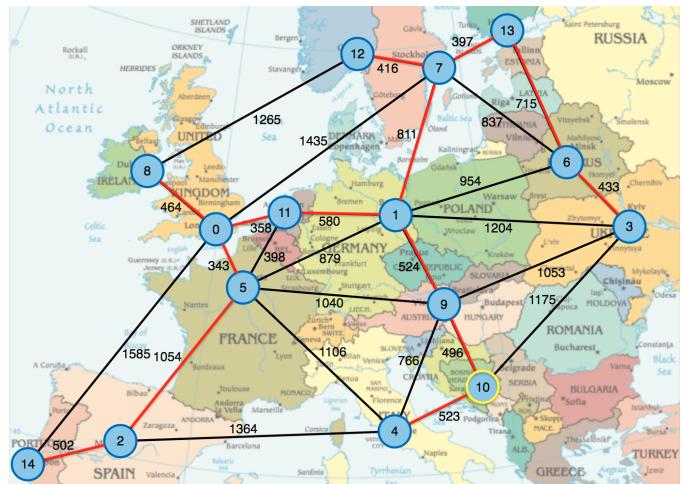


Slika 13: Povezivanje evropskih gradova

Kako se graf sastoji od 15 gradova (čvorova) neće biti prikazana svaka iteracija Primovog algoritma, već samo konačno rješenje. Ukoliko je grad Sarajevo (10) označen kao početni čvor, dobija se sljedeće minimalno povezuće stablo:



Slika 14: MST za evropske gradove - Python plot



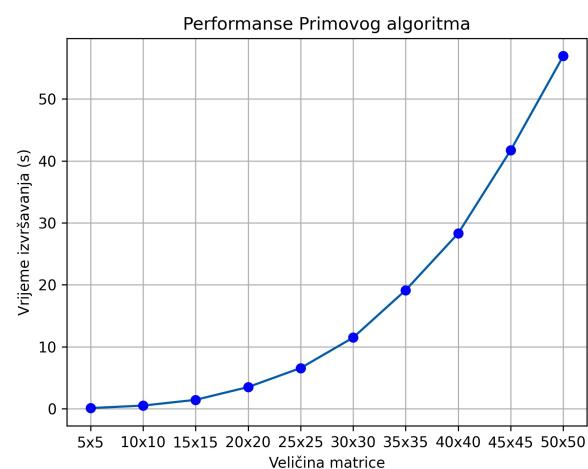
Slika 15: MST za evropske gradove

Kostur je formiran nakon što je izabrano 14 grana (označene crvenom bojom): (Sarajevo - Beč), (Sarajevo - Rim), (Beč - Berlin), (Berlin - Amsterdan), (Amsterdan - London), (London - Pariz), (London - Dablin), (Berlin - Štokholm), (Štokholm - Helsinki), (Štokholm - Oslo), (Helsinki - Minsk), (Minsk - Kiev), (Pariz - Madrid) i (Madrid - Lisabon). Traženo minimalno povezuće stablo ima ukupnu težinu: $496 + 523 + 524 + 580 + 358 + 343 + 464 + 811 + 397 + 416 + 715 + 433 + 1053 + 502 = 7616$. Dakle za povezivanje gradova je potrebno 7616 kilometara optičkog kabla.

Na isti način kao u prethodnom problemu, odredit će se ukupna maksimalna ušteda resursa. U ovom slučaju najveći bi utrošak bio od 14777 kilometara iz čega slijedi je pomoću Primovog algoritma ostvarena ušteda od 7161 kilometara optičkog kabla što je procentualno ($\sim 48\%$) uštede.

C. Performanse implementiranog algoritma

U ovom pottoplavlju je izvršeno testiranje performansi implementiranog Primovog algoritma nad različitim veličinama potpuno gustih grafova.



Slika 16: Prosječno vrijeme izvršavanja Primovog algoritma za različite veličine potpuno gustih grafova

Na x osi je predstavljeno prosječno vrijeme izvršavanja, a na y osi veličina matrice susjedstva. Testiranje je izvršeno nad veličinama od 5×5 do 50×50 , gdje se u svakom koraku veličina povećava za 5.

Kao rezultat testiranja, dobijena je funkcija za eksponencijalnim rastom. Ova verzija Primovog algoritma za pronađenje minimalnog povezujućeg stabla nad gustim grafovima dimenzija većih od 50×50 zahtjeva više od 1 minute. Eksponencijalni rast je očekivan s obzirom da se radi o osnovnoj verziji Primovog algoritma.

IV. ZAKLJUČAK

Problem rješavanja minimalnog povezujućeg stabla predstavlja jedan od najpopularnijih i najstarijih problema teorije grafova koji ima svoju prepoznatljivu i široku primjenu u praksi. Rješavanjem ovog problema se uspješno pronađe lista relacija koje je neophodno оформiti među objektima problemskog prostora da bi se ostvario minimalni utrošak resursa što rezultira postizanjem potrebnih ušteda.

Generalno za problem rješavanja minimalnog povezujućeg stabla postoji cijeli spektar metoda i algoritama koji omogućavaju uspješno rješavanje istog. Treba također imati u vidu da iako su razvijeni efikasni algoritmi, oni ne mogu imati optimalne performanse za grafove različitih gustoća. Tako da se primjena istih treba prilagoditi u skladu s gustoćom razmatranog grafa. Za implementacionu fazu je odabrana osnovna verzija Primovog algoritma koja je kroz simulacije dala optimalne rezultate. Osim Primovog algoritma, ostali često korišteni algoritmi iz ove oblasti su opisani u kratkim crtama.

Zadatak ovog rada je bila da se odabrani algoritam implementira u programskom jeziku Python i obavi detaljna analiza principa rada istog. Ova verzija Primovog algoritma relativno brzo pronađe rješenje za grafove manje gustoće, međutim kako raste gustoća grafa tako vrijeme izvršenja postaje lošije i doseže eksponencijalne nivo. Da bi se obavio proces simuliranja ovog algoritma osmišljena su dva praktična problema; problem povezivanja dalekovoda i problem povezivanja gradova optičkim kablom. Primjenom ovog algoritma nad praktičnim problemima je zamjećena značajna ušteda ($\sim 50\%$ za oba problema) što je i osnovni cilj ovog algoritma. Osim osnovne verzije Primovog algoritma postoje i poboljšane verzije koje koriste drugačije pristupe nalaženja grana najmanje težine. Jedan od često korištenih pristupa je korištenje prioritetnog reda. Ovaj pristup, odnosno način implementacije bi mogao smanjiti kompleknost algoritma na $\mathcal{O}(E \log_2 V)$ [2] i samim time imati bolje performanse nad većim gustim grafovima.

LITERATURA

- [1] Željko Jurić, Predavanja na predmetu Diskretna matematika, Elektrotehnički fakultet, Univerzitet u Sarajevu, ak. 2020/2021. godina.
- [2] Haris Šupić, Predavanja na predmetu Algoritmi i strukture podataka, Elektrotehnički fakultet, Univerzitet u Sarajevu, ak. 2020/2021. godina.
- [3] Marpaung, F & Piliang, Arnita. (2020). Comparative of prim's and boruvka's algorithm to solve minimum spanning tree problems. Journal of Physics: Conference Series. 1462. 012043. 10.1088/1742-6596/1462/1/012043.
- [4] Tadej Mateljan, Željko Jurić, Predavanja na predmetu Osnove operacionih istraživanja, Elektrotehnički fakultet, Univerzitet u Sarajevu, ak. 2020/2021. godina.