UNIVERZITET U BEOGRADU

Matematički fakultet

IVANA STAMENOV

REŠAVANJE PROBLEMA ODREĐIVANJA TRASA LINIJA GRADSKOG SAOBRAĆAJA PRIMENOM RAZNIH METAHEURISTIČKIH METODA

Master rad

Beograd, septembar 2016.

Mentor:	**Doc. dr Aleksandar Savić**
	Matematički fakultet u Beogradu
Članovi komisije:	**Prof. dr Zoran Stanić**
	Matematički fakultet u Beogradu
	Doc. dr Zorica Dražić
	Matematički fakultet u Beogradu

Datum odbrane: ____

Predgovor

U ovom radu razmatra se problem određivanja trasa linija gradskog saobraćaja primenom raznih metaheurističkih metoda.

Rad se sastoji od 4 poglavlja.

Uvodno poglavlje sadrži, pored osnovnih pojmova o problemu optimizacije na mrežama, informacije o egzaktnim i heurističkim metodama kao i informacije o vremenskoj složenosti korišćenih algoritama. Nakon toga, opisana je i postavka problema.

U drugom poglavlju, predstavljen je prvi matematički model za pokrivenost mreže sa jednim autobusom, i to za dva podmodela: kružna i linijska putanja. Dalje, u istom poglavlju se opisuju heurističke metode Pohlepnog algoritma i Genetskog algoritma. Prikazan je opis kodiranja svakog algoritma, kao i načina kako su realizovani svi genetski operatori u GA. Takođe u ovom poglavlju je nakon uspešnosti primene implementiranih algoritama na poznatim instancama, usledila hibridizacija GA i pohlepnog algoritma što je takođe opisano u ovom poglavlju. Zatim su prikazani ekperimentalni rezultati na osnovu kojih se može steći uvid u kvalitet predloženih metaheuristika. Dodatak drugom poglavlju je Turistička tura kroz gradove Srbije, čiji je cilj pokrenuti nove ideje po pitanju određivanja trasa i primeniti predložene metaheurističke metode na gradove naše države pa i šire.

U trećem poglavlju, predstavljen je drugi matematički problem za pokrivenost mreže sa dva ili više autobusa, što je i na primerima instanci dimenzija 5 i 10 čvorova (stanica) objašnjeno. Primenom implementiranog Genetskog algoritma prilagođenog za određivanje više trasa, predstavljeni su ekperimentalni rezultati za 3-8 linija gradskog saobraćaja na instancama dimenzija 30, 40 i 50 čvorova iz prvog matematičkog modela.

Zaključna razmatranja su izložena u četvrtom poglavlju, gde je dat osvrt na postignute rezultate, najznačajnije naučno-istraživačke doprinose, kao i ideje za dalja istraživanja i unapređivanje razmatrane problematike.

Želim da istaknem zahvalnost mentoru doc. dr Aleksandru Saviću na korisnim savetima, srdačnoj podršci, nesebičnoj pomoći i razumevanju tokom izrade ovog rada. Zahvaljujem se članovima komisije prof. dr Zoranu Staniću i doc. Dr Zorici Dražić na pažljivom čitanju rukopisa i konstruktivnim predlozima i savetima. Posebnu zahvalnost bih istakla članovima porodice koji su mi osim moralne podrške pružili i podršku pri tehničkoj izradi samog rada. Bratu Igoru, takođe dugujem veliku zahvalnost zato što je i uprskos velikoj prostornoj dislokaciji, u svakom trenutku bio spreman da me sasluša i posavetuje, a i pruži nesebičnu moralnu podršku. Rad posvećujem mojoj porodici, mojim najmilijima.

Beograd, 2016.

Kandidat
Ivana Stamenov

Rešavanje problema određivanja trasa linija gradskog saobraćaja primenom raznih metaheurističkih metoda

Rezime

Cilj ovog rada je konstruisanje i prilagođavanje metaheurističkih metoda genetskih i pohlepnih algoritama radi rešavanja NP-teških problema nalaženja optimalnih trasa linija gradskog saobraćaja. Instance su generisane tako da pokrivaju više slučajeva kako gustine saobraćajne mreže tako i različitih dužina trasa. Takođe, kvalitet metaheurističkih pristupa biće opravdan eksperimentalnim poređenjem sa tačnim rešenjima na instancama manjih dimenzija koja će biti određena metodom potpune pretrage. Sve ovo je ilustrovano i potkrepljeno određenim brojem primera i većih dimenzija koji prikazuju primenu ovih metoda u stvarnom životu, poređenih sa već postojećim odgovarajućim instancama. Pošto metode predstavljene u radu daju optimalna rešenja za manje, a veoma kvalitetna rešenja za instance većih dimenzija, moguća je primena metoda na mnoge postojeće slične probleme rutiranja.

Ključne reči: NP-teški problemi, Metaheurističke metode, Potpuno pretraživanje, Pohlepni algoritam, Genetski algoritam, Hibridizacija

Solving public transportation route problem with different metaheuristics methods

Abstract

The scope of this thesis is construction and adaptation of metaheuristic methods of genetic and greedy algorithms for solving the public transportation route problem, which has been proven to be NP-hard. Multiple usage scenarios, such as various transportation grid density and route length, will be covered in detail. Also, the quality of metaheuristic approaches will be discussed and compared with exact solutions for small size instances obtained through exhaustive search method. Additional solution examples for larger scale problems will be presented and compared with existing approaches for use in real-life situations. Since proposed method gives optimal solutions for small instances and close-to-optimal solutions for larger scale instances, in the end we conclude its feasibility in solving similar classes of existing routing problems.

Key words: NP-hard problems, Metaheuristic methods, Total Enumeration, Greedy algorithms, Genetic algorithms, Hibridization

Sadržaj

Spisak tabela:

Spisak slika:

1. *UVOD*

U današnje vreme razvoj i izgradnja saobraćajnih, energetskih, telekomunikacionih, računarskih i drugih mreža ne mogu se zamisliti bez unapred planiranog dobrog projekta. Kvalitet tog projekta, odnosno struktura mreže i efikasnost njenog funkcionisanja zavise od rešavanja odgovarajućih optimizacionih zadataka. U tu svrhu se preporučuje korišćenje pogodnih matematičkih modela optimizacije. Međutim, klasični modeli imaju jedan kriterijum i jednu funkciju cilja minimizacije ili maksimizacije u zavisnosti od problema, ali u praksi javlja se potreba i za modelima sa više kriterijuma, kako bi doneta odluka bila ispravnija.

Cilj ovog rada je odrediti dopustivu mrežu gradskog prevoza uz minimizaciju troškova (odnosno, pređenog puta), pri čemu je poznat broj autobusa i broj stanica sa datim međusobnim rastojanjima između svake od stanica. U ovom radu ograničavamo se na minimizaciju pređenog puta jednog ili više autobusa koji treba da posete sve stanice tačno jednom, što se u slučaju jednog autobusa svodi na Problem trgovačkog putnika (eng. *Travelling Salesman Problem*, skraćenica *TSP*). U prvom delu ćemo opisati osnovne pojmove potrebne za optimizaciju datog problema na mrežama koji će se koristiti u daljem radu, a zatim formiramo dva matematička modela. Prvi će se baviti pokrivenošću mreže sa jednim autobusom, a drugi pokrivenošću sa više autobusa uz obrazloženja za sva predložena ograničenja. Ideje algoritama koji se koriste za rešavanje datog optimizacionog problema biće potkrepljene primerima i eksperimentalnim rezultatima.

1.1. O problemu optimizacije na mrežama

U rešavanju problema određivanja trasa linija gradskog saobraćaja, odnosno problema optimizacije na mrežama koristimo *matematičke modele, teoriju grafova* i *algoritme*.

Matematički model sastoji se od ***funkcije cilja*** $f: X \to \mathbb{R}$ koja zavisi od upravljačkih promenljivih, ograničenja nad tim promenljivima koje opisuju skup dopustivih rešenja X. U našem slučaju X će biti diskretan skup. Cilj rešavanja modela je naći u skupu X dopustivih rešenja ono rešenje za koje funkcija cilja $f(x)$ dostiže optimalnu, u našem slučaju minimalnu vrednost:

$$\min_{x \in X} f(x).$$

Graf G je uređen par (V, E) gde je V konačan neprazan skup (čiji elementi se zovu ***čvorovi*** - *vertices,* a elementi skupa E nazivaju se ***grane*** - *edges*).

Graf se predstavlja geometrijskom figurom sastavljenom od tačaka i linija koje spajaju pojedine parove tačaka. Tačke predstavljaju čvorove grafa dok linije predstavljaju grane grafa. Grane mogu biti *neorijentisane* ili *orijentisane*. Za dva čvora grafa kažemo da su *susedna* ako su spojena granom. Niz grana koje se nadovezuju jedna na drugu bez ponavljanja grana u neorijentisanom grafu naziva se ***put***. ***Cikl*** je put koji se završava u istom čvoru u kojem počinje. Ako put ili cikl prolazi kroz svaki čvor najviše jednom, put ili cikl se naziva *elementaran (prost)*. ***Hamiltonov put*** je put u neorijentisanom grafu koji sadrži svaki čvor tačno jednom [Del07]. ***Težinski graf*** je graf u kojem je svakoj grani dodeljena neka težina, odnosno neki broj (dužina, propusna moć, kapacitet, cena, prenos…). ***Rastojanje čvorova*** u grafu se definiše kao dužina najkraćeg puta koji povezuje ta dva čvora.

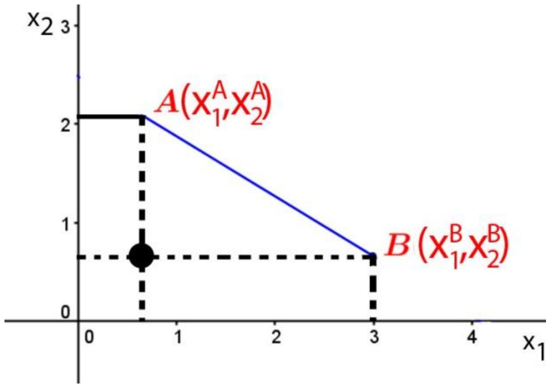
Neka su u opštem slučaju čvorovi mreže numerisani sa $1, 2, \dots, n$. Posmatrajmo *potpuni težinski graf G* u kojem je grani između dva čvora pridružena težina (dužina) s_{ij}. Od ove veličine se može formirati kvadratna matrica oblika

$$L = [s_{ij}]_{n \times n}.$$

Matrica $\boldsymbol{L}$ se zove *težinska matrica* ili ***matrica rastojanja***. Ova matrica formira mrežu za koju važi da je $\boldsymbol{s_{ij}} = \boldsymbol{s_{ji}}$ za svako $(i, j) \in E$. U modelima ćemo koristiti potpune grafove, jer nam oni garantuju povezanost. To znači da ukoliko u realnom problemu ne postoji ulica koja povezuje dve stanice označene sa i i j, onda se grani (i, j) u grafu G pridružuje beskonačno veliko rastojanje, takvo da posmatrana grana neće ući u optimalno rešenje.

Euklidsko rastojanje se koristi u Lokacijskim problemima za izračunavanje rastojanja između dve tačke (stanice) u posmatranom prostoru. Neka su u prostoru R^2 zadate dve tačke $A = (x_1^A, x_2^A)$ i $B = (x_1^B, x_2^B)$ rastojanje d je jednako

$$\boldsymbol{d(A, B)} = \sqrt{(x_1^B - x_1^A)^2 + (x_2^B - x_2^A)^2}.$$



Slika 1: Rastojanje između dve tačke u koordinatnom sistemu

1.2. NP-teški problemi i složenost algoritama

Algoritam za rešavanje nekog problema je niz postupaka čije izvršavanje dovodi do rešenja tog problema ili da problem nema rešenje. *Kompleksnost algoritma* se izražava vremenom rada algoritma, odnosno brojem elementarnih koraka unutar algoritma potrebnih za dobijanje rešenja postavljenog problema.

Vremenska složenost algoritma se obično definiše pomoću funkcije $f\colon \mathbb{N} \to \mathbb{N}$, gde je $f(n)$ ukupan broj osnovnih operacija koje su algoritmu potrebne za rešavanje problema ulazne veličine n. U praksi nam se najčešće javljaju ***polinomske i eksponencijalne vremenske složenosti*** pa samim tim razlikujemo polinomske i eksponencijalne algoritme. Više o složenosti algoritama iz oblasti kombinatorne optimizacije se može pogledati u [Brs88], [Grn72], [Pau97].

Veliki broj praktičnih problema ima eksponencijalnu složenost pa je njihova primena vremenski veoma zahtevna čak i na instancama vrlo malih dimenzija. Zbog ovoga pribegava se drugim metodama koje imaju polinomsku složenost, ali ne garantuju optimalnost rešenja i njih nazivamo heuristikama [Ree93]. Primeri nekih složenosti su dati u sledećoj Tabeli 1.

složenost algoritma	**logaritamska i polinomska**				**eksponencijalna**	
primeri algoritma	*Pretraga uređenog niza*	*Pretraga neuređenog niza*	*Sortiranje elemenata niza*	*Nalaženje najkraćeg puta*	*Potpuno pretraživanje*	*Gruba sila*
f(n) / ***n***	$\log_2 n$	n	$n \log_2 n$	n^2	2^n	$n!$
10	$0.003\ \mu s$	$0.01\ \mu s$	$0.033\ \mu s$	$0.1\ \mu s$	$1.024\ s$	$60.5\ s$
20	$0.004\ \mu s$	$0.02\ \mu s$	$0.086\ \mu s$	$0.4\ \mu s$	1×10^3 god	1×10^6 god
50	$0.006\ \mu s$	$0.05\ \mu s$	$0.282\ \mu s$	$0.05\ s$	∞	∞
100	$0.007\ \mu s$	$0.1\ \mu s$	$0.644\ \mu s$	$0.1\ s$	∞	∞
1000	$0.010\ \mu s$	$1\ \mu s$	$9.966\ \mu s$	$1\ s$	∞	∞

Tabela 1: Klase algoritama i njihova složenost kao i vremena izvršavanja na računaru Asus sa procesorom AMD Athlon™ II X2 250 3.00GHz sa 4 jezgra i 8GB RAM memorije

U klasu $\boldsymbol{P}$ spadaju problemi koji se efikasno rešavaju algoritmima sa polinomskom složenošću. Klasa ***NP****-kompletnih problema* za dato rešenje obezbeđuje optimalnost u polinomskom vremenu. ***NP-teški problemi*** predstavljaju probleme za čije rešavanje nisu poznati algoritmi čija se složenost može izraziti polinomskom funkcijom (npr. linearnom, kvadratnom, kubnom…). Prema definiciji, svaki algoritam koji nije polinomski smatra se eksponencijalnim.

Tu spadaju mnogi poznati optimizacioni problemi kao što su problemi koji se mogu opisati celobrojnim programiranjem, 0-1 programiranjem, problem trgovačkog putnika, lokacijski problemi i tako dalje. Više o NP-kompletnim i NP-teškim problemima u literaturi [Coo71], [Kra00], [Brs88], [Rad11].

1.3. Egzaktne metode

Modeliranje različitih problema iz prakse često dovodi do NP-teških problema, pa se egzaktne metode, zbog njihovog dugog vremenskog izvršavanja, retko koriste za njihovo rešavanje. ***Egzaktne metode*** su najpoželjnije, jer one garantuju optimalno rešenje, ali najčešće su vremenski veoma zahtevne na problemima velikih dimenzija, što je prikazano u Tabeli 2. U tim slučajevima pribegava se primeni heurističkih metoda.

❖ ***Metoda potpunog pretraživanja*** (eng. *Total Enumeration*)

Metoda totalne enumeracije (potpuno pretraživanje) generiše sve permutacije skupa $\{1,2,\dots n\}$. U našem problemu izračunava dužine svih Hamiltonovih puteva generisanih ovim permutacijama i predstavlja najkraći put kao konačno rešenje. Ovo je neefikasan algoritam, jer je broj permutacija $n!$ pa je složenost ovog algoritma $O(n!)$, što je prikazano u Tabeli 2.

Broj čvorova $\boldsymbol{n}$	**Broj kombinacija** $\boldsymbol{n!}$	**Broj trasa** $\frac{\boldsymbol{(n-1)!}}{\boldsymbol{2}}$	**Vreme izvršavanja** $\boldsymbol{O(n!)}$
10	3628800	181440	≈1min 30sec
11	39916800	1814400	≈18min 15sec
12	479001600	19958400	≈3h 40min
13	6227020800	239500800	≈2dana
14	87178291200	3113510400	≈17dana
15	1.3076744e+12	43589145600	≈175 dana
16	2.092279e+13	653837184000	≈5 godina
20	2.432902e+18	6.082255e+16	≈47565 godina

Tabela 2: Složenost algoritma potpunog pretraživanj u zavisnosti od broja čvorova (vremena data u tabeli su ukoliko bi se algoritam puštao na računaru Asus sa procesorom AMD Athlon™ II X2 250 3.00GHz sa 4 jezgra i 8GB RAM memorije)

U ovom radu se koristi egzaktna metoda Totalnog pretraživanja za određivanje optimalnih rešenja problema manjih dimenzija koja će kasnije biti upotrebljena za poređenje sa rešenjima dobijenim heurističkim metodama.

1.4. Heuristike i metaheuristike

Kako se u oblasti kombinatorne optimizacije nailazi na probleme koje uz pomoć egzaktnih metoda je nemoguće rešiti zbog velikog utroška vremena, pribegava se ranije pomenutim heuristikama. Heuristike ne garantuju nalaženje optimalnog rešenja problema. Heuristike daju dovoljno dobra rešenja koja su bliska optimalnom rešenju u nekom unapred zadatom smislu. Rešenja koja se dobijaju su često optimalna rešenja, ali se optimalnost ne može dokazati.

- ***Pohlepni algoritam*** (eng. Greedy algorithm)

Ovo je algoritam koji u svakom koraku bira lokalno najbolje rešenje kako bi stigao do globalnog optimuma. Mana ove metode je što za posledicu može imati preuranjenu konvergenciju. Pohlepni algoritmi su veoma dobri za brzu aproksimaciju optimalnog rešenja. Najkraće rastojanje koje želimo da pronađemo se postupno izgrađuje dodavanjem uvek najkraćeg mogućeg puta, najbližeg suseda (eng. Nearest Neighbour). Pri tom ne sme nastati kontura koja ima manji broj čvorova od broja gradova, niti se pojaviti čvor (stanica) sa stepenom većim od 2, tj. posetiti neki čvor više od jednom. Između ostalog treba paziti da se ista grana (put) ne doda više od jednom. Složenost pohlepnog algoritma je $O(n^2 \log_2 n)$ za problem TSP od n čvorova.

- ***Metaheuristike***

Od sredine 70-tih godina razvija se i primenjuje niz opštih heuristika koje daju opšta uputsva za rešavanje problema, nezavisno od strukture konkretnog problema i upravo te heuristike zovemo metaheuristike. Neke od poznatijih metaheuristika su Genetski algoritam (eng. *Genetic algorithms - GA*), Simulirano kaljenje (eng. *Simulated Annealing*), Tabu pretraživanje (eng. *Tabu search - TS*), Mravlja kolonija (eng. *Ant Colony*), Metoda promenljivih okolina (eng. *Variable Neighborhood Search - VNS*) i mnoge druge koje su tek u razvoju. Mi ćemo se u ovom radu zadržati na *Genetskom algoritmu (GA).*

GA je metoda optimizacije koja imitira proces genetske evolucije. Analogija evolucije kao prirodnog procesa i genetskog algoritma kao metode optimizacije, ogleda se u procesu selekcije i genetskim operatorima. Genetski algoritam je vrlo prirodan algoritam i najefikasniji za rešavanje problema sa malo ograničenja i relativno složenom funkcijom cilja kao što je u našem slučaju. Osnovni koncept GA se ogleda u postojanju živog sveta, odnosno uopšte nastanku živih organizama. Svaki organizam se sastoji od ćelija, dok svaka ćelija nosi određeni jedinstveni skup hromozoma, koji u sebi sadrži DNK (eng. DeoxyriboNucleic Acid) odnosno lanac koji nosi genetske informacije. U procesu reprodukcije genetski materijali roditelja se ukrštaju i formiraju novu jedinku sa po malo izmenjenim osobinama koje imaju oba roditelja uz prisustvo mutacije, odnosno male promene genetskog materijala. Pri tome kvalitet ili prilagođenost novonastale jedinke (u našem slučaju permutacija čvorova grafa (gradova)) se meri njenim prilagođavanjem i uspehom u životnoj okolini. Ovo i predstavlja osnovu za nastanak genetskih algoritama. Analogija između živog sveta i računarskog jeste ta da se upravo odabirom početne populacije jedinki (trasa) slučajnim izborom, kao i i daljim formiranjem novih generacija, prenosi evolucija prirodnog procesa na zadati problem određivanja trasa linije gradskog saobraćaja. U prirodi se smatra da jedinka koja je najbolje prilagođena okolini i uslovima života ima najveću verovatnoću opstanka i daljeg razmnožavanja, a samim tim i prenošenja svog genetskog materijala na svoje potomstvo. Nakon odabira populacije sledi proces selekcije što je moguće boljih jedinki, ukrštanja i mutacije koji se ponavlja sve dok se ne zadovolji neki unapred postavljen uslov ili postigne vreme kojim je proces ograničen. Na taj način dolazimo do jednog od mogućih rešenja datog problema. Nakon svakog ciklusa genetski materijal koji nose nove jedinke primenom genetskih operatora garantuje da najbolja jedinka nove generacije neće biti lošije prilagođena od prethodnih.

```
Unošenje_Ulaznih_Podataka();
Generisanje_Početne_Populacije();

while not Kriterijum_Zaustavljanja_GA()
for i=1 to Npop do
F[i] = Funkcija_Cilja(i);
end
Funkcija_Prilagođenosti();
Selekcija();
Ukrštanje();
Mutacija();
end

Štampanje_Izlaznih_Podataka();
```

Šema 1: Šema genetskog algoritma

Prost genetski algoritam (Šema 1) predstavlja osnovnu varijantu GA i sastoji se od selekcije, ukrštanja i mutacije, o čemu ćemo u sledećem poglavlju detaljnije. Najčešći problem u primeni prostog genetskog algoritma je preuranjena konvergencija.

John Koza je 1992. godine predložio upotrebu genetskog algoritma za razvoj računarskih programa [Koz92] za izvršavanje određenih zadataka i otvorio područje genetskog programiranja. Mogućnosti korišćenja genetskih algoritama u mašinskom učenju prikazane su u radu [Vug96].

Primena GA u oblasti operacionih istraživanja može se naći u [Dow96]. Dok u literaturi [Gol10] piše da se genetski algoritam primenjuje i daje dobre rezultate u području učenja o neuronskim mrežama, pri traženju najkraćeg puta, problemu trgovačkog putnika, strategiji igara, problemima sličnim transportnom problemu, problemu raspoređivanja, optimizacije upita nad bazom podataka, prilikom projektovanja komunikacijskih (putnih, vodovodnih, elektroenergetskih...) mreža, kao i za finansijske i ekonomske analize i planiranja [Rad11].

Pri rešavanju problema kombinatorne optimizacije treba imati u vidu činjenicu da ne postoji metoda za njihovo rešavanje koja je univerzalna i koja daje najbolje rezultate na svim problemima. Zato se često vrše kombinacije raznih metoda i dobijaju hibridni algoritmi [Cve96] . Tako ćemo i u ovom radu izvršiti hibridizaciju Pohlepnog i Genetskog algoritma (GA+), a eksperimentalne rezultate ćemo prikazati u sledećem poglavlju.

1.5. Postavka problema

Zadat nam je konačan skup stanica

$$\mathcal{S} = \{\mathrm{S}_1, \mathrm{S}_2, \mathrm{S}_3, \dots, \mathrm{S}_\mathrm{n}\}$$

i matrica rastojanja između susednih stanica koju ćemo označiti sa

$$L = \left[s_{ij}\right]_{n \times n}$$

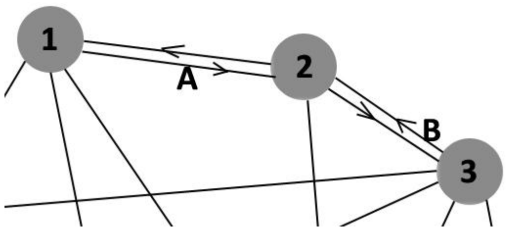
$$s_{ij} = d\left(S_i - S_j\right), \qquad i, j = 1, \dots n$$

$$s_{ij} = s_{ji}$$

$$s_{ij} = 0, i = j$$

$$L = \begin{bmatrix} 0 & \cdots & s_{1j} \\ \vdots & \ddots & \vdots \\ s_{i1} & \cdots & 0 \end{bmatrix}$$

Posmatrajmo mrežu $G = (S, s_{ij}), i, j = 1, \ldots n$, gde $\mathcal{S} = \{\mathrm{S}_1, \ldots, \mathrm{S}_\mathrm{n}\}$ predstavlja skup čvorova (stanica), a s_{ij} (puteve) težine grana koji povezuju date (stanice) čvorove. Grafički gradsku mrežu možemo predstaviti pomoću opisanog grafa na osnovu zadate matrice rastojanja L koja će u našem slučaju biti simetrična matrica i obrazovati težinski graf. Svi putevi (grane) biće dvosmerni (neorijentisane) (Slika 2). Pod tim podrazumevamo da su stanice u smeru A i u smeru B na istom rastojanju samo na suprotnim stranama ulice.



Slika 2: Putevi su dvosmerni

Zbog toga i imamo simetričnu matricu kako ne bismo dodatno komplikovali zadati problem sa asimetričnošću i jednosmernim ulicama. Međutim, pridružićemo i mogućnost zadavanja matrice pomoću 2-D euklidskih koordinata, odnosno pomoću longituda g_n i latituda l_n matricom dimenzije $n \times 3$ u sledećem obliku:

$$\begin{bmatrix} 1 & g_1 & l_1 \\ 2 & g_2 & l_2 \\ \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots \\ n-1 & g_{n-1} & l_{n-1} \\ n & g_n & l_n \end{bmatrix}_{n \times 3}.$$

Prilikom zadavanja matrice u euklidskom obliku, algoritam (ugrađena funkcija, *distmat*(X)) u programskom paketu Matlab2010b pomoću koordinata skicira grafik položaja tačaka i formira novu simetričnu matricu rastojanja D, pomenutu na početku postavke problema (Šema 2).

```
function D = distmat(X)
%DISTMAT Određuje matricu rastojanja na osnovu zadatih 2-D koordinata
O = load('euklids.txt');
[h,dim] = size(O);
X=O(1:h,2:3);
[n,dim] = size(X);
D = zeros(n);
for j = 1:n
    for k = 1:dim
        v = X(:,k) - X(j,k);
        D(:,j) = D(:,j) + v.*v;
    end
end
D = sqrt(D);
```

Šema 2: Matlab funkcija distmat() za određivanje simetrične matrice rastojanja D

Potrebna je pokrivenost čitave mreže puteva autobusima i mogućnost da putnik iz svake stanice S_i može stići u svaku stanicu S_j, $i, j = 1, \dots n$, nakon konačnog broja presedanja.

Dato je n *stanica* uz ograničenje da postoji više od 2 stanice kako bi problem rutiranja i kružnih linija imao smisla. Autobusi treba da obiđu sve stanice tako da pređu najkraći put. Pri tome imaju dve mogućnosti: da se vrate u polaznu stanicu ili ne. Kako bismo tačno definisali ograničenja i probleme pristupimo predstavljanju prvog modela.

2. MATEMATIČKI MODEL I

Modelirajmo problem rutiranja jednog autobusa tako da obezbedimo pokrivenost celokupne gradske mreže G, odnosno posetu svih n stanica uz minimalne troškove. Pošto zahtevamo da kroz svaku stanicu autobus prođe tačno jednom, ovaj problem se svodi na nalaženje minimalne Hamiltonove konture (puta). Ovakav problem rutiranja mreže jednim autobusom možemo posmatrati kao problem trgovačkog putnika (*Travelling Salesman Problem – TSP*), jednom od najpoznatijih i najproučavanijih problema kombinatorne optimizacije. Pošto postoje dve varijante TSP:

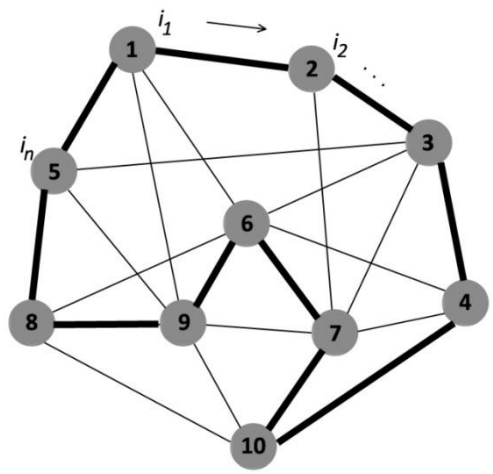
a) autobus se vraća u polaznu stanicu posle obilaska svih stanica (*kružna putanja*)
b) početna i završna stanica su različite (*linijska putanja*)

podelićemo *matematički model I* na ova dva podmodela.

2.1. Podmodel a - Kružna putanja

Autobus posle obilaska svih stanica vraća se u polaznu stanicu i tako obrazuje Hamiltonovu konturu H (Slika 3) što ćemo matematički zapisati na sledeći način:

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



Slika 3: Skica Hamiltonove konture za 10 čvorova zadatih matricom rastojanja

Smatraćemo da se za polaznu stanicu može uzeti proizvoljna stanica iz sledećeg skupa

$$\mathcal{S} = \{1,2,3 \dots n\}.$$

Upravo ovako definisan TSP je jedan od zadataka (0,1) celobrojnog linearnog programiranja čiji matematički model formulišemo na sledeći način:

$$(1) \qquad \boldsymbol{min} \sum_{i=1}^{n} \sum_{j=1}^{n} \boldsymbol{s_{ij} x_{ij}}$$

p.o.

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

$$(3) \qquad \sum_{\substack{i=1 \\ i \neq j}}^{n} x_{ij} = 1\,, \qquad j \in \mathcal{S}$$

$$(4) \qquad \sum_{i \in \mathcal{R}} \sum_{j \in \mathcal{R}} x_{ij} \leq |\mathcal{R}| - 1, \qquad \mathcal{R} \subset \mathcal{S}, \mathcal{R} \neq \emptyset, \quad \mathcal{R} \neq \mathcal{S}$$

$$(5) \qquad x_{ij} \in \{0,1\}, \qquad i,j \in \mathcal{S}$$

$[s_{ij}]_{n \times n}$ – simetrična matrica rastojanja (težine grana) između stanica i i j, pri čemu je $s_{ij} = \infty$ ako nema direktnog puta od i do j , gde su $i,j \in \mathcal{S}$

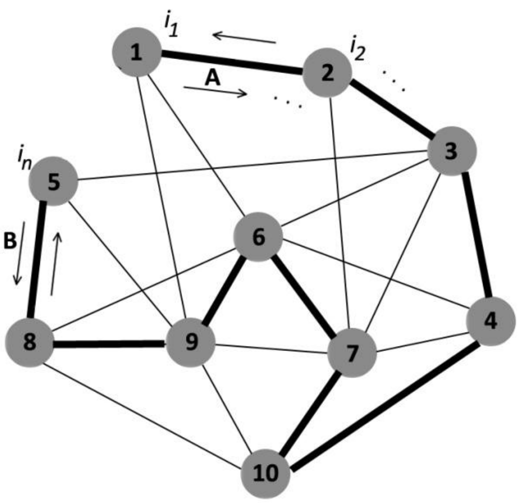
$$x_{ij} = \begin{cases} 1, & ako\ autobus\ prolazi\ granom\ (i,j) \\ 0, & inače \end{cases}, i,j \in \mathcal{S}$$

Promenljive $x_{ij}, i,j \in \mathcal{S}$ su binarne promenljive i ukoliko grana (i,j) ne pripada Hamiltonovoj konturi H tada je $x_{ij} = 0$, a ukoliko pripada konturi onda je $x_{ij} = 1$. Našu funkciju cilja smo označili sa (1), pri čemu njom minimiziramo ukupni pređeni put autobusa. Ograničenje (2) obezbeđuje da autobus može izaći iz svake stanice tačno jednom, dok ograničenje (3) obezbeđuje da autobus može ući u svaku stanicu tačno jednom, čime se obezbeđuje poseta svake stanice tačno jednom. Zatim ograničenje (4) eliminiše mogućnost da se u rešenju pojavi više od jedne kružne putanje i obezbeđuje da autobus obiđe svih n stanica.

2.2. Podmodel b – Linijska putanja

U ovom podmodelu prvog predloženog modela smatraćemo da se autobus u smeru A posle obilaska svih stanica ne vraća u polaznu stanicu već ostaje u poslednjoj posećenoj, i istom rutom se u smeru B vraća nazad do polazne stanice, jer smo već definisali da su sve ulice dvosmerne. Tako se sada ne obrazuje Hamiltonova kontura, već Hamiltonov put H (Slika 4) što ćemo matematički zapisati na sledeći način:

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



Slika 4: Skica Hamiltonovog puta

Ovo je jedino ograničenje koje ćemo promeniti u odnosu na algoritam rešavanja podmodela **a** dok sva ostala ograničenja zadržavamo zajedno sa funkcijom cilja. Pošto je ovo vrlo prirodna mogućnost u realnom životu i učestala pojava na ulicama našeg grada da autobusi kojim putem idu od stanice 1 do stanice n u smeru A, istom rutom se vraćaju samo u smeru B. Ovakavo rezonovanje je od koristi pri definisanju II matematičkog modela za rutiranje više od jednog autobusa u zadatoj gradskoj mreži, no o tome ćemo detaljnije nešto kasnije. Sada ćemo izložiti dobijene eksperimentalne rezultate za ovako definisan model upotrebom ranije opisanih algoritama uz proizvoljan izbor početne stanice što se takođe često sreće u gradskom prevozu. Unapred su određene polazne stanice odnosno u žargonu rečeno postoje "okretaljke".

2.3. Heurističke metode za određivanje trase linije gradskog saobraćaja

Prethodno definisan problem TSP pripada teško rešivim problemima u smislu što su svi poznati algoritmi za njegovo rešavanje eksponencijalne složenosti. Iscrpnom pretragom mreže moguće je u konačno mnogo koraka pronaći traženu konturu, ali u praksi je podjednako važno vreme u kojem se to postiže, pa su nam zato i potrebni brzi i efikasni algoritmi. Heurističke metode za rešavanje TSP u ovom radu delimo na:

- **Pohlepni algoritam**
- **Genetski algoritam**

2.3.1. Pohlepni algoritam

Pohlepni algoritam se sastoji iz sledeća dva koraka:

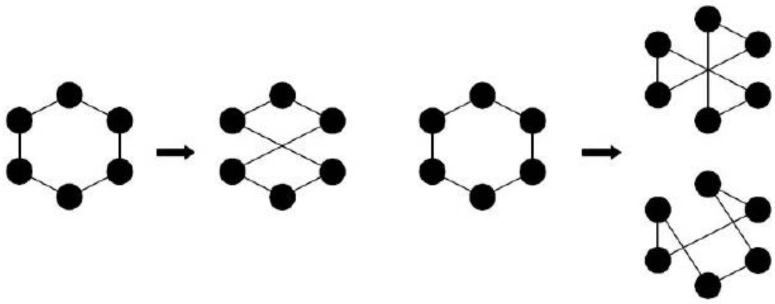
- ***Korak 1*** *(Nalaženje početnog rešenja):*

  Efikasno pronalaženje početnog rešenja algoritmom najbližeg suseda. Postupak pronalaženja početnog rešenja svodi se na pojednostavljenju matematičkog modela TSP koji smo definisali, tako što se originalan zadatak zamenjuje jednostavnijim problemom raspoređivanja. Ako se za rešenje tada dobije jedna kontura onda će to baš i biti naše rešenje, naša Hamiltonova kontura (put), a ukoliko nije pristupamo *Koraku 2,* metodama za poboljšanje predloženog rešenja.

- ***Korak 2*** *(Metode za poboljšanje postojećeg rešenja):*

  k-optimalne heuristike

  2-opt, 3-opt ili k-opt algoritam bira proizvoljno *2* (*3* ili *k*) nesusedne grane u postojećoj konturi, uklanja ih i ponovo spaja sa 2 (*3* ili *k*) novo nastala puta (Slika 5). Spajanje se vrši tako da se zadrže ograničenja, a dobijena kontura se dalje koristi samo ako je kraća od prethodne. Postupak se ponavlja sve dok postoje poboljšana rešenja, inače se algoritam zaustavlja. Ovaj algoritam daje dobre rezultate i ima polinomsku složenost.



Slika 5: Primer 2-opt i 3-opt obilaska konture

2.3.2. *Genetski algoritam*

Genetski algoritam (GA) je sam po sebi prilagodljiva metoda, jer simulira mehanizam prirodne evolucije i pomoću njega se rešavaju mnogi problemi kombinatorne optimizacije. Prvi radovi su nastali još 60-ih godina prošlog veka, i to su principi zasnovani na Darvinovoj teoriji o postanku živog sveta [Dar59]. GA je metoda optimizacijskog problema u računarstvu u cilju pronalska tačnog ili približnog rešenja. Više o GA se može naći u [Bok87], [Gol89], [Dav91], [BeD93a], [BeD93b].

2.3.2.1. *Definisanje jedinki i populacije*

Za početak je potrebno generisati konačni skup jedinki, odnosno populaciju za koju se najčešće uzima između 10 i 200 jedinki. Jedinka inače predstavlja jedno od mogućih rešenja datog problema, a zadata je posebnim genetskim kodom. Osvrnimo se i na reprezentaciju jedinki koja može biti zadata binarnim ili prirodnim brojevima [Ant89], [Bea93], [Gol89], pa čak i pomoću alfabeta ukoliko je u pitanju reprezentacija permutacijama (Tabela 3).

jedinke	reprezentacije									
binarna	1	1	0	1	0	0	1	1	0	1
prirodna	5	9	10	2	3	6	7	8	4	1
alfabet	A	B	C	D	E	F	G	H	I	J

Tabela 3: Razne reprezentacije jedinki na primeru od 10 stanica

U običnom genetskom algoritmu početnu jedinku zadajemo na slučajan način i to random metodom određivanja trasa (čvorova) koje autobus treba posetiti (Tabela 4).

jedinke	Trase linija									
jedinka 1	1	2	3	4	5	6	7	8	9	10
jedinka 2	3	4	7	5	1	8	6	10	2	9
...	...	...	...	...	...	...	...	...	...	...
jedinka *popSize*	5	9	10	2	3	6	7	8	4	1

Tabela 4: Implementacija populacije dimenzije popSize slučajnim izborom na primeru od 10 stanica

Radi poboljšanja genetskog algoritma, jedna jedinka u populaciji se dobija na osnovu pohlepnog algoritma (Tabela 5) što povećava brzinu dobijanja rešenja, a zatim se ostatak populacije formira nizom slučajno određenih jedinki. Veličinu populacije (*popSize*) je moguće zadavati proizvoljno po pokretanju programa u Matlab-u.

jedinke	Trase linija									
jedinka 1	1	5	8	10	9	6	7	4	3	2
jedinka 2	3	4	7	1	5	8	6	10	9	2
...	...	...	...	...	...	...	...	...	...	...
jedinka *12*	5	9	10	2	3	6	7	8	4	1

Tabela 5: Implementacija populacije dimenzije popSize=12 pomoću pohlepnog algoritma za 10 stanica

Broj jedinki u populaciji bi trebalo da bude dovoljno veliki da omogući pravljenje što raznovrsnije populacije, a sa druge strane ne suviše veliki zbog uštede u pogledu vremena. U eksperimentalnim rezultatima koristimo populacije od 10, 100 i 150 jedinki čija rešenja upoređujemo, što će i detaljno biti prikazano u narednom poglavlju.

Prilikom implementacije genetskog algoritma koristi se najčešće jedan od naredna tri tipa zamene generacija, a to su:

- *Stacionarni tip* - zamenjuje se samo deo populacije dok se preostale jedinke prenose u narednu generaciju
- *Elitistički tip* – propušta se određeni broj elitnih (najboljih) jedinki u narednu generaciju bez primene genetskih operatora
- *Generacijski tip* – u svakoj narednoj generaciji se menjaju sve jedinke.

U daljem radu u implementaciji problema određivanja trasa linija gradskog saobraćaja primenjuje se stacionarni tip sa elitističkom strategijom pri čemu se trenutno najbolje jedinke (i to njih $popSize/4$) prenose u narednu generaciju, dok se ostale menjaju pomoću genetskih operatora selekcije i mutacija. Dakle, četvrtina populacije najbolje prilagođenih jedinki ostaje nepromenjena, odnosno direktno se propušta u sledeću generaciju, dok se tri četvrtine populacije zamenjuju u svakoj narednoj generaciji primenom operatora selekcije i mutacije. Ovakav način pristupa organizovanja generacija obezbeđuje čuvanje dobrih rešenja uz dodatnu uštedu vremena. Primetimo da se operator ukrštanja direktno ne koristi, jer standardni operatori ukrštanja te permutacije neće dati permutaciju.

2.3.2.2. Funkcija prilagođenosti

Funkcija prilagođenosti u problemu određivanja trasa linija gradskog saobraćaja je funkcija cilja našeg problema.

Funkcija prilagođenosti je naša funkcija cilja i početna prilagođenost svake jedinke je beskonačno velika. Svakoj jedinki se pridružuje funkcija prilagođenosti koja ima zadatak da ocenjuje kvalitet jedinke. Genetski algoritam, uzastopnom primenom operatora selekcije i mutacije, obezbeđuje da se iz generacije u generaciju poboljšava apsolutna prilagođenost svake jedinke u populaciji, a time i srednja prilagođenost celokupne populacije. Ovim mehanizmom se dobijaju sve bolja rešenja datog konkretnog problema. Selekcija nagrađuje natprosečno prilagođene jedinke tako što one dobijaju veću šansu za reprodukciju pri formiranju nove generacije. Sa druge strane, slabije prilagođenim jedinkama se smanjuju šanse za reprodukciju pa one postepeno izumiru.

U literaturi se srećemo sa različitim načinom računanja funkcije prilagođenosti kao što su direktno i linearno skaliranje, skaliranje u jedinični interval i sigma odsecanje [Mar08]. Izbor odgovarajućeg načina najviše zavisi od specifičnosti problema, a često je neophodno i kombinovanje različitih principa. Mi ćemo se zadržati na određivanju funkcije prilagođenosti, Tabela 6.

Funkcija prilagođenosti	Trase linija									
3450	1	2	3	4	7	10	9	6	8	5
Inf	4	9	3	7	1	8	10	2	6	5
Inf	4	6	3	5	8	2	7	10	1	9
Inf	10	8	6	4	5	2	3	7	9	1

Tabela 6: Implementacija populacije i funkcije prilagođenosti pomoću pohlepnog algoritma za 10 stanica

2.3.2.3. Operator selekcije

Operator selekcije bira jedinke iz trenutne populacije koje će ostaviti potomke za sledeću generaciju i obično se definiše tako da one jedinke, koje imaju bolju prilagođenost, imaju i veće šanse za prelazak u sledeću generaciju. Osnovne metode selekcije su prosta rulet selekcija, selekcija zasnovana na rangu, turnirska selekcija i selekcija primenom ostataka.

Nedostatak rulet selekcije je preuranjena konvergencija, jer jedinke sa boljom funkcijom prilagođenosti imaju veće šanse da budu izabrane, a pri tom ostale jedinke otpadaju iz populacije. Selekcija zasnovana na rangu je pogodna za prevazilaženje anomalija proste selekcije, jer funkcija prilagođenosti jedinki u ovoj selekciji zavisi samo od poretka jedinki, a ne od konkretne vrednosti, pa se samim tim i preuranjena konvergencija retko javlja.

Za rešavanje problema u ovom radu je korišćena turnirska selekcija, kod koje se populacija na slučajan način deli u grupe od po *4* jedinki, koje se zatim takmiče radi preživljavanja i prelaska u narednu generaciju. Pobednik je ona jedinka sa najboljom funkcijom prilagođenosti u datoj grupi. Broj turnira je jednak ukupnom broju jedinki u populaciji umanjenih za broj elitnih jedinki. Metoda selekcije primenom ostataka bazira na prostoj selekciji koja i dalje poseduje velike nedostatke po pitanju preuranjene konvergencije. Više o tipovima selekcija u literaturi [Kra00], [Fil98].

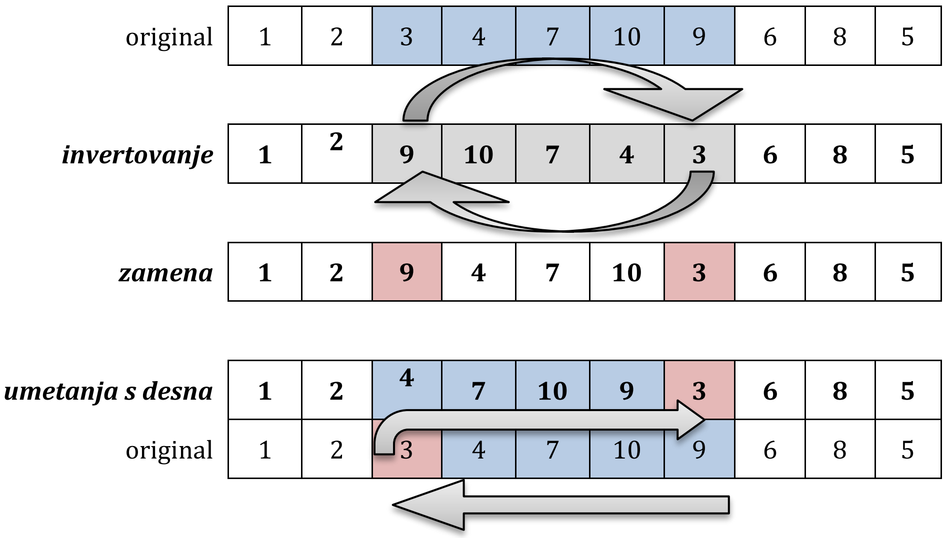
2.3.2.4. Operatori mutacije

Mutacija je zajedničko ime za sve operatore koji iz samo jedne jedinke odnosno od jednog roditelja formira samo jednu jedinku, uz pomoć određenih promena. Ukoliko je jedinka predstavljena prirodnim brojevima postoje dva osnovna oblika mutacije i to su slučajna mutacija i mutacija pomeraja. Slučajna mutacija je tip mutacije koji slučajno bira dve vrednosti iz skupa dozvoljenih vrednosti (stanica, npr. 1 2 3 4 5 6 7 8 9 10) i zamenjuje ih., što je prikazano na slici 6 sa stanicom broj 3 i 9. Mutacija je primenjivana na 4 najbolje neelitne jedinke.

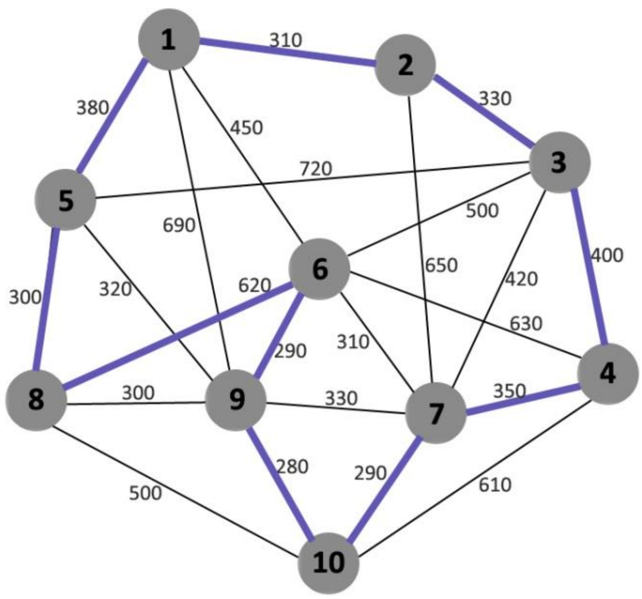
Mutacija invertovanjem funkcioniše na sledeći način: nasumično odaberemo 2 pozicije u genu i zamenimo poredak vrednosti između njih. Primer mutacije inverzijom prikazan je na slici 6, između pozicija 3 i 9.

Mutacija zamenom je tip mutacije kada slučajno biramo dve pozicije u genu i zamenimo njihove vrednosti.

Mutacija umetanja s desna, je tip mutacije takav da odaberemo dve pozicije, i prvi član u nizu izbacimo dok ostale pomerimo u levo za to jedno mesto sve do druge odabrane pozicije, a zatim vratimo izbačeni element na upravo poslednju poziciju, slika 6.



Slika 6: Mutacija najbolje prilagođene jedinke na 10 stanica



Slika 7: Skica gradske mreže G od 10 stanica sa jednom trasom (original sa slike 6)

2.3.2.5. *Kriterijum zaustavljanja*

Kriterijumi zaustavljana koji se najčešće primenjuju u toku izvršavanja GA su:

- postizanje optimalnog rešenja ukoliko je ono unapred poznato
- dostignut je maksimalan broj generacija
- pojava velikog broja sličnih jedinki u populaciji

- dokazana je optimalnost najbolje jedinke ukoliko je to moguće
- postignuto je vremensko ograničenje za izvršavanje GA
- prekidanje izvršavanja GA od strane korisnika

Ukoliko je ispunjen unapred zadati kriterijum zaustavljanja GA se završava i štampa se izveštaj sa postignutim rešenjem datog problema, odnosno najboljom jedinkom u poslednoj generaciji. U ovom radu je korišćeno kombinovanje dva kriterijuma i to maksimalni broj iteracija od 10000 generacija i ukoliko je broj istih jedinki u populaciji jednak četvrtini populacije.

2.3.3. Hibridizacija genetskog i pohlepnog algoritma GA+

Iscrpnom pretragom mreže moguće je u konačno mnogo koraka pronaći traženu konturu, ali u praksi je podjednako važno vreme u kojem se to postiže, pa su nam zato i potrebni brzi i efikasni algoritmi.

Metode za rešavanje TSP koje ćemo koristiti u ovom radu su upravo opisane GA i pohlepni algoritam, i dodatna pogodnost hibridizacije ove dve metode uz korišćenje dobrih strana od svake heuristike. Hibridizacija je urađena tako što je svaka polazna jedinka u populaciji izabrana pomoću pohlepnog algoritma.

2.3.4. Dobre strane genetskog algoritma

Dobre strane genetskog algoritma su:

- primenljiv je na velikom broju problema
- struktura algoritma nudi velike mogućnosti nadogradnje i povećanja efikasnosti algoritma
- jednostavnim ponavljanjem postupka se može povećati pouzdanost rezultata
- ako već ne nađe rešenje (globalni optimum), daje neko dobro rešenje
- kao rezultat daje skup rešenja, a ne jedno rešenje
- rešava sve probleme koji se mogu predstaviti kao optimizacioni, bez obzira da li funkcija koju treba optimizovati ima za argumente realne brojeve ili bitove ili znakove ili bilo koju vrstu informacije
- vrlo jednostavno je primenljiv na višedimenzionalne probleme
- jednostavnost ideje i dostupnost programske podrške

2.3.5. *Loše strane genetskog algoritma*

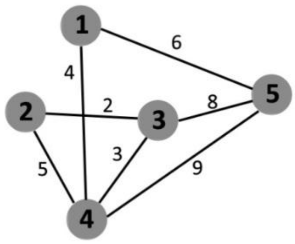
Lošee strane genetskog algoritma su:

- teško se definiše dobra funkcija prilagođenosti
- potrebno je prilagoditi GA zadatim ograničenjima
- često je potrebno prilagoditi problem algoritmu
- konvergencija je zavisna od vrste problema
- potrebno je posebno definisati genetske operatore za posebne vrste prikaza
- zbog stohastičnosti nikad ne znamo prirodu nađenog rešenja
- zbog izvođenja velikog broja računskih operacija GA, traži se velika procesorska snaga

Više o osobinama genetskog algoritma se može pročitati u literaturi [Jak96] [Gol10].

2.4. *Eksperimentalni rezultati*

Neka su nam zadate gradske mreže G sa po 5 i 10 stanica (Slike 8 i 9), gde su čvorovi sa vrednostima od 1 do 5, odnosno 10, zadate stanice, a vrednosti na granama kojima su povezani čvorovi predstavljaju težine grana. Težinama grana (Slika 8) odgovara vreme za koje autobus pređe određeni put u npr. minutama, dok je (Slika 9) rastojanje između stanica zadato u metrima. U odgovarajućim simetričnim matricama (Slike 8 i 9) ukoliko stanice nisu direktno povezane, njihove težine obeležavamo proizvoljno velikom vrednošću, kako bismo obezbedili sve potrebne i dovoljne uslove za rešavanje našeg problema (objašnjeno u postavci problema). Dakle, smatramo ta rastojanja beskonačnim ∞ (inf), jer zbog korišćenja programskog paketa Matlab2010b moramo prilagoditi naš problem ovom programskom jeziku.



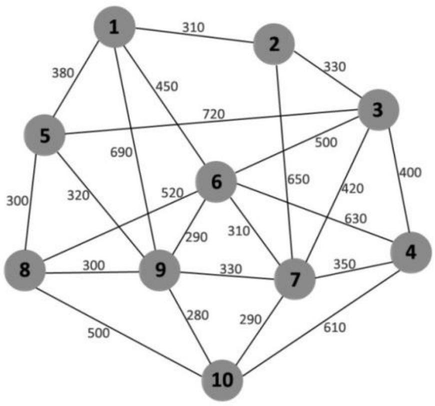

```
>> M=[inf,inf,inf,4,6;inf,inf,2,5,inf;inf,2,inf,3,8;4,5,3,inf,9

M =

   Inf   Inf   Inf     4     6
   Inf   Inf     2     5   Inf
   Inf     2   Inf     3     8
     4     5     3   Inf     9
     6   Inf     8     9   Inf

>> 
```

Slika 8: Skica gradske mreže G od 5 stanica i implementacija matrice rastojanja u MatLabu




```
>> D=[inf,310,inf,inf,380,450,inf,inf,690,inf; 310,inf,330,inf,in

D =

   Inf   310   Inf   Inf   380   450   Inf   Inf   690   Inf
   310   Inf   330   Inf   Inf   Inf   650   Inf   Inf   Inf
   Inf   330   Inf   400   720   500   420   Inf   Inf   Inf
   Inf   Inf   400   Inf   Inf   630   350   Inf   Inf   610
   380   Inf   720   Inf   Inf   Inf   Inf   300   320   Inf
   450   Inf   500   630   Inf   Inf   310   520   290   Inf
   Inf   650   420   350   Inf   310   Inf   Inf   330   290
   Inf   Inf   Inf   Inf   300   520   Inf   Inf   300   500
   690   Inf   Inf   Inf   320   290   330   300   Inf   280
   Inf   Inf   Inf   610   Inf   Inf   290   500   280   Inf
```

Slika 9: Skica gradske mreže G od 10 stanica i implementacija matrice rastojanja u MatLabu

Zadatak nam je da odredimo najkraći put za jedan autobus tako da on poseti sve stanice tačno jednom i da se omogući vraćanje autobusa u polaznu stanicu, odnosno rešavamo predloženi *matematički model I* uz poštovanje zadatih ograničenja.

2.4.1. Rezultati podmodela a

Sledi upoređivanje rezultata na osnovu predloženih algoritama za rešavanje zadatog problema u programskom paketu Matlab2010b koji su testirani na računaru Asus sa procesorom AMD Athlon™ II X2 250 3.00GHz sa 4 jezgra i 8GB RAM memorije. Instance koje su korišćene za testiranje predloženih algoritama su instance [instance1], [instance2], [instance3], [instance4] dostupne na internetu i to dva tipa: prvi tip sadrži instance koje su zadate pomoću simetričnih matrica, dok je drugi tip zadat pomoću geografskih koordinata, odnosno, matrica dimenzija $3 \times n$, gde n predstavlja broj čvorova (stanica).

Postojeći algoritmi su modifikovani i prilagođeni konstruisanom modelu, pri čemu imamo četiri tipa algoritama čije rezultate upoređujemo, a njihove ulazne i izlazne vrednosti su:

❖ ***Potpuno pretraživanje***

Metoda ranije opisana koja je implementovana u programskom paketu MatLab, sastoji se od sledećih ulaznih i izlaznih podataka:

→ ***Ulaz:***

- Simetrična matrica rastojanja (vremena)

Izlaz:→

- ispisuje se redosled poseta stanica (čvorova) mreže od strane jednog autobusa
- minimalno rastojanje
- vreme izvršavanja algoritma

❖ ***Heuristika - Heuristika( )***

Početno rešenje je pronađeno metodom pohlepnog algoritma, a zatim je primenjena metoda 2-opt radi poboljšanja rešenja. Napomenimo da se u kôd-u implementriranom u MatLab-u ulazne vrednosti, tj. matrica rastojanja učitava direktno iz **.txt* datoteke sačuvanog u folderu u kojem se nalazi i odgovarajući **.m* file, u našem slučaju *tsp_heuristic_circle(start)* koju pozivamo u Command Window-u i imamo mogućnost izbora polazne(start) stanice.

→***Ulaz:***

- polazna stanica

Izlaz:→

- ispisuje se redosled poseta stanica (čvorova) mreže od strane jednog autobusa
- minimalno rastojanje
- vreme izvršavanja algoritma

❖ ***Metaheuristika – Genetski algoritam(GA)***

Početna populacija se određuje na slučajan način, a na dalje, primenom genetskog algoritma postigli poboljšano rešenje.

→***Ulaz:***

- polazna stanica, dimenzija populacije za GA

Izlaz:→

- ispisuje se redosled poseta stanica (čvorova) mreže od strane jednog autobusa
- minimalno rastojanje
- vreme izvršavanja algoritma

❖ ***Hibridizacija Metaheuristika – (GA)+Greedy***

Početnu jedinku smo dobili na osnovu pohlepnog algoritma, a zatim formiramo ostatak populacije na slučajan način, i na dalje primenom genetskog algoritma postižemo poboljšano rešenje.

→***Ulaz:***

- polazna stanica, dimenzija populacije za GA+

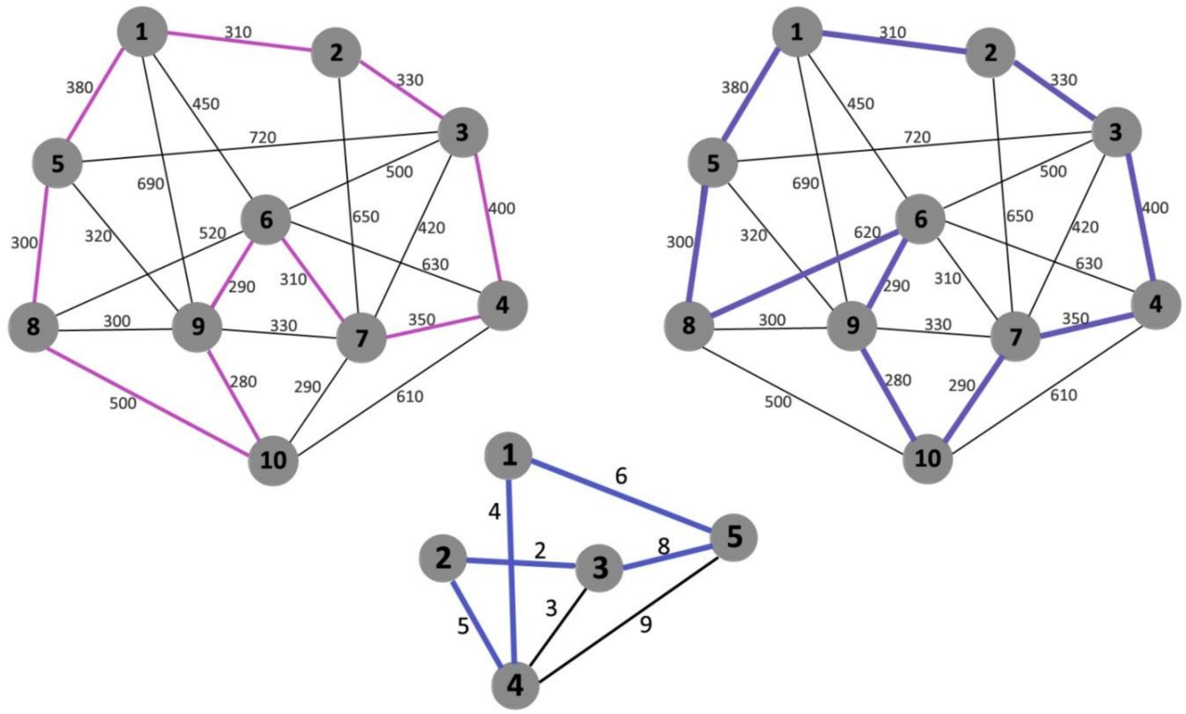
Izlaz:→

- ispisuje se redosled poseta stanica (čvorova) mreže od strane jednog autobusa
- minimalno rastojanje
- vreme izvršavanja algoritma

Za navedeni primer sa 5 i 10 stanica predstavili smo dobijena optimalna rešenja pomoću grafova i uporedili dobijene rezultate korišćenjem opisanih metoda (Tabela 7, Slika 10). Primetimo da je vreme izvršavanja algoritma potpunog pretraživanja mnogo veće od vremena izvršavanja ostalih metoda, pri čemu pokazujemo ranije navedeno da egzaktne metode nisu adekvatne za rešavanje ovakvih problema.

n=5 stanica, k=1 autobus			
	Potpuno pretraživanje	*Heuristika*	*Genetski Algoritam*
Vreme izvršavanja	≈0,02 sec	≈0.008 sec	≈6 sec
Dopustiva ruta	1-4-2-3-5-1	3-2-4-1-5-3	4-2-3-5-1-4
Minimalna ruta (min)	25	25	25
n=10 stanica, k=1 autobus			
	Potpuno pretraživanje	*Heuristika*	*Genetski Algoritam*
Vreme izvršavanja	≈100 sec	≈0.02 sec	≈6 sec
Dopustiva ruta	1-2-3-4-7-6-9-10-8-5-1	6-9-10-7-4-3-2-1-5-8-6	4-3-2-1-5-8-6-9-10-7-4
Minimalna ruta (m)	3450	3450	3450

Tabela 7: 5 i 10 stanica obilazi 1 autobus sa vraćanjem u polaznu stanicu



Slika 10: Optimalne rute jednog autobusa kroz zadatu gradsku mrežu G sa 5 i 10 stanica

U tabeli 8 uočavamo da se rezultati skoro i ne razlikuju, ista rešenja smo dobili za iste funkcije cilja, dok je razlika vidljiva jedino u vremenskom izvršavanju. Ovo je jedna od bitnijih stavki kada su u pitanju metode za rešavanje ovakvih problema.

Dobijeni rezultati pomoću potpunog pretraživanja, heuristike Greedy, metaheuristike GA i hibridizovanog GA+ algoritma za 5 i 10 stanica na populaciji od 100 jedinki sa početnom stanicom 1 predstavljeni su u tabeli 8.

Minimalne trase / ***instance***	*Vreme izvršavanja sec*	***Potpuno pretraživanje***	*Vreme izvršavanja sec*	***Heuristika( )***	*Vreme izvršavanja sec*	***GA(1, 100)***	*Vreme izvršavanja sec*	***GA+(1, 100)***
Matrica5	0.01939	**25**	0.00783	**25**	6.26595 4	**25**	3.08583 1	**25**
Matrica10	98.9860	**3450**	0.02298	**3450**	6.55697	**3450**	3.52265 2	**3450**
trase	**1 5 3 2 4 1**		**1 5 8 10 9 6 7 4 3 2 1**					

Tabela 8: Rezultati za matrice dimenzije 5 i 10 stanica

Možemo uočiti da vreme izvršavanja algoritma za veći broj stanica drastično raste što se tiče Potpunog pretraživanja, što smo u prethodnom poglavlju i objasnili. Metaheuristike problem rešavaju za drastično manje vremena i takođe pružaju optimalna rešenja za probleme manjih dimenzija. Sledi poređenje rezultata za probleme određivanja trase do dimenzije 13 stanica (Tabela 9).

instance	*Vreme izvršavanja sec*	***Potpuno pretraživanje***	*Vreme izvršavanja sec*	***Heuristika( )***	*Vreme izvršavanja sec*	***GA(1, 10)***	*Vreme izvršavanja sec*	***GA(1, 100)***	*Vreme izvršavanja sec*	***GA(1, 150)***
matrica3	0.002	**13**	0.001	**13**	0.939	**13**	5.900	**13**	8.915	**13**
matrica5	0.019	**25**	0.001	**25**	0.929	**25**	5.992	**25**	9.085	**25**
matrica7	0.150	**25**	0.001	**25**	0.469	**25**	6.143	**25**	9.365	**25**
matrica10	98.98	**3450**	0.002	**3450**	1.007	**3450**	6.473	**3450**	9.554	**3450**
matrica11	1089.37	**3060**	0.002	**3060**	0.979	**3060**	6.796	**3060**	9.618	**3060**
matrica12	13099.94	**2960**	0.002	3100	1.035	**2960**	6.582	**2960**	9.932	**2960**
matrica13	171634.79	**2850**	0.003	**2850**	1.0245	**2850**	6.569	**2850**	10.14	**2850**
matrica13		***GA+(1,pop) poboljšan sa Greedy***			0.520	**2850**	3.375	**2850**	4.367	**2850**

Tabela 9: Rezultati za matrice dimenzije 3,5,7,10,11,12 i 13 stanica

Kao što smo ranije i naveli, ovde potvrđujemo da je metoda potpunog pretraživanja neefikasna zbog svoje vremenske složenosti (osenčena polja crvenom nijansom u tabeli gde se ne preporučuje upotreba ovog algoritma), dakle u problemima većih dimenzija primena je bespredmetna. Upravo zbog vremenskog ograničenja i pribegavamo metaheuristikama koje su neuporedivo brže, što nam i rezultati pokazuju.

Objasnimo još i pojavu u slučaju matrice sa 12 stanica, gde prilikom korišćenja pohlepnog algoritma dolazi do preuranjene konvergencije, odnosno upadanja funkcije cilja u lokalni minimum, pa zato i dobijamo rešenje koje nije optimalno (3100). Možemo zaključiti da se i za matrice manjih dimenzija (do 13) preporučuje korišćenje GA i to hibridizovanog heuristikom pohlepnog algoritma, jer pruža tačna rešenja u najkraćem vremenskom roku (Tabela 10). Poslednji red u tabeli predstavlja rezultate za hibridizovani GA na instance dimenzije 13 na 10, 100 i 150 jedinki.

Broj testiranja ***Instance*** *dimenzije 12*	*Vreme izvršavanja sec*	***GA(1, 10) Min***	*Vreme izvršavanja sec*	***GA(1, 50) Min***	*Vreme izvršavanja sec*	***GA+(1,50) Poboljšan sa Greedy***	*Vreme izvršavanja sec*	***GA(1, 100) Min***	*Vreme izvršavanja sec*	***GA(1, 150) Min***	*Vreme izvršavanja sec*	***GA+(1,150) Poboljšan sa Greedy***
1	1.035	**2960**	3.548	3070	1.850	**2960**	6.582	**2960**	9.932	**2960**	5.118	**2960**
2	1.075	3080	3.595	**2960**	1.811	**2960**	6.578	**2960**	9.789	**2960**	5.040	**2960**
3	0.998	**2960**	3.552	**2960**	1.769	**2960**	6.536	**2960**	10.05	**2960**	4.954	**2960**
4	1.039	3070	3.614	**2960**	1.814	**2960**	6.715	**2960**	9.831	3070	5.075	**2960**
5	1.009	**2960**	3.539	**2960**	1.771	**2960**	6.633	**2960**	9.907	**2960**	5.051	**2960**
6	0.981	**2960**	3.577	**2960**	1.752	**2960**	6.613	**2960**	9.959	**2960**	5.008	**2960**
7	1.001	3080	3.528	3070	1.805	**2960**	6.556	**2960**	9.881	**2960**	5.235	**2960**
8	1.005	**2960**	3.542	**2960**	1.878	3070	6.702	**2960**	10.00	**2960**	5.008	**2960**
9	0.972	**2960**	3.503	**2960**	1.811	**2960**	6.595	**2960**	9.840	**2960**	5.034	**2960**
10	0.991	3080	3.549	**2960**	1.797	**2960**	6.852	**2960**	10.01	**2960**	4.923	**2960**

Tabela 10: Poređenje rezultata za 10 izvršavanja algoritama na instanci matrice dimenzije 12

instance	*Vreme izvršavanja sec*	***Heuristika( )***	*Vreme izvršavanja sec*	***GA(1, 10)***	*Vreme izvršavanja sec*	***GA+(1, 10)***	*Vreme izvršavanja sec*	***GA(1, 100)***	*Vreme izvršavanja sec*	***GA+(1, 100)***	*Vreme izvršavanja sec*	***GA(1, 150)***	*Vreme izvršavanja sec*	***GA+(1, 150)***
m20-1	0.01	**43**	1.032	**43**	0.56	**43**	7.16	**43**	3.62	**43**	10.6	**43**	5.61	**43**
m20-2	0.01	**193**	1.058	**193**	0.59	**193**	6.86	**193**	3.682	**193**	10.51	**193**	5.553	**193**
m30-1	0.02	223	1.125	221	0.60	221	7.67	**208**	3.950	**208**	11.38	**208**	5.869	**208**
m30-2	0.03	254	1.083	264	0.65	254	7.60	251	4.062	**246**	11.55	**246**	5.774	249
m40-1	0.034	237	1.188	248	0.70	231	8.15	226	4.365	**210**	12.47	220	6.745	**210**
m40-2	0.029	**244**	1.183	252	0.64	**244**	8.28	254	4.227	**244**	12.52	247	6.501	**244**
m50-1	0.052	288	1.242	330	0.75	288	8.63	304	4.439	**283**	13.31	298	6.846	284
m50-2	0.066	266	1.254	309	0.74	265	8.90	275	4.754	**252**	13.09	264	6.777	255

Tabela 11: 20, 30, 40 i 50 stanica obilazi 1 autobus sa vraćanjem u polaznu stanicu

U Tabeli 10 prikazani su rezultati za matricu dimenzije 12 stanica, zadate preko **.txt* dokumenta i testirane 10 puta GA i GA+ na populacijama od 10, 50, 100 i 150 jedinki. Na osnovu tabelarnih rezultata zaključujemo i da veličina populacije utiče na kvalitet rešenja, tako da što je populacija veća, to je veća i verovatnoća pronalaska boljeg rešenja, odnosno pronalaska bolje prilagođene jedinke. Upravo iz tog razloga u slučaju sa 10 i 50 jedinki uočljiva je veća disperzija u odnosu na optimalno rešenja.

U Tabeli 11 prikazani su rezultati za 20, 30, 40 i 50 (m20-1, m20-2, m30-1, m30-2, m40-1, m40-2, m50-1, m50-2) [instance1], [instance2], [instance3], [instance4] stanica, zadatih preko **.txt* dokumenta i testiranih heuristikom, GA i GA+ na populacijama od 10, 100 i 150 jedinki.

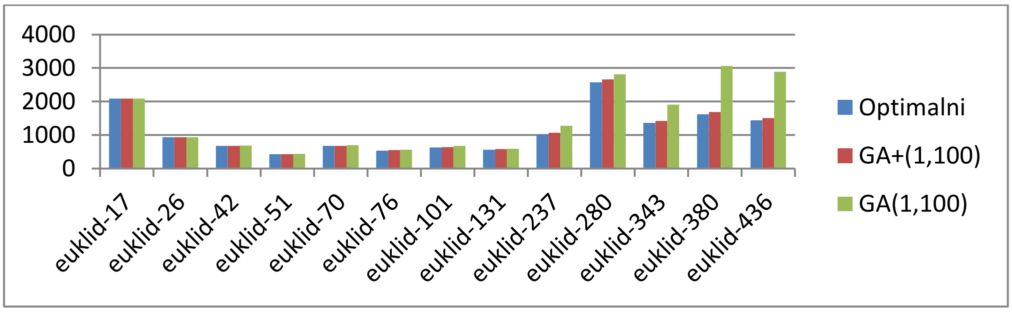
Povećanje veličine populacije pomaže nam u postizanju što boljeg rešenja, ali zbog rasta populacije, raste i vreme potrebno za implementaciju algoritma, što možemo zaključiti čitajući vrednosti za *Vreme izvršavanja* metoda u Tabeli 11. Uočimo još i to da smo za većinu testiranih primera bolja rešenja dobili upravo pomoću GA+, mada su ta rešenja vrlo bliska rešenjima heuristikom i običnim GA. Ovo nam govori još i to da moderna tehnologija hibridizovanog GA uz dobru procenu i podešavanje odgovarajućih početnih jedinki, može dati dovoljno dobro rešenje zadatog problema.

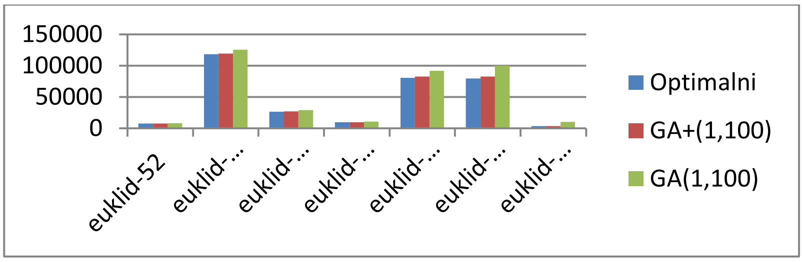
Na osnovu dobijenih rezultata možemo zaključiti da treba egzaktne metode izbegavati prilikom rešavanja problema velikih dimenzija, jer vreme izvršavanja algoritma zavisi od ukupnog broja osnovnih operacija, a zavisi i od veličine ulaznih podataka. Bez obzira što je njihova prednost dobijanje optimalnih rešenja, velika mana je vremenski problem. Tako da se u rešavanju ovakvih problema prednost daje heuristikama koje relativno jednostavno i računarski efikasno pronalaze rešenje, za koje možemo tvrditi da je "dovoljno dobro".

Ako pogledamo genetski algoritam uočavamo da za problem malih dimenzija nema smisla da se koriste, dok za problem mnogo većih dimenzija nego do sad prikazanih npr. više od 60 stanica, algoritam će u prihvatljivo kratkom vremenskom intervalu predložiti dopustiva rešenja. Ne možemo garantovati da je to baš optimalno rešenje kao u slučaju prve metode, ali postojanje tog rešenja je zagarantovano. Upravo slede eksperimentalni rezultati za probleme većih dimenzija i to zadatih preko Euklidskih koordinata. Za sledeća testiranja korišćene su javne instance sa [instance1], [instance2], [instance3], [instance4] i upoređeni su dobijeni rezultati pomoću metoda GA i GA+ sa rezultatima iz literature, pri čemu je i predstavljeno procentualno odstupanje od poznatih rešenja. U Tabeli 12 podebljane su vrednosti koje se poklapaju sa optimalnim.

instance	***Poznati optimalni rezultati***	*Vreme izvršavanja sec*	***GA(1,100) Min***	***Procenat odstupanja***	*Vreme izvršavanja sec*	***GA+(1,100) Poboljšan sa Greedy***	***Procenat odstupanja***
euklid-17	**2085**	6.923545	**2085**	0.00%	3.482964	**2085**	0.00%
euklid-26	**937**	7.485061	**937**	0.00%	3.910978	**937**	0.00%
euklid-42	**679**	8.617436	688.3	1.53%	4.488160	**679**	0.00%
euklid-51	**426**	8.92140	441.54	3.65%	4.678251	430.24	1.00%
euklid-52	**7542**	9.310959	7777.33	3.12%	4.691493	7544.3	0.03%
euklid-70	**675**	10.485481	691.7	2.47%	5.433650	682.1	1.05%
euklid-76	**538**	10.728151	564.62	4.95%	5.606298	552	2.60%
euklid-101	**629**	12.063686	673.5	7.07%	6.637476	642.7	2.18%
euklid-127	**118282**	13.950213	125374.3	6.00%	8.300201	119064.8	0.66%
euklid-130	**6110**	14.508344	6530.2	6.88%	8.737450	6250	2.29%
euklid-131	**564**	14.697346	591	4.79%	8.462868	582	3.19%
euklid-150	**26524**	15.730950	28842	8.74%	11.073891	27027	1.90%
euklid-194	**9352**	17.719414	10613	13.48%	18.108287	9624	2.91%
euklid-226	**80369**	20.887824	91415	13.74%	18.238060	82185	2.26%
euklid-237	**1019**	21.508053	1276	25.22%	26.409344	1065	4.51%
euklid-280	**2579**	24.231661	2817.7	9.26%	33.110068	2663.35	3.27%
euklid-343	**1368**	31.122896	1909	39.55%	87.041186	1425	4.17%
euklid-380	**1621**	34.029749	3060	88.77%	88.055618	1687	4.07%
euklid-436	**1443**	38.155713	2891	100.35%	188.281127	1507	4.44%
euklid-734	**79114**				2233.837469	82214.6	3.92%
euklid-813	**3119**				2363.174937	3363	7.82%

Tabela 12: Poređenje rezultata za euklidski zadate koordinate (javne instance) na populaciji od 100 jedinki

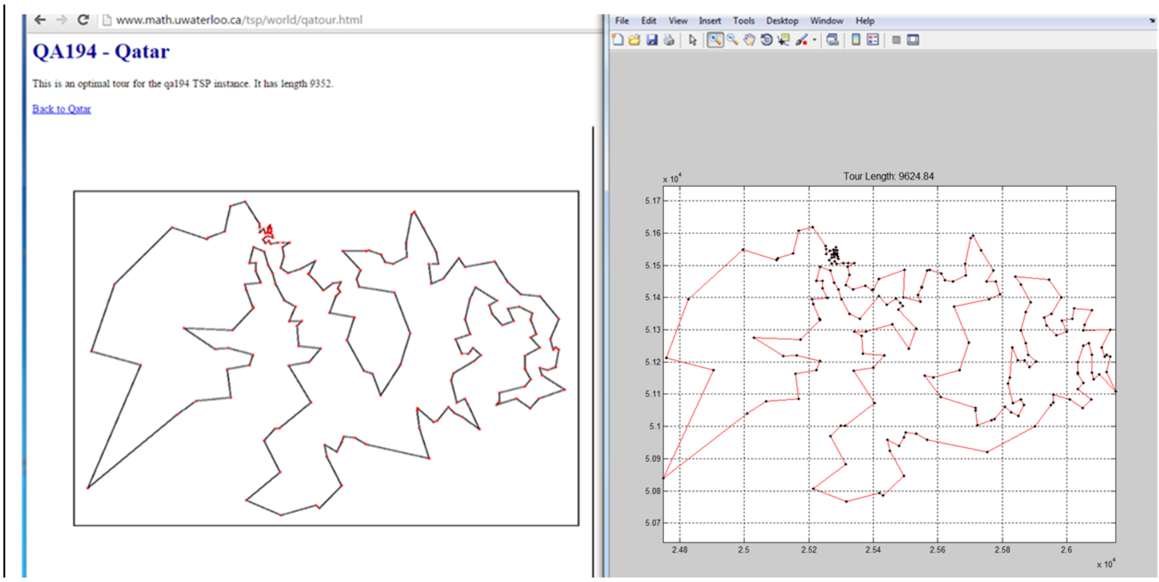




Slika 11: Grafički prikaz odstupanja rezultata GA i GA+ od javnih instanci

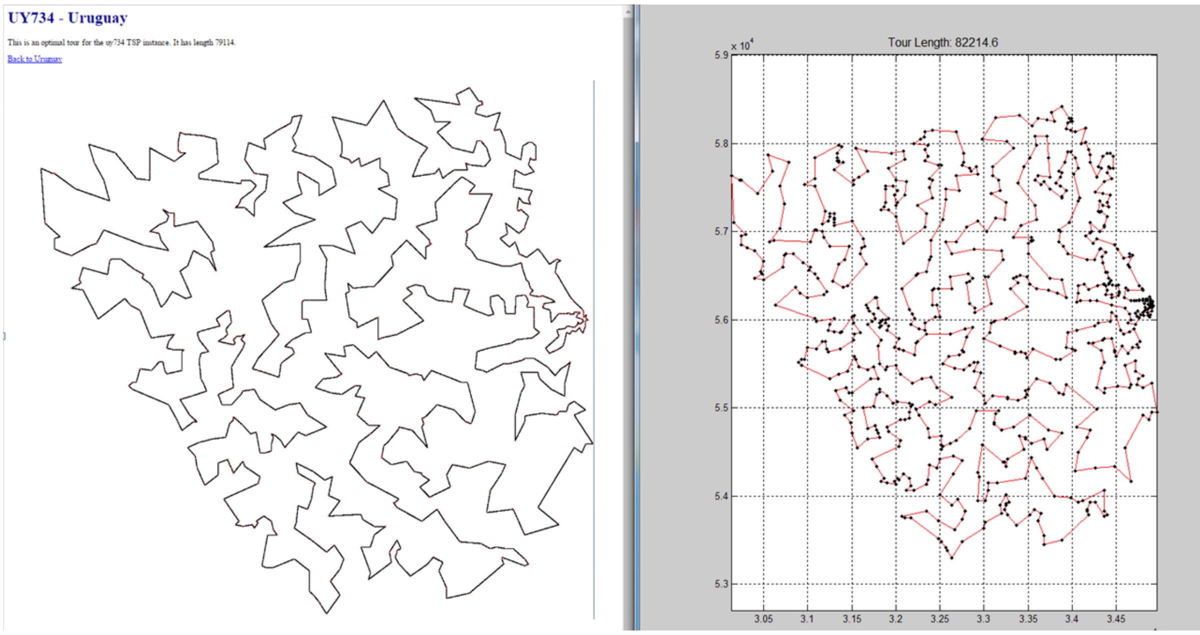
Primetimo (Slika 11) da GA+ pruža veoma zadovoljavajuća rešenja koja su bliska optimalnim za probleme većih dimenzija, dok prost GA ima veću fluktuaciju rezultata kako dimenzija problema raste. Slika 12 i 13 predstavljaju poređenje rezultata dobijenih predloženim algoritmima u programskom paketu Matlab 2010b sa instancama [instance2].

Instanca euklid-194 - Katar	Poznata vrednost	*GA+(1,100)* *Poboljšan sa Greedy*	*Procenat greške*
minimalna trasa je dužine	9352	9624	2.91%
Vreme izvršavanja u sec	Nije javno objavljeno	18.1	-



Slika 12: levo - Optimalna trasa kroz zadatu mrežu 194 stanice (država Katar) javno objavljeno rešenje, desno (Matlab GA+ dobijeno eksperimentalno rešenje)

Instanca euklid-734 - Urugvaj	Poznata vrednost	*GA+(1,100)* *Poboljšan sa Greedy*	*Procenat greške*
minimalna trasa je dužine	79114	82214.6	3.92%
Vreme izvršavanja u sec	Nije javno objavljeno	2233.83	-

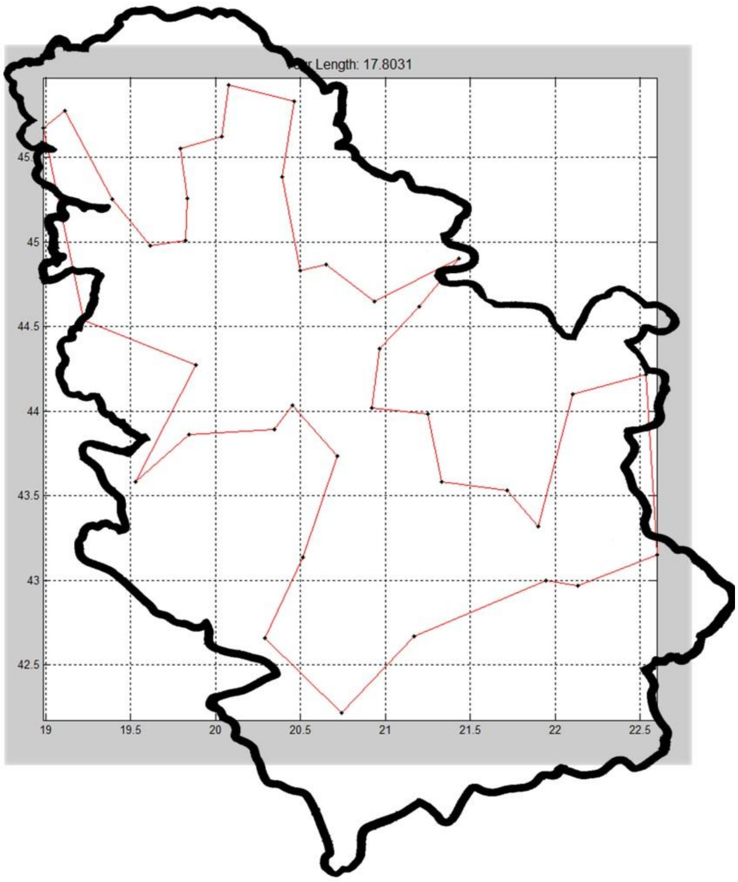


Slika 13: levo - Optimalna trasa kroz zadatu mrežu 734 stanice (država Urugvaj), javno objavljeno rešenje, desno - (Matlab GA+ dobijeno eksperimentalno rešenje)

2.4.2. *Dodatak - Turistička tura kroz gradove Srbije- helikopterom*

Pošto nisu bila dostupna međusobna drumska rastojanja između gradova, odlučila sam se za primenu euklidski zadate matrice[1] i pomoću algoritma GA+ odredila sam najkraću trasu vazdušnog saobraćaja, odnosno obilaska gradova Srbije helikopterom. Dakle, primena predložene metaheurističke metode poboljšanog Genetskog algoritma je moguća u mnogim problemima, što pruža raznovrsnu primenu u rešavanju NP-teških problema.

Redni broj	*grad*	*Latituda*	*Longituda*
1	**Aleksinac**	43.533	21.717
2	**Apatin**	45.671	18.985
3	**Bačka Palanka**	45.251	19.392
4	**Bečej**	45.623	20.036
5	**Bela Crkva**	44.900	21.433
6	**Beograd**	44.833	20.500
7	**Bor**	44.100	22.100
8	**Čačak**	43.892	20.348
9	**Gornji Milanovac**	44.033	20.450
10	**Jagodina**	43.983	21.250
11	**Kikinda**	45.830	20.462
12	**Kragujevac**	44.017	20.917
13	**Kraljevo**	43.733	20.717
14	**Kruševac**	43.583	21.333
15	**Leskovac**	42.998	21.946
16	**Loznica**	44.534	19.224
17	**Negotin**	55.217	22.533
18	**Novi Sad**	45.259	19.833
19	**Niš**	43.317	21.900
20	**Novi Pazar**	43.137	20.512
21	**Pančevo**	44.867	20.650
22	**Peć**	42.660	20.292
23	**Pirot**	43.150	22.600
24	**Požarevac**	44.617	21.200
25	**Priboj**	43.584	19.526
26	**Priština**	42.667	21.167
27	**Prizren**	42.215	20.740
28	**Ruma**	45.008	19.821
29	**Senta**	45.928	20.078
30	**Smederevo**	44.650	20.933
31	**Smederevska Palanka**	44.367	20.967
32	**Sombor**	45.774	19.112
33	**Srbobran**	45.550	19.793
34	**Sremska Mitrovica**	44.977	19.615
35	**Užice**	43.861	19.843
36	**Valjevo**	44.272	19.885
37	**Vlasotince**	42.967	22.133
38	**Zrenjanin**	45.383	20.391



Slika 14: Optimalna trasa turističke ture kroz gradove Srbije

[1] *http://static.astronomija.org.rs/razno/koordinate/koordinate.htm*

2.4.3. Rezultati podmodela b

Postojeći algoritmi su modifikovani i prilagođeni konstruisanom podmodelu b, pri čemu imamo četiri tipa algoritama čije rezultate upoređujemo, a to su algoritmi slični onima u podmodelu a.

U Tabelama 13 i 14 uporedili smo dobijene rezultate za unos 5 i 10 stanica korišćenjem egzaktnog algoritma Potpuno pretraživanje, heuristike koja se sastoji od Pohlepnog i 2-opt algoritma za poboljšanje rešenja gde je unapred fiksirana prva stanica, metaheuristika Genetski algoritam i GA+.

n=5 stanica, k=1 autobus			
	Potpuno pretraživanje	*Pohlepni algoritam*	*Genetski Algoritam*
Vreme izvršavanja	0.01 min	0.001 min	0.01 min
Dopustiva ruta	2-3-4-1-5	1-4-2-3-5	2-3-4-1-5
Minimalna ruta (min)	15	19	15

Tabela 13: 5 stanica obilazi 1 autobus bez vraćanja u polaznu stanicu

n=10 stanica, k=1 autobus			
	Potpuno pretraživanje	*Pohlepni algoritam*	*Genetski Algoritam*
Vreme izvršavanja	1 min	0.005 min	0.02 min
Dopustiva ruta	6-7-10-9-8-5-1-2-3-4	1-2-3-4-7-6-9-10-8-5	6-7-10-9-8-5-1-2-3-4
Minimalna ruta (m)	2900	3070	2900

Tabela 14: 10 stanica obilazi 1 autobus bez vraćanja u polaznu stanicu

Zaključak koji možemo izvesti iz dosadašnjih eksperimentalnih rezultata je isti kao i u prethodnom modelu. Prednost dajemo heurističkim metodama koje uvek daju “dovoljno dobro” rešenje za relativno kratak vremenski period, što znači rešenje će sigurno biti dopustivo, a možda baš upravo i optimalno.

Uporedimo u tabeli 14 dobijene vrednosti funkcije cilja. Primetimo da je vrednost 2900 manja od 3070, ali da je vreme izvršavanja ove dve heuristike suprotno srazmerno odnosno redom 0.02 minuta i 0.005 minuta. U tabelama 15 i 16 prikazani su rezultati na istim instancama kao u prethodnom podmodelu a, što je prikazano u tabelama 9 i 11, samo što su ovde trase linijske, bez povratka u polaznu stanicu, pa su samim tim i funkcije cilja manje.

instance	Vreme izvršavanja sec	Potpuno pretraživanje	Vreme izvršavanja sec	Heuristika( )	Vreme izvršavanja sec	GA+(1, 10)	Vreme izvršavanja sec	GA+(1, 100)	Vreme izvršavanja sec	GA+(1, 150)
matrica3	0.002	**7**	0.001	**7**	1.008	**7**	6.58	**7**	9.23	**7**
matrica5	0.019	**15**	0.002	**19**	1.01	**15**	6.6	**15**	9.95	**15**
matrica7	0.150	**19**	0.009	**19**	1.04	**19**	6.74	**19**	10.33	**19**
matrica10	97.62	**2900**	0.011	3070	1.03	**2900**	6.71	**2900**	10.42	**2900**
matrica11	1074.32	**2660**	0.013	2680	1.05	**2660**	7.11	**2660**	10.55	**2660**
matrica12	13062.24	**2560**	0.011	2720	1.07	**2560**	6.97	**2560**	10.56	**2560**
matrica13	171513.52	**2450**	0.012	2470	1.16	**2450**	7.03	**2450**	10.42	**2450**

Tabela 15: Rezultati za matrice dimenzije 3,5,7,10,11,12 i 13 stanica

instance	Vreme izvršavanja sec	Heuristika( )	Vreme izvršavanja sec	GA(1, 10)	Vreme izvršavanja sec	GA+(1, 10)	Vreme izvršavanja sec	GA(1, 100)	Vreme izvršavanja sec	GA+(1, 100)	Vreme izvršavanja sec	GA(1, 150)	Vreme izvršavanja sec	GA+(1, 150)
m20-1	0.008	38	1.09	36	1.15	**35**	7.42	**35**	7.56	36	11.5	**35**	11.3	**35**
m20-2	0.009	186	1.08	161	1.2	**156**	7.15	**156**	7.47	**156**	11.6	166	11.89	**156**
m30-1	0.01	219	1.16	198	1.24	208	8.1	**188**	8.39	179	12.34	**175**	12.67	**175**
m30-2	0.01	251	1.17	220	1.23	223	8.1	**210**	8.42	**209**	12.28	215	12.67	**209**
m40-1	0.02	236	1.34	244	1.32	203	8.61	197	9.12	**191**	13.51	203	13.12	**191**
m40-2	0.02	239	1.39	263	1.33	235	8.92	242	9.17	**210**	13.05	227	13.61	**210**
m50-1	0.05	281	1.34	287	1.45	263	9.22	285	9.67	**258**	14.46	**258**	14.54	**258**
m50-2	0.05	260	1.36	256	1.51	242	9.67	252	9.95	240	14.15	248	14.33	**238**

Tabela 16: 20, 30, 40 i 50 stanica obilazi 1 autobus bez vraćanja u polaznu stanicu

3. MATEMATIČKI MODEL II

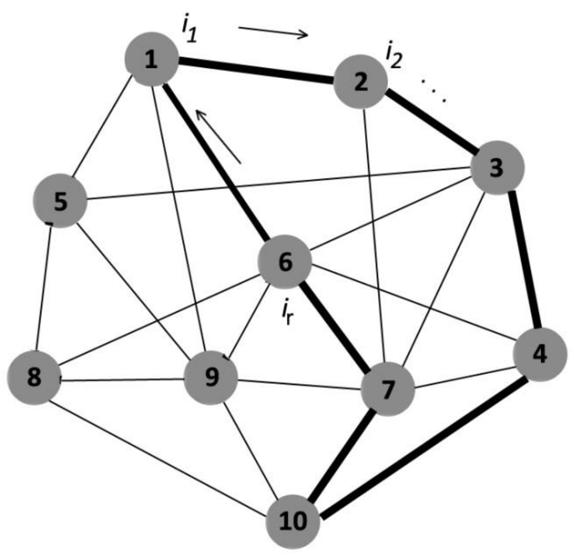
Proširimo i prilagodimo prethodni model sa jednom linijom na novi model sa više od jedne linije (autobusa). Funkciju cilja malo ćemo modifikovati zbog pojave više linija i samim tim težiti što boljoj optimizaciji mreže po svakoj pojedinačnoj postojećoj liniji. Međutim i ovaj model možemo takođe podeliti na dva podmodela i to na:

a) autobusi se vraćaju u polaznu stanicu posle posete svih svojih stanica

b) autobusi se ne vraćaju u polaznu stanicu posle obilaska svih stanica

Ali u ovom radu posvetićemo se samo podmodelu kada se autobusi vraćaju u polaznu stanicu posle posete svih svojih stanica, odnosno kružnih linija. Hamiltonova kontura, pomenuta u prvom matematičkom modelu (Slika 15) i u ovom modelu biće jedno od ograničenja kako bi obezbedili povratak autobusa u polaznu stanicu.

$$H_l = (i_1, i_2, \dots, i_{r-1}, i_r, i_1), \qquad i_p \neq i_l \ \forall p, l \in \mathcal{R}_r, \qquad p \neq l.$$



Slika 15: Skica Hamiltonove konture

Smatraćemo da se za polaznu stanicu svih autobusa uzima jedna stanica iz skupa

$$\mathcal{S} = \{1,2,3 \dots n\}.$$

Ostale promenljive i parametri u ovom modelu biće:

$r = \{1,2, \dots k\}$ – skup svih autobusa (linija), $k > 1$

$n > 3$ – postojanje više od 3 stanice obezbeđuje uvođenje bar dve linije

$\mathcal{R}_r$ - skup stanica koje posećuje r-ti autobus, r ∈ {1,2, … k}

$$x_{ij} = \begin{cases} 1, & \text{ako autobus iz stanice i posećuje stanicu j} \\ 0, & \text{inače} \end{cases}, \qquad i, j \in \mathcal{R}_r$$

(5) $$\min_{\sum_{r=1}^{k} |\mathcal{R}_r| = 3k, \ldots, k(n-1)} \min \sum_{r=1}^{k} \sum_{i \in \mathcal{R}_r} \sum_{j \in \mathcal{R}_r} c_{ij} x_{ij}$$

(6) $$\sum_{j \in \mathcal{R}_r} x_{ij} = 1, \qquad i \in \mathcal{R}_r, \qquad r = 1, \ldots k$$

(7) $$\sum_{i \in \mathcal{R}_r} x_{ij} = 1, \qquad j \in \mathcal{R}_r, \quad r = 1, \ldots k$$

(8) $$\sum_{i \in \mathcal{S}'} \sum_{j \in \mathcal{S}'} x_{ij} \le |\mathcal{S}'| - 1, \qquad \mathcal{S}' \subset \mathcal{R}_r, \ \mathcal{S}' \neq \emptyset, \ \ \mathcal{S}' \neq \mathcal{R}_r, r = 1, \ldots, k$$

(9) $$\mathcal{R}_r \subset \mathcal{S}, \qquad \bigcup_{r=1}^{k} \mathcal{R}_r = \mathcal{S}, \quad r = 1, \ldots k$$

(10) $$2 < |\mathcal{R}_r| < n, \qquad r = 1, \ldots k$$

(11) $$\forall \mathcal{R}_r \ \exists \mathcal{R}_p \Rightarrow \ \mathcal{R}_r \cap \mathcal{R}_p \neq \emptyset, \qquad r \neq p, \qquad r, p = 1, \ldots k$$

(12) $$\mathcal{R}_r \neq \mathcal{R}_p, \qquad r \neq p, \qquad r, p = 1, \ldots k$$

Funkcija cilja (5) minimizira ukupan pređeni put svih autobusa. Međutim, uveli smo ujedno i minimizaciju sume kardinalnosti svih skupova stanica, tj. zbira broja stanica svake linije pojedinačno, čime se obezbeđuje minimalan broj zajedničkih stanica za različite autobuse. Ograničenja (6) - (8) analogna su prethodnom modelu samo što se posmatraju na skupu $\mathcal{R}_r$ pojedinačno za svaku liniju (autobus) i obezbeđuju autobusu tačno jedan prolazak kroz svaku stanicu odgovarajuće linije. Ograničenje (9) obezbeđuje postojanje minimum dva autobusa i pokrivenost svih zadatih stanica mreže sa dva ili više autobusa, pri čemu (10) kaže da svaki od autobusa mora u svojoj liniji sadržati najmanje 3 stanice, a strogo manje od n stanica, kako bi se obezbedilo uvođenje i drugog autobusa što je zadovoljeno i na osnovu (9). Ograničenje (11) obezbeđuje presedanja i stizanje iz svake stanice u svaku, tj. pruža jednu zajedničku stanicu sa jednim od preostalih autobusa (linija). Na kraju, ograničenje (12) sprečava pojavu više identičnih linija, odnosno sprečava postojanje više od jednog autobusa koji bi vozili istom rutom, čime obezbeđujemo postojanje različitih ruta.

Kao što smo i ranije naveli, ovo je težak problem kombinatorne optimizacije, posebno za problem većih dimenzija i rešava se najefikasnije pomoću metaheuristika, koje predstavljaju ozbiljan aparat za dalji razvoj optimizacije u budućnosti. Ilustrujmo primer za formirani matematički model II pri čemu ćemo kao rešenje problema, dobiti upravo skupove stanica $\mathcal{R}_r, r = 1, \dots k$ koje svaki od autobusa treba posetiti, zajedno sa odgovarajućom minimalnom vrednošću zadate funkcije cilja.

Iz postavke problema u ovom modelu može se videti da se zapravo radi o višestrukom problemu putujućeg trgovca (Travelling Salesman Problem – TSP). Transportni problem je jedan od problema celobrojnog linearnog programiranja među koje spada i određivanje najpovoljnijeg puta odnosno problem rutiranja vozila - *VRP* (eng. *Vehicle Routing Problem*). Međutim, cilj TSP-a je minimizirati pređeno rastojanje tako da trgovac poseti sve lokacije tačno jednom i da se vrati u polaznu tačku, dok funkcija cilja VRP-a može biti i drugačija [Tot01]. Rešenje VRP-a predstavlja skup ruta (puteva) pri čemu moraju sva unapred definisana ograničenja biti zadovoljena. U slučaju da je na raspolaganju samo jedno vozilo i ako ne postoje dodatna ograničenja tada VRP prelazi u dobro poznati problem trgovačkog putnika, gde je potrebno jednim vozilom obići sve tačke grafa uz minimalni pređeni put (ili minimalnu cenu, vreme, kapacitet…).

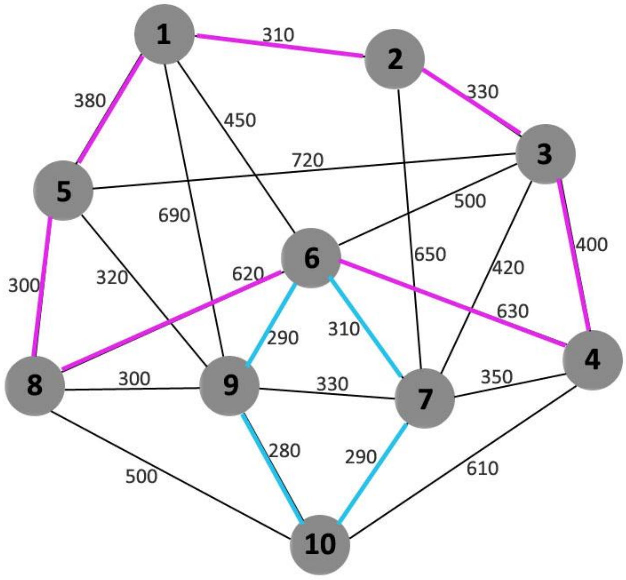
3.1. Eksperimentalni rezultati

Kao u prethodnom primeru koristićemo iste instance za 10 stanica, ali u ovom slučaju ćemo unapred fiksirati broj autobusa tako da ih bude dva i svi će kretati sa iste polazne stanice.

Takođe, vidi se da su genetski algoritmi korisni za one klase problema koje se ne mogu rešiti na klasične načine. Iako po brzini nisu u vrhu, po veličini područja koje pretražuju su verovatno daleko bolji od svih ostalih metoda. To se jako dobro vidi na primeru problema trgovačkog putnika, čiji je prostor rešenja ogroman već i za stotinjak gradova.

a) dva autobusa (dve trase)
b) tri autobusa (tri trase).

a)



Slika 16: Gradska mreža G, prikaz optimalnog rutiranja dva autobusa

Na Slici 15, vidimo optimalno rešenje za rutiranje dva autobusa, tako da imaju jednu zajedničku stanicu 6 i ispunjavaju sva zadata ograničenja za pokrivenost cele mreže.

Trasa autobusa 1 je:

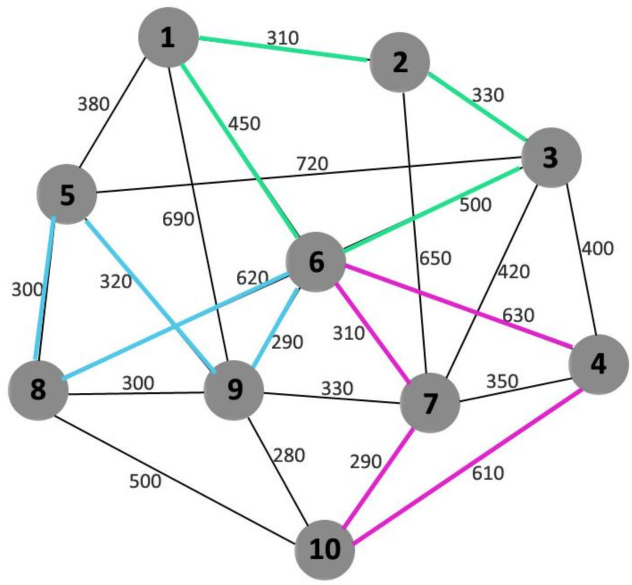
6 – 8 – 5 – 1 – 2 – 3 – 4 – 6

Trasa autobusa 2 je:

6 – 9 – 10 – 7 – 6

Pri tome je minimalna vrednost naše funkcije cilja (5), tj. minimalni ukupan pređeni put **4040** kada nam je zajednička stanica 6, a do datog rešenja se pomoću GA dolazi za 3.74 sec.

b)



Slika 17: Gradska mreža G, prikaz optimalnog rutiranja tri autobusa

Trasa autobusa 1 je:

6 – 7 – 10 – 4 – 6

Trasa autobusa 2 je:

6 – 8 – 5 – 9 – 6

Trasa autobusa 3 je:

6 – 1 – 2 – 3 – 6

pri čemu je minimalna vrednost naše funkcije cilja (5), tj. minimalni pređeni put **4860** kada nam je zajednička stanica 6, a do datog rešenja se pomoću GA dolazi za 3.917sec.

U narednoj tabeli 17 prikazani su rezultati matematičkog modela II za obilazak date mreže sa 3 ili više autobusa što je predstavljeno sa GA (start,broj_autobusa). GA je testiran na populacijama od 100 jedinki za instance matrica dimenzije 30, 40 i 50. Uporedili smo rezultate i za tri različita slučajna izbora zajedničkih stanica 1, 5 i 10 i izvršili po tri testiranja za svaku instancu. Dobijena su dopustiva rešenja koja možemo uočiti i da su vrlo bliska u svakom od tri merenja. Možemo uočiti i da se rešenja za probleme manjih dimenzija brže dobijaju.

instance	*Zajednička stanica*	*Redni broj testiranja*	*Vreme sec*	*GA(start,3)*	*Vreme sec*	*GA(start,4)*	*Vreme sec*	*GA(start,5)*	*Vreme sec*	*GA(start,6)*	*Vreme sec*	*GA(start,7)*	*Vreme sec*	*GA(start,8)*
m30-1	**1**	***1.***	22.08	286	21.53	372	23.87	422	26.45	512	22.69	639	25.82	675
		2.	22.08	321	21.44	368	23.86	427	25.49	506	22.56	618	25.72	671
		3.	22.10	325	21.57	350	23.87	452	24.77	552	22.52	641	25.76	734
	5	***1.***	22.24	333	21.29	450	24.07	510	24.38	583	22.05	705	25.68	780
		2.	22.11	402	21.42	414	24.07	474	24.45	590	22.48	688	25.67	770
		3.	22.08	334	21.32	390	23.90	504	24.50	567	22.43	673	25.62	759
	10	***1.***	22.05	310	21.31	323	23.89	359	24.79	441	22.59	506	25.69	598
		2.	22.13	321	21.44	364	23.94	361	24.62	463	22.49	503	25.65	598
		3.	22.09	287	21.41	331	23.87	393	24.61	452	22.45	522	25.78	579
m40-1	**1**	***1.***	23.19	380	23.71	366	24.26	353	24.69	424	25.39	503	25.99	550
		2.	22.91	337	23.44	352	24.14	404	24.59	411	25.39	496	26.08	543
		3.	22.94	366	23.41	328	24.07	456	24.64	445	25.26	492	26.03	601
	5	***1.***	22.91	357	23.61	353	24.16	406	24.52	473	25.26	520	26.02	614
		2.	23.14	461	23.40	332	24.09	434	24.46	477	25.13	570	25.86	573
		3.	23.11	379	23.27	336	24.01	431	24.36	501	25.09	492	25.96	576
	10	***1.***	23.07	424	23.60	409	24.20	467	24.56	493	25.14	599	25.95	646
		2.	22.91	399	23.25	442	24.35	448	24.42	466	25.44	579	26.08	618
		3.	22.86	364	23.48	432	24.23	443	24.76	523	25.36	567	26.10	677
m50-2	**1**	***1.***	22.15	364	22.13	498	24.91	496	22.63	547	24.99	589	23.67	656
		2.	21.85	408	22.17	463	24.63	542	22.61	568	25.01	650	23.53	715
		3.	22.04	386	22.22	472	24.54	468	22.68	544	25.06	590	23.51	643
	5	***1.***	22.03	484	22.21	509	24.82	473	22.70	574	24.93	696	23.72	754
		2.	21.96	422	22.14	463	25.58	546	22.93	641	24.99	633	23.48	758
		3.	22.03	403	22.25	422	25.25	479	22.71	582	24.96	715	23.54	743
	10	***1.***	21.94	409	22.44	474	24.63	485	22.72	537	25.07	548	23.73	658
		2.	21.94	420	22.26	402	24.67	470	22.87	559	24.96	656	23.55	637
		3.	21.95	374	22.28	448	24.69	496	22.65	491	25.00	592	23.50	687

Tabela 17: 20, 30, 40 i 50 stanica obilaze 3-8 autobusa sa vraćanjem u polaznu stanicu

Na osnovu dosadašnjih rezultata zaključujemo da se povećanjem broja vozila i funkcija cilja povećava, a samim tim i troškovi transporta. Prema tome, cilj optimizacije jeste smanjenje troškova, odnosno maksimalna ušteda u pogledu vozila, ali sa druge strane, nama je cilj i pokriti što je više moguće datu mrežu puteva. Upravo zbog mnogih ograničenja i kompleksnosti problema ovo i spada u teške probleme kombinatorne optimizacije koji se rešavaju raznim heurističkim metodama, odnosno intuitivnom kombinacijom poznatih algoritama.

Navedimo još jedan rezultat, a to je upravo pokrivenost mreže od 50 čvorova (stanica) sa 8 autobusa. Za instancu m50_2 u slučaju ako uzmemo stanicu broj 5 za polaznu, dobijamo sledeće trase autobusa:

Trasa 1

5 7 46 41 10 26 23 5

Trasa 2

5 1 16 22 35 29 40 5

Trasa 3

5 42 4 14 49 44 28 5

Trasa 4

5 32 43 20 13 34 48 5

Trasa 5

5 17 50 3 27 24 25 5

Trasa 6

5 38 31 11 39 37 36 5

Trasa 7

5 2 30 8 19 47 18 21 5

Trasa 8

5 45 12 33 15 9 6 5

Pri tome je minimalna vrednost naše funkcije cilja (5), tj.minimalni ukupan pređeni put **743** kada nam je zajednička stanica 5, a do datog rešenja se pomoću GA dolazi za 23.54 sekunde.

4. ZAKLJUČAK

Nakon dugog i ozbiljnog proučavanja raznih metoda modeliranja i optimizacije u oblastima rutiranja i transporta, dolazimo do zaključka da je pomenuti problem određivanja trasa linija gradskog saobraćaja mnogo kompleksniji nego što izgleda na prvi pogled.

U ovom radu opisana su dva modela i implementacije algoritama koji su namenski konstruisani za rešavanje razmatranog problema optimizacije. Predloženi matematički model je model TSP-a. Traženje Hamiltonovog ciklusa najmanje težine u težinskom grafu, kao i utvrđivanje da li je graf Hamiltonov, spadaju u kategoriju najtežih algoritamskih zadataka. Zbog svoje složenosti, problem zahteva rešavanje tehnikama koje su u mogućnosti da proizvedu kvalitetna rešenja u ograničenom vremenskom roku, a takve tehnike su upravo heurističke. Konstruisani Genetski algoritam i Heuristika pohlepnog algoritma, kao i njihova hibridizacija pružili su kvalitetne rezultate kako u pogledu postizanja optimalnog rešenja tako i u pogledu vremenskog izvršavanja. Primena ovih specijalnih metoda modifikovanog Genetskog algoritma poboljšanog pohlepnim algoritmom efikasno dovodi do rešenja na testiranim instancama.

Značaj postignutih rezultata su:

- uvođenje hibridizacije Genetskog i pohlepnog algoritma u cilju bržeg postizanja kvalitetnijeg rešenja,
- prilagođavanje genetskih operatora datom problemu i načinu kodiranja,
- dobijanje rešenja za problem organizovanja turističkog obilaska gradova Srbije koji do sada nije rešavan.

Doprinosi budućem razvoju su:

- rešavanje mnogih problema iz realnog života koji se mogu predstaviti kao problem rutiranja,
- optimizacija gradskog i vazdušnog saobraćaja, turističkih tura i mnogih drugih problema raspoređivanja i pokrivenosti,

Eksperimentalni rezultati dobijeni u ovom radu podržavaju korišćenje metaheurističkih metoda u rešavanju NP-teških problema većih dimenzija, za razliku od egzaktnih metoda koje su neupotrebljive u tom slučaju.

Zbog svega navedenog, a što se može videti iz priloženih naučnih dostignuća i eksperimentalnih rezultata, jeste da metaheurističke metode predložene u ovom radu rešavaju veoma uspešno probleme velikih dimenzija. Zahvaljujući tome, ostaje prostora za nastavak istraživanja u oblasti rutiranja, gde je potencijalna ideja rešiti varijantu rutiranja vozila sa vremenskim ograničenjem (eng. Vehicle Routing Problem with Time Windows - VRPTW). Da bi predloženi matematički model u ovom radu bio još funkcionalniji, potrebno je uvesti i vremenske promenljive od kojih bi zavisila i frekvencija putnika na stanicama, a samim tim i broj autobusa, odnosno frekvencija prolaska autobusa kroz stanice. Za sada ovaj problem iz realnog života ostavljamo za neko naredno istraživanje.

LITERATURA

[Ant89] **Antonisse J.**, A New Interpretation of Schema That Overturns the Binary Encoding Constraint, in: Proceedings of the Third International Conference on Genetic Algorithms, Morgan Kaufmann, San Mateo, California, pp. 86-91, (1989)

[Bea93] **Beasley D., Bull D.R., Martin, R.R.**, An Overview of Genetic Algorithms, Part 2, Research Topics, University Computing, Vol. 15, No. 4, pp. 170-181, (1993)

[BeD93a] **Beasley D., Bull D., Martin R.**, „An overview of genetic algorithms, part 1, fundamentals“, University Computing, Vol. 15, No. 2, pp. 58-69, 1993. ftp://ralph.cs.cf.ac.uk/pub/papers/GAs/ga_overview1.ps

[BeD93b] **Beasley D., Bull D., Martin R.**, „An overview of genetic algorithms, part 2, research topics“, University Computing, Vol. 15, No. 4, pp. 170-181, 1993. ftp://ralph.cs.cf.ac.uk/pub/papers/GAs/ga_overview2.ps

[Bok87] **Booker L.,** „Improving search in genetic algorithms“, Genetic Algorithms And Simulated Annealing, Pitman Publishing, London, pp. 61-73, 1987

[Brs88] **Brassard G., Bratley P.**, "Algorithms: Theory and Practice", Prentice-Hall Int., Englewoods Cliffs NJ (1988).

[Cve96] **Cvetković D., Čangalović M., Dugošija Đ, Kovačević-Vujčič V., Simić S., Vuleta J.,** Kombinatorna Optimizacija – Matematička teorija i algoritmi, Društvo operacionih istraživača Jugoslavije- DOPIS, Beograd 1996,

[Coo71] **Cook S.A,** "The Complexity of theorem proving procedures", Proceedings of the Third Annual ACM, Symposium on Theory of Computing, pp.151-158 ACM Press, New York, NY USA, (1971)

[Dav91] **Davis L.,** „Handbook of genetic algorithms“, Van Nostrand Reinhold, New York, 1991.

[Dar59] **Darwin C.,** The origin of species, London, 1859

[Del07] **Delibašić M.,** Analiza i implementacija grafovskih algoritama, Diplomski rad, Elektrotehnički fakultet, Beograd, 2007,

[Dow96] **Dowsland K. A.**, "Genetic algorithms a tool for OR?", Journal of Operational Research Society, Vol. 47, pp.550-561 (1996).

[Fil06] **Filipović V.**, „Operatori selekcije i migracije i WEB servisi kod paralelnih evolutivnih algoritama“, Doktorska disertacija, Matematički fakultet, Beograd, 2006.

[Fil98] **Filipović V.**, "Predlog poboljšanja operatora turnirske selekcije kod genetskih algoritama", Magistarski rad, Univerzitet u Beogradu, Matematički fakultet, (1998)

[Gol10] **Golub M.**, "Genetski algoritmi", FER, Zagreb ,(2010).

[Gol89] **Goldberg D.E.,** Genetic algorithms in search, optimization and machine learning. Reading., MA: Addison-Wesley, (1989)

[Jak96] **Jakobović D**, Genetski algoritam, Diplomski rad, FER, Zagreb

[Koz92] **Koza J.R**. "Genetic Programming: On the Programming of Computers by Means of Natural Selection", MIT Press, Cambridge, MA, (1992).

[Kra00] **Kratica J. J.**, Paralelizacija genetskih algoritama za rešavanje nekih NP-kompletnih problema, Doktorska disertacija, Univerzitet u Beogradu, Matematički fakultet, Beograd, 2000.

[Mar08] **Marić M.**, Rešavanje nekih NP-teških hijerarhijsko-lokacijskih problema primenom genetskih algoritama, Doktorska disertacija, Univerzitet u Beogradu, Matematički fakultet, Beograd, 2008.

[Ognj04] **Ognjanović Z., Krdžavac N.,** „Uvod u teorijsko računarstvo“, Fakultet organizacionih nauka, Beograd, 2004.

[Pau97] **Paunić Đ.**, "Strukture podataka i algoritmi", Univerzitet u Novom Sadu, Prirodno-Matematički fakultet, Novi Sad (1997).

[Rad11] **Radojičić N.**, Rešavanje nekih NP-teških problema diskretne optimizacije, Master rad, Matematički fakultet, Beograd, 2011

[Ree93] **Reeves, C. R.**, Modern heuristic techniques for combinatorial problems, John Wiley & Sons, Inc., New York, 1993

[Sch00] **Schabel A.**, Optimization in Public Transportation, Stop Location, Delay Management, Springer 2000,

[Sch03] **Schrijver A.**, Combionatorial Optimization Polyhedra and Efficiency, Springer 2003,

[Sim12] **Simićević A.**, Lokacijski problem na mrežama, diplomski rad, Matematički fakultet, Beograd, 2012,

[Sta13] **Stanković M.**, Rešavanje nekih problema kombinatorne optimizacije algoritmom tabu pretraživanja, Master rad, Matematički fakultet, Beograd, 2013,

[Tot01] **Toth P., Vigo D.**, Monographs on Discrete Mathematics and Applications – The Vehicle Routing Problem, SIAM Philladelphia 2001,

[Vug96] **Vugdelija M., Kratica J., Filipović V., Radojević S.**, "Mogućnosti genetskih algoritama u mašinskom učenju", XXII Jupiter konferencija, Zbornik radova, Mašinski fakultet, Beograd, str.4.55-4.59, (1996).

[instance1] https://people.sc.fsu.edu/~jburkardt/datasets/tsp/tsp.html

[instance2] http://www.math.uwaterloo.ca/tsp/vlsi/index.html

[instance3] http://www.math.uwaterloo.ca/tsp/world/countries.html

[instance4] http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/

KOMPJUTERSKA PODRŠKA

Matlab 2010b - korišćenje ugrađenih funkcija za konstrukciju pogodnog algoritma koji će rešiti naše modele sa datim ograničenjima

PRILOG

1* Algoritam *heuristic_circle(start)*

```
function [] = heuristic_circle(start)
% Ucitavanje vrednosti iz datoteke
fid = fopen('matrica10.txt', 'r');
l = fscanf(fid, '%g %g', [1 1]);
h = fscanf(fid, '%g %g', [l l]);
X=h';
for i=1:l
    for j=1:l
         if X(i,j)==0
               X(i,j)=inf;
           end
    end
end
X
fclose(fid);

m = size(X,1);
[n,dim] = size(X);
s = randperm(n);

Lmin = inf;
for k = 1:m
    p = greedy(s(k),D);
    [p,L] = exchange2(p,D);
        if L < Lmin
        Lmin = L;
        pmin = p;
    end
end

p=pmin;
L = Lmin;
index = find(p == start);
finalRoute=p([index:end 1:index-1])
sprintf('Minimalni put je: %d', L)
%Funkcija pohlepni algoritam
function p = greedy(s,D)
n = size(D,1);
p = zeros(1,n,'uint16');
p(1) = s;
for k = 2:n
    D(s,:) = inf;
    [junk,s] = min(D(:,s));
    p(k) = s;
end
%Funkcija 2*opt
function [p,L] = exchange2(p,D)
n = numel(p);
zmin = -1;
 while zmin < 0
    zmin = 0;
    i = 0;
    b = p(n);
    while i < n-2
        a = b;
        i = i+1;
        b = p(i);
        Dab = D(a,b);
        j = i+1;
        d = p(j);
        while j < n
            c = d;
            j = j+1;
            d = p(j);
            z = (D(a,c) - D(c,d)) + D(b,d) - Dab;
            if z < zmin
                zmin = z;
                imin = i;
                jmin = j;
            end         end     end
```

```
    if zmin < 0
        p(imin:jmin-1) = p(jmin-1:-1:imin);
    end
end
q = double(p);
ind = sub2ind([n,n],q,[q(2:n),q(1)]);
L = sum(D(ind));
```

2* Algoritam *GA_circle(start,popSize)*

```
function [] = GA_circle(start,popSize)

fid = fopen('matrica10.txt', 'r');
l = fscanf(fid, '%g %g', [1 1]);
h = fscanf(fid, '%g %g', [l 1]);
matrica=h';
for i=1:l
    for j=1:l
         if matrica(i,j)==0
               matrica(i,j)=inf;
         end
    end
end
matrica
dmat=matrica;
fclose(fid);

n = size(dmat,1);
numIter = 10000;
popSize = 4*ceil(popSize/4);
pop = zeros(popSize,n);
pop(1,:) = (1:n);
for k = 2:popSize   %ispisivanje populacije
    pop(k,:) = randperm(n);
end
% GA
globalMin = Inf;
totalDist = zeros(1,popSize);
distHistory = zeros(1,numIter);
tmpPop = zeros(4,n);
newPop = zeros(popSize,n);

for iter = 1:numIter
    % ocena svakog clana populacije odredjivanje rastojanja
    for p = 1:popSize
        d = dmat(pop(p,n),pop(p,1));
        for k = 2:n
            d = d + dmat(pop(p,k-1),pop(p,k));
        end
        totalDist(p) = d;
    end

    % trazenje najbolje rute od predlozenih
    [minDist,index] = min(totalDist);
    distHistory(iter) = minDist;
    if minDist < globalMin
        globalMin = minDist;
        optRoute = pop(index,:);
    end
% GA operatori
    randomOrder = randperm(popSize);
    for p = 4:4:popSize
        rtes = pop(randomOrder(p-3:p),:);
        dists = totalDist(randomOrder(p-3:p));
        [ignore,idx] = min(dists);
        bestOf4Route = rtes(idx,:);
        routeInsertionPoints = sort(ceil(n*rand(1,2)));
        I = routeInsertionPoints(1);
        J = routeInsertionPoints(2);
```

```
for k = 1:4 % Mutacija najbolje za dobijanje 3 nove
            tmpPop(k,:) = bestOf4Route;
            switch k
                case 2 % Flip
                    tmpPop(k,I:J) = tmpPop(k,J:-1:I);
                case 3 % Swap
                    tmpPop(k,[I J]) = tmpPop(k,[J I]);
                case 4 % Slide
                    tmpPop(k,I:J) = tmpPop(k,[I+1:J I]);
                otherwise % Do Nothing
            end
        end
        newPop(p-3:p,:) = tmpPop;
    end
    pop = newPop;
end

index = find(optRoute == start);
finalRoute=optRoute([index:end 1:index-1]);
optRoute;
finalRoute
sprintf('Minimalni put je: %d', minDist)
end
```

3* Algoritam *GA+_poboljsan_sa_Greedy_Euklid(start,popSize)*

```
function [] = GA+_poboljsan_sa_Greedy_Euklid(start,popSize)
tic;

 O = load('euklid10.txt');
[h,dim] = size(O);
  X=O(1:h,2:3);

 m = size(X,1);
[n,dim] = size(X);

D = distmat(X);

s = randperm(n);

Lmin = inf;
for k = 1:m
       p = greedy(s(k),D);
        [p,L] = exchange2(p,D);
     if L < Lmin
        Lmin = L;
        pmin = p;
    end
end

p=pmin;
L = Lmin;

 index = find(p == start);
finalRoute=p([index:end 1:index-1]);
sprintf('Minimalni put Greedy je: %d', L)

dmat=D;

n = size(dmat,1);
numIter = 10000;
popSize = 4*ceil(popSize/4);
pop = zeros(popSize,n);
pop(1,:) = finalRoute;
for k = 2:popSize
    pop(k,:) = randperm(n);
end
```

```
globalMin = Inf;
totalDist = zeros(1,popSize);
distHistory = zeros(1,numIter);
tmpPop = zeros(4,n);
newPop = zeros(popSize,n);

for iter = 1:numIter

    for p = 1:popSize
        d = dmat(pop(p,n),pop(p,1));
        for k = 2:n
            d = d + dmat(pop(p,k-1),pop(p,k));
        end
        totalDist(p) = d;
    end

    [minDist,index] = min(totalDist);
    distHistory(iter) = minDist;
    if minDist < globalMin
        globalMin = minDist;
        optRoute = pop(index,:);
    end

    randomOrder = randperm(popSize);
    for p = 4:4:popSize
        rtes = pop(randomOrder(p-3:p),:);
        dists = totalDist(randomOrder(p-3:p));
        [ignore,idx] = min(dists);
        bestOf4Route = rtes(idx,:);
        routeInsertionPoints = sort(ceil(n*rand(1,2)));
        I = routeInsertionPoints(1);
        J = routeInsertionPoints(2);
        for k = 1:4
            tmpPop(k,:) = bestOf4Route;
            switch k
                case 2
                    tmpPop(k,I:J) = tmpPop(k,J:-1:I);
                case 3
                    tmpPop(k,[I J]) = tmpPop(k,[J I]);
                case 4
                    tmpPop(k,I:J) = tmpPop(k,[I+1:J I]);
                otherwise
            end
        end
        newPop(p-3:p,:) = tmpPop;
    end
    pop = newPop;
end

index = find(optRoute == start);
finalRoutee=optRoute([index:end 1:index-1]);

optRoute;
finalRoutee
 sprintf('Minimalni put je: %d', minDist)

tspplot(finalRoutee,X)

 toc

 function tspplot(p,X,nodenum)

x = X(p,1);
x = [x;x(1)];
y = X(p,2);
y = [y;y(1)];

plot(y,x,'r',y,x,'k.')
grid
axis equal

L = sqrt(diff(x).^2 + diff(y).^2);
str = sprintf('Duzina trase: %g',sum(L));
title(str,'fonts',12)

if nargin > 2
```

```
    for k = 1:numel(p)
        str = sprintf(' %d',p(k));
        text(x(k),y(k),str)
    end
end


 function D = distmat(X)
%DISTMAT Compute euclidian distance matrix from coordinates

[n,dim] = size(X);
D = zeros(n);
for j = 1:n
    for k = 1:dim
        v = X(:,k) - X(j,k);
        D(:,j) = D(:,j) + v.*v;
    end
end
D = sqrt(D);

%-------------------------------------------------------------------------
function p = greedy(s,D)
%GREEDY.

n = size(D,1);
p = zeros(1,n,'uint16');
p(1) = s;

for k = 2:n
    D(s,:) = inf;
    [junk,s] = min(D(:,s));
    p(k) = s;
end

%-------------------------------------------------------------------------
function [p,L] = exchange2(p,D)

n = numel(p);
zmin = -1;
while zmin < 0

    zmin = 0;
    i = 0;
    b = p(n);

    while i < n-2
        a = b;
        i = i+1;
        b = p(i);
        Dab = D(a,b);
        j = i+1;
        d = p(j);
        while j < n
            c = d;
            j = j+1;
            d = p(j);

            z = (D(a,c) - D(c,d)) + D(b,d) - Dab;

            if z < zmin
                zmin = z;
                imin = i;
                jmin = j;
            end
        end
    end

    if zmin < 0
        p(imin:jmin-1) = p(jmin-1:-1:imin);
    end

end

q = double(p);
ind = sub2ind([n,n],q,[q(2:n),q(1)]);
L = sum(D(ind));
```