Rešavanje TSP-a pomoću inteligencije roja

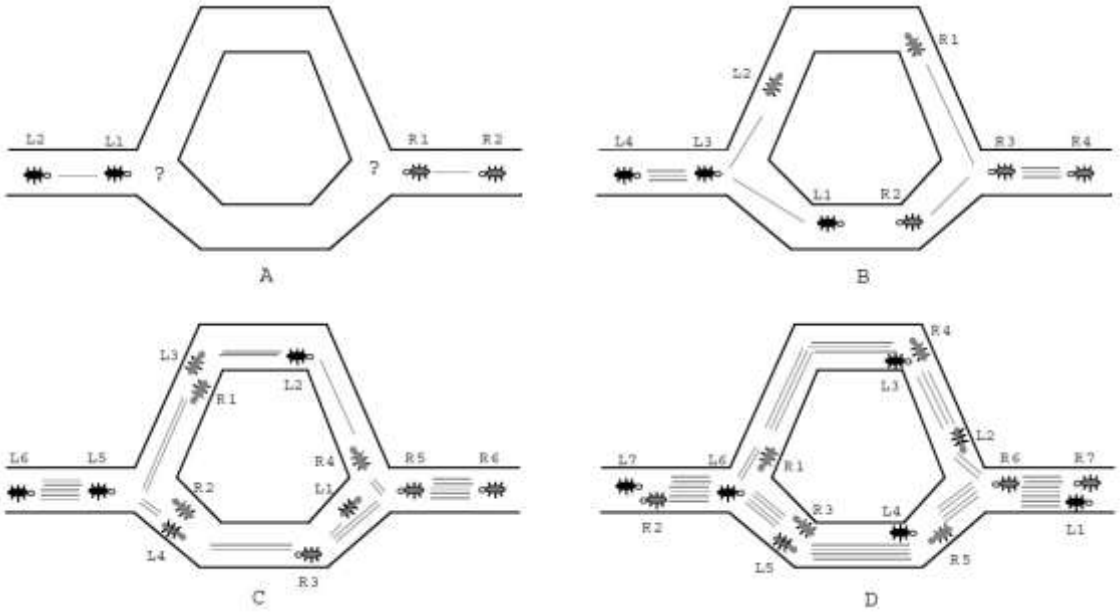
Ognjen Zelenbabić

Definicija problema

Problem: Rešiti TSP na primeru od 30 gradova (Oliver30) za koji su date koordinate. Napisati program u MATLAB-u koji primenjuje jedan od ACO algoritama obrađenih na predavanjima (Ant System ili Ant Colony System). Prikazati graf najkraće dobijene rute sa dužinom pređenog puta.

1. Algoritmi kolonije mrava

Algoritmi mrava su zasnovani na prirodnoj metafori kolonija mrava. Pravi mravi su sposobni da pronađu najkraći put od izvora hrane do njihovog gnezda bez korišćenja vizuelnih informacija već korišćenjem informacija koju dobijaju iz feromona. Dok hodaju, mravi ostavljaju feromon na zemlji, i prate, sa verovatnoćom, feromon koji je prethodno ostavljen od strane drugih mrava. Način na koji mravi koriste feromone da pronađu najkraći put između dve tačke je prikazan na slici 1.



Slika 1: Primer kako pravi mravi pronalaze najkraći put.

Mravi koji kreću sa leve strane su označeni sa L a mravi koji kreću sa desne strane sa R. Na slici 1.A. mravi dolaze do mesta na kome donose odluku. 1.B. Neki od mrava biraju gornji put dok neki biraju donji put. Izbor je nasumičan. 1.C. kako se mravi kreću otprilike konstantnom brzinom, mravi koji izaberu donji, kraći, put dolaze do drugog mesta odlučivanja pre nego što to urade mravi koji su odabrali gornji, duži, put. 1.D. Feromoni se akumuliraju brže na kraćem putu. Broj iscrtanih linija na slikama je otprilike proporcionalan količini feromona koji su mravi ostavili duž puta.

Ovo ponašanje realnih mrava je inspirisalo stvaranje *Ant system*-a, to je algoritam u kome skup veštačkih mrava sarađuje da bi pronašli rešenje problema razmenjujući informacije putem feromona ostavljenim po ivicama grafova (grane između čvorova). Algoritam mrava se primenjuje u kombinatornim optimizacionim problemima kao što je problem trgovačkog putnika.

Pre nego što počnem da opisujem metodologiju kojom sam ja došao do rešenja opisaću razlike između *Ant system*-a (AS) i *Ant colony System*-a. Ovo sam uradio da bi razjasnio drugima kao i sebi razlog iz koga sam odabrao AS. Prvi korak u rešavanju svake problematike je podrobno upoznavanje sa teorijom ciljne oblasti tako da svedem problematiku rešenja na algoritamske a ne idejne i metodološke probleme. Kada sam shvatio šta ću, kada i na koji način upotrebljavati predamnom je bio lak posao „iskucavanja“ koda programa.

1.1. Ant system (AS)

Ant system (sistem mrava) radi na sledećem principu. Svaki mrav generiše kompletnu turu birajući gradove po stohastičkom pravilu prelaznog stanja (***state transition rule***): Mrav preferira da se kreće do gradova koji su povezani kraćim putevima (granama) sa visokom količinom feromona. Kada svi mravi završe svoje ture primenjuje se pravilo globalnog ažuriranja feromona (***global pheromone updating rule***). Ovo pravilo kaže: Delić feromona isparava na svim granama (putevi koji se ne ažuriraju postaju manje poželjni), i tada svaki mrav ostavlja količinu feromona na putevima koji odgovaraju njegovoj turi u odnosu na to koliko je kratka (ili duga) njegova tura bila. Drugim rečima, putevi koji pripadaju mnogim kratkim turama su putevi koji primaju veću količinu feromona. Proces tada prelazi u narednu iteraciju.

Stohastičko pravilo prelaznog stanja, koje se naziva ***nasumično-proporcionalno pravilo*** je dato sledećim izrazom.

$$p_k(r,s) = \frac{[\tau(r,s)]\cdot[\eta(r,s)]^{\beta}}{\sum_{u\in J_k(r)}[\tau(r,u)]\cdot[\eta(r,u)]^{\beta}}, \text{ ako } s \in J_k(r)$$

$$p_k(r,s) = 0, \text{ inače.}$$

Ovaj izraz daje verovatnoću kojom će mrav *k* u gradu *r* odabrati da dođe do grada *s*. τ je količina feromona, $\eta = 1/\delta$ je recipročna vrednost daljine $\delta(r,s)$, $J_k(r)$ je skup gradova koji su preostali da se posete od strane mrava *k* pozicioniranog u gradu *r*, i β je parametar koji određuje relativan značaj uticaja feromona u odnosu na udaljenost $(\beta > 0)$.

U AS-u, pravilo ***globalnog ažuriranja feromona*** je uvedeno na sledeći način. Kada su svi mravi završili svoje ture, feromon se ažurira na svim granama prema sledećem izrazu

$$\tau(r,s) \leftarrow (1-\alpha)\cdot\tau(r,s) + \sum_{k=1}^{m}\Delta\tau_k(r,s),$$

gde je

$$\Delta\tau(r,s) = 1/L_k, \text{ ako } (r,s) \in \text{turi od mrava } k$$

$$0, \text{ inače}$$

$0 < \alpha < 1$ je parametar raspadanja feromona, L_k je dužina ture koju je mrav *k* napravio, i *m* je broj mrava. Smisao formule za ažurirane feromona je da simulira promenu u količini feromona na granama usled dodavanja novih feromona od strane mrava koji su posetili te grane kao i raspadanja (isparavanja) izvesne količine feromona.

Iako je ovaj sistem koristan pri pronalaženju dobrih ili optimalnih rešenja za male skupove gradove (za TSP problem), kada imamo veće skupove gradova on je nedovoljno efikasan, tj. sistem ne konvergira ka najboljem optimalnom putu dovoljno brzo. Iz tog razloga se uvode tri promene u ovaj algoritam da bi se poboljšala njegova delotvornost. Ove promene su objašnjene u *Ant Colony System* algoritmu.

1.2. Ant Colony System (ACS)

ACS se razlikuje od AS-a iz tri glavna razloga:

1. Pravilo prelaznog stanja obezbeđuje direktan način da se balansira između istraživanja novih puteva i eksploatacije akumulisanog znanja o problemu;
2. Pravilo globalnog ažuriranja se primenjuje samo na puteve koji pripadaju najboljoj ruti mrava;
3. Dok mravi konstruišu rešenje primenjuje se pravilo lokalnog ažuriranja feromona.

Sistem kolonije mrava (ACS) funkcioniše na sledeći način: m mrava je inicijalno pozicionirano u n gradova koji su izabrani po nekom inicijalizacionom pravilu (uglavnom nasumično). Svaki mrav pravi svoju turu tako što primenjuje iz koraka u korak pravilo prelaznog stanja za biranje sledećeg grada. Dok mrav konstruiše svoju turu on takođe menja (modifikuje) količinu feromona na posećenim putevima između gradova primenjujući pravilo lokalnog ažuriranja feromona. Kada su svi mravi završili svoje ture, količina feromona na putevima se opet modifikuje tako što se primenjuje pravilo globalnog ažuriranja feromona.

1.2.1. Pravilo prelaznog stanja ACS-a

U sistemu kolonije mrava pravilo prelaznog stanja se uvodi na sledeći način: mrav pozicioniran u gradu *r* bira grad *s* kao sledeći grad prema pravilu koje je dato sledećim izrazom.

$$s = \arg\max_{u \in J_k(r)} \{[\tau(r,u)] \cdot [\eta(r,u)]^{\beta}\}, \text{ ako je } q \le q_0$$

$$s = S, \text{ inače}$$

gde je q nasumični broj sa uniformnom raspodelom u [0...1], q_0 je parametar $(0 \le q_0 \le 1)$, a S je nasumična promenljiva koja se bira prema raspodeli verovatnoće utvrđene prvom formulom sa strane 2.

Pravilo prelaznog stanja koje dobijamo iz pomenute prve jednačine i malopre prikazane jednačine se naziva ***pseudo-nasumično proporcionalno pravilo***. Ovo pravilo favorizuje prelazak ka čvoru koji je povezan kraćim putem i koji ima veću količinu feromona na odgovarajućem putu. Parametar q_0 određuje značaj eksploatacije postojećih znanja naspram istraživanja novih znanja. Svaki put kada mrav iz grada *r* izabere grad *s* u koji će se pomeriti, on odabira nasumični broj $(0 \le q_0 \le 1)$. Ako je $q \le q_0$ tada je izabran najbolji put (eksploatacija), a ako nije zadovoljen pomenuti uslov tada se put bira prema verovatnoći određenoj prvom formulom.

ACS algoritam je simbolički prikazan na sledeći način.

```
petlja
      {svaki mrav se postavlja na početnu poziciju}
      petlja
            {svaki mrav primenjuje pravilo prelaznog stanja da
            bi napravio rešenje, i primenjuje se pravilo
            lokalnog ažuriranja feromona}
      završi petlju {kada svi mravi naprave kompletna rešenja}
      {primenjuje se pravilo globalnog ažuriranja feromona}
završi petlju {završni uslov}
```

1.2.2. Pravilo globalnog ažuriranja feromona ACS-a

U sistemu kolonije mrava, samo je najboljem mravu (mrav koji je napravio najbolju turu od početka programa) dozvoljeno da ostavlja feromon duž svog puta. Ovaj izbor, zajedno sa upotrebom pseudo nasumičnog proporcionalnog pravila je namenjeno usmeravanju prostora pretraživanja (mravi traže samo u susedstvu najbolje pronađene ture). Pravilo globalnog ažuriranja feromona se primenjuje nakon što su svi mravi završili svoje ture. Nivo feromona se ažurira primenom pomenutog pravila na sledeći način:

$$\tau(s,r)\ \tau(r,s) \leftarrow (1-\alpha) \cdot \tau(r,s) + \alpha \cdot \Delta\tau(r,s),$$

gde je

$$\Delta\tau(r,s) = (L_{gn})^{-1}, \text{ ako } (r,s) \in \text{globalno-najboljoj turi}$$

$$\Delta\tau(r,s) = 0, \text{ inače.}$$

$0 < \alpha < 1$ je parametar raspadanja feromona (isparavanja), a L_{gn} je dužina globalno najbolje ture od početka programa. Cilj ovog pravila je da obezbedi ostavljanje veće količine feromona na kraćim turama. Poslednja jednačina diktira da samo putevi koji pripadaju globalno najboljoj turi dobiju pojačanje u vidu feromona.

1.2.3. Pravilo lokalnog ažuriranja feromona ACS-a

Dok nalaze rešenje TSP-a mravi posećuju puteve i menjaju njihove nivoe feromona primenjujući pravilo lokalno ažuriranja feromona.

$$\tau(r,s) \leftarrow (1-\rho)\cdot\tau(r,s)+\rho\cdot\Delta\tau(r,s),$$

gde je $0<\rho<1$ parametar. Za biranje vrednosti $\Delta\tau(r,s)$ imamo tri opcije, (i) $\Delta\tau(r,s)=0$ (daje najgore rezultate), tj. u ovom slučaju nema lokalnog ažuriranja feromona, (ii) $\Delta\tau(r,s)=\tau_0$, i (iii) $\Delta\tau(r,s)=\gamma\max_{z\in J_k(s)}\tau(s,z)$ gde je $0\le\gamma\le1$ parametar (formula je identična kao formula za Q-obučavanje).

2. Pristup rešavanju problema

Iako ACS deluje kao bolji izbor za rešavanje optimizacionih problema, ja sam se ipak odlučio za AS i to samo zbog nedostatka vremena koje sam imao da posvetim ovoj problematici.

AS jeste prostiji od dva algoritma ali i on takođe zahteva podešavanje određenog broja parametara koji direktno utiču na brzinu i kvalitet pronalaženja rešenja. Za odabir ovih parametara ne postoje egzaktna pravila, već pravila empirijski utvrđena nakon velikog broja eksperimenata.

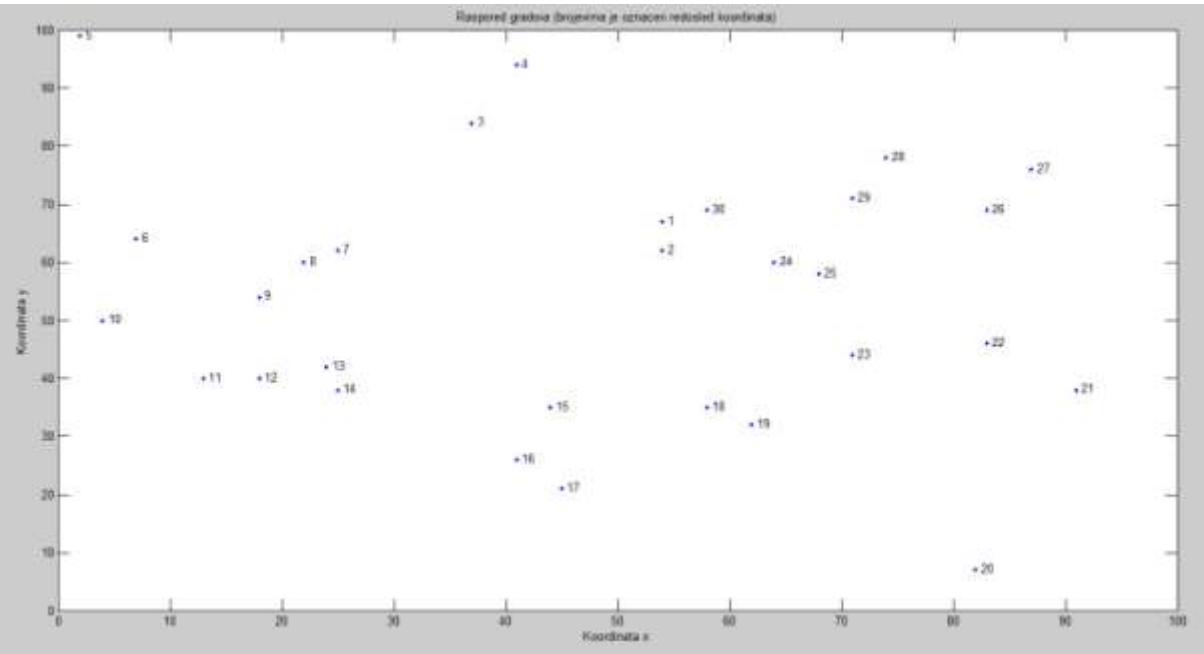
Nakon što sam završio svoj algoritam za AS posvetio sam izvesno vreme „štimanju“ rešenja, odnosno podešavanju parametara i posmatranju zavisnosti rešenja od vrednosti ovih parametara. Kako sam napisao algoritam, koje sam parametre odabrao i modifikovao, opisao sam u narednom tesktu.

Za razliku od druga dva zadatka koja sam imao da uradim, početni podaci su ovde bili najmanji problem jer su tekstom zadatka već zadati. U početku programa inicijalizovao sam koordinate gradova i izračunao dužine svih puteva između njih a nakon toga sam izračunao i η koje predstavlja poželjnost određenog puta a definiše se najčešće kao recipročna vrednost dužine.

```
% Koordinate gradova
x=[...
    54 54 37 41  2  7 25 22 18  4 13 18 24 25 44 ...
    41 45 58 62 82 91 83 71 64 68 83 87 74 71 58];
y=[...
    67 62 84 94 99 64 62 60 54 50 40 40 42 38 35 ...
    26 21 35 32  7 38 46 44 60 58 69 76 78 71 69];

% Racunam rastojanja izmedju svih gradova i zelju za granom (ni)
d=zeros(br_grd,br_grd);
ni=zeros(br_grd,br_grd);
for i=1:br_grd
    for j=1:br_grd
        d(i,j)=sqrt((x(j)-x(i))^2+(y(j)-y(i))^2);
        if (d(i,j)==0)
            ni(i,j)=0;
        else
            ni(i,j)=1/d(i,j);
        end
    end
end
```

Raspored gradova zadatih koordinatama je prikazan na sledećoj slici (slika 2).



Slika 2: Geometrijski raspored gradova za TSP.

Oslanjajući se na gore navedenu teoriju, svoj algoritam sam koncipirao na sledeći način:

```
{inicijalizujem početne podatke}
for 1:broj_iteracija
for 1:broj_mrava
{biram početnu poziciju mrava}
{mrav obilazi kompletan put}
end
{ažuriram količine feromona na svim putevima}
{računam lokalni minimum}
end
{računam globalni minimum}
```

Kompletan algoritam sam napisao dva puta na dva različita načina misleći da dobijam pogrešno rešenje zahvaljujući grešci u kodu koju sam mislio da imam. Nakon izrade drugog algoritma na potpuno drugačiji način sam dobijao identična rešenja. Analizirao sam svoj pristup rešavanju problema i uvideo da u pristupu ne postoji nedoslednost (ili je ja barem nisam mogao uočiti). U svakom slučaju, primenom AS algoritma nisam mogao dobiti konvergenciju niti rešenje kakvo sam ja želeo da vidim.

Prvi put sam program pisao tako da svaku celinu stavim u zasebnu funkciju, međutim, zbog bolje preglednosti koda, drugi put sam se odlučio da kompletan algoritam pišem u jednom „prozoru“.

Davajući početne parametre, Vi ste verovatno zamislili da se algoritam radi pomoću ACS-a pošto sam ja sa drugim setom početnih parametara dobijao bolja rešenja. Da ne dužim, objašnjenje algoritma koji sam napisao sledi.

3. Odabir putanje mrava

Prvo sam svim mravima koji učestvuju u eksperimentu nasumično dodelio početnu poziciju. Odabirom početne pozicije znao sam i njihovu poslednju poziciju koju sam odmah memorisao.

```
% Svaki mrav nasumicno zauzima pocetnu poziciju
ppm=fix(1+br_grd*rand(1)); % Pocetna Pozicija Mrava
putanja(m,1)=ppm; % Pocetna pozicija mrava u putanji
putanja(m,br_grd+1)=putanja(m,1); % Poslednja pozicija mrava u putanji
```

Sledeći korak mi je donekle sličan kao u algoritmu koji sam napisao za GA, tj. napravio sam matricu preostalih neposećenih gradova (sa i bez nula). Kao i kod GA ovim korakom sam si skratio posao na drugom mestu u algoritmu.

```
% Pravim matricu preostalih gradova koje mrav mora da poseti (sa nulama)
ostali_grd_0=zeros(1,br_grd);
k=0; % Pomocni brojac
for i=1:br_grd
    if (i~=ppm)
        k=k+1;
        ostali_grd_0(1,k)=i;
    end
end
% Pravim matricu preostalih gradova koje mrav mora da poseti (bez nula)
ostali_grd=zeros(1,k);
for i=1:k
    ostali_grd(1,i)=ostali_grd_0(1,i);
end
```

Nakon toga sam proračunao verovatnoću izbora svih preostalih gradova prema nasumičnom-proporcionalnom pravilu na sledeći način:

```
% Verovatnoca da mrav poseti sledeci grad
suma=0;
for i=1:k % Broj preostalih gradova da se posete
   suma=suma+(tau(ppm,ostali_grd(i)).^alfa).*(ni(ppm,ostali_grd(i)).^beta);
end
p=zeros(1,br_grd); % Verovatnoca da mrav poseti jedan od preostalih gradova
for i=1:k
    privr=((tau(ppm,ostali_grd(i)).^alfa).*(ni(ppm,ostali_grd(i)).^beta));
    p(1,ostali_grd(i))=privr/suma;
end
```

Posle proračunavanja verovatnoća izbora sledećeg grada birao sam grad sa najvećom verovatnoćom, odnosno onaj put koji je najkraći i ima najviše feromona.

```
% Odabir sledeceg grada u koji mrav ide
max_p=max(p); % Trazim maksimalnu verovatnocu
for i=1:br_grd
    if (p(1,i)==max_p)
        % Odabran je sledeci grad (grad sa najvecom verovatnocom)
        sled_grd=i;
        putanja(m,n)=sled_grd; % Azuriram informacije o putanji
        % Azuriram duzinu putanje
        duz_putanje(m,1)=duz_putanje(m,1)+d(ppm,sled_grd);
        % Azuriram pocetnu poziciju mrava
        ppm=sled_grd;
    end
end
```

Odabrao sam sledeći neposećeni grad, ažurirao putanju (dodao odabrani grad putanji), ažurirao dužinu putanje (sabrao dužinu puta od trenutnog do sledećeg grada sa dotadašnjim pređenim putem) i setovao početnu poziciju mrava u grad koji je odabran kao sledeći.

Nakon toga sam ažurirao listu (matricu) preostalih gradova koje mrav treba da poseti.

```
% Azuriram listu preostalih gradova koje mrav ima da poseti
ostali_grd_1=zeros(1,k-1);
broj=k;
k=0; % Pomocni brojac
for i=1:broj
    if (ostali_grd(1,i)~=sled_grd)
        k=k+1;
        ostali_grd_1(1,k)=ostali_grd(1,i);
    end
end
ostali_grd=zeros(1,k);
for i=1:k
    ostali_grd(1,i)=ostali_grd_1(1,i);
end
```

Sa ovim korakom je gotov deo algoritma koji se odnosi na biranje puta mrava, odnosno, svaki mrav je odabrao svoju kompletnu putanju (rutu). Preostalo je još samo da dodam pređenom putu svakom mrava i udaljenost od poslednjeg grada do grada iz koga je mrav počeo svoj put kao i da primenim pravilo globalnog ažuriranja feromona.

```
% Azuriram duzinu putanje koju je svaki mrav presao
for m=1:br_mrava
    duz_putanje(m,1)=duz_putanje(m,1)+d(putanja(m,1),putanja(m,br_grd));
end
```

4. Pravilo globalnog ažuriranja feromona

Pre nego što prikažem pravilo globalnog ažuriranja feromona trebao bi da napomenem da sam usvojio da je matrica feromona simetrična. Simetrična je u tom smislu da kada mrav pređe put od grada *r* ka gradu *s* feromon ostavljam na $\tau(r,s)$ kao i na $\tau(s,r)$.

```
% Primenjujem pravilo globalnog azuriranja feromona
for m=1:br_mrava
    for i=1:br_grd
        for j=1:br_grd
            for f=1:br_grd
                if (i==putanja(m,f) && j==putanja(m,f+1))
                    delta_tau(i,j)=delta_tau(i,j)+1/duz_putanje(m,1);
                    delta_tau(j,i)=delta_tau(j,i)+1/duz_putanje(m,1);
                end
            end
        end
    end
end

% Azuriram kolicinu feromona na svim granama
for i=1:br_grd
    for j=1:br_grd
        tau(i,j)=(1-ro)*tau(i,j)+delta_tau(i,j);
        tau(j,i)=(1-ro)*tau(j,i)+delta_tau(j,i);
    end
end
```

Pravilo globalnog ažuriranja feromona sam takođe napisao koristeći se navedenom teorijom. U ovom delu programa se nalazi najveće usporenje, četiri ugneždene *for* petlje. Pretpostavljam da je rešenje moglo biti elegantnije ali se sa optimizacijom samog algoritma nisam mnogo bavio.

Ostao je samo još jedan deo iteracije koji obuhvata računanja lokalnog minimuma.

```
% Trazim lokalni minimum
min_duz_putanje(a,1)=min(duz_putanje);
for m=1:br_mrava
    if (duz_putanje(m,1)==min_duz_putanje(a,1))
        lok_min(a,:)=putanja(m,:);
    end
end
```

Do sada prikazani deo programa mi čini jednu iteraciju. Nakon toga sam pronašao globalni minimum i iscrtao rešenje. Algoritam za traženje globalnog minimuma je sledeći:

```
% Trazim globalni minimum
globalni_minimum=zeros(1,br_grd+1);
duzina_optimalnog_puta=min(min_duz_putanje);
for i=1:br_iter
    if (duzina_optimalnog_puta==min_duz_putanje(i,1))
        globalni_minimum(1,:)=lok_min(i,:);
    end
end
```

Kompletan algoritam za AS sam naveo u dodatku A.

5. Analiza rešenja

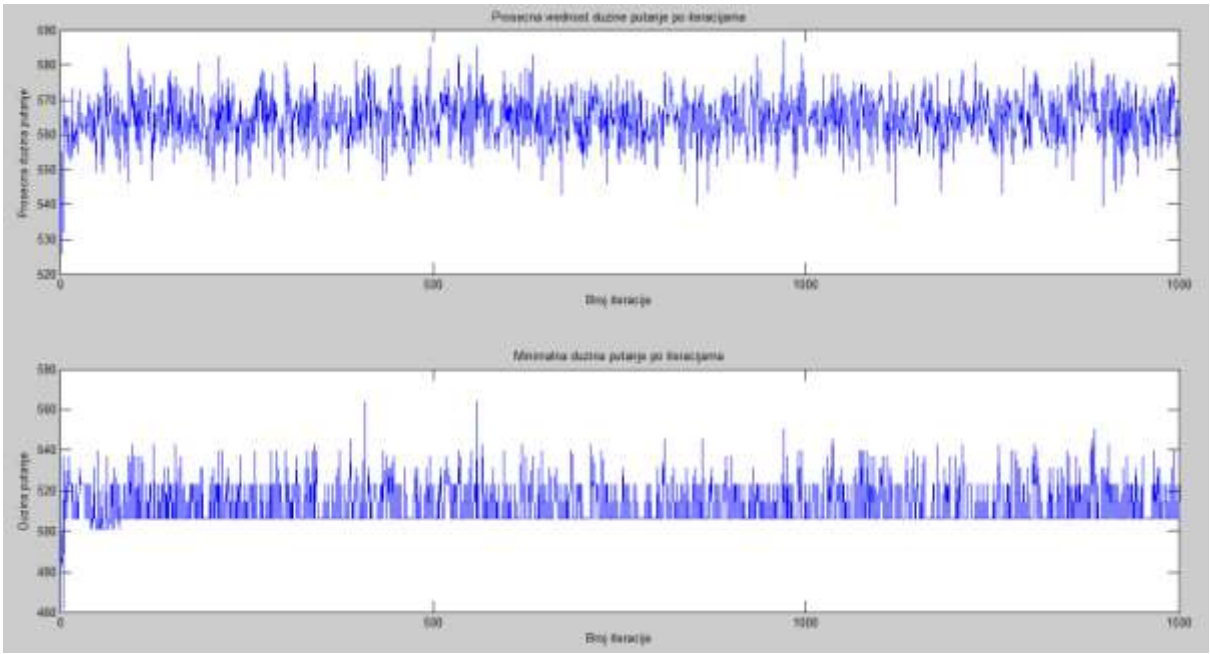
Kao što sam već naveo, izvesne parametre sam menjao da bi pronašao najbolje rešenje. Primetio sam da je AS algoritam robustan i da su potrebne relativno velike promene određenih parametara kako bi se došlo do drugačijeg rešenja. Broj mrava ne utiče mnogo na rešenje ako se ima u vidu da je broj mrava dovoljno veliki. Slična rešenja sam dobijao sa 20 i sa 200 mrava. Ipak, razlike u dužini izvršavanja programa su znatne za različit broj mrava pa sam iz tog razloga odabrao broj od 15 mrava.

Izvesne analize pokazuju da se AS i ACS uspešno rešavaju i sa 10 mrava. Broj iteracija je bio fiksan jer je tako zadato uslovom zadatka.

Koliko god pokušavao da podesim parametre ipak nisam uspeo da dobijem zadovoljavajuću konvergenciju minimalne dužine optimalnog puta i prosečne dužine puta mrava (prosečna dužina puteva svih mrava tokom jedne iteracije).

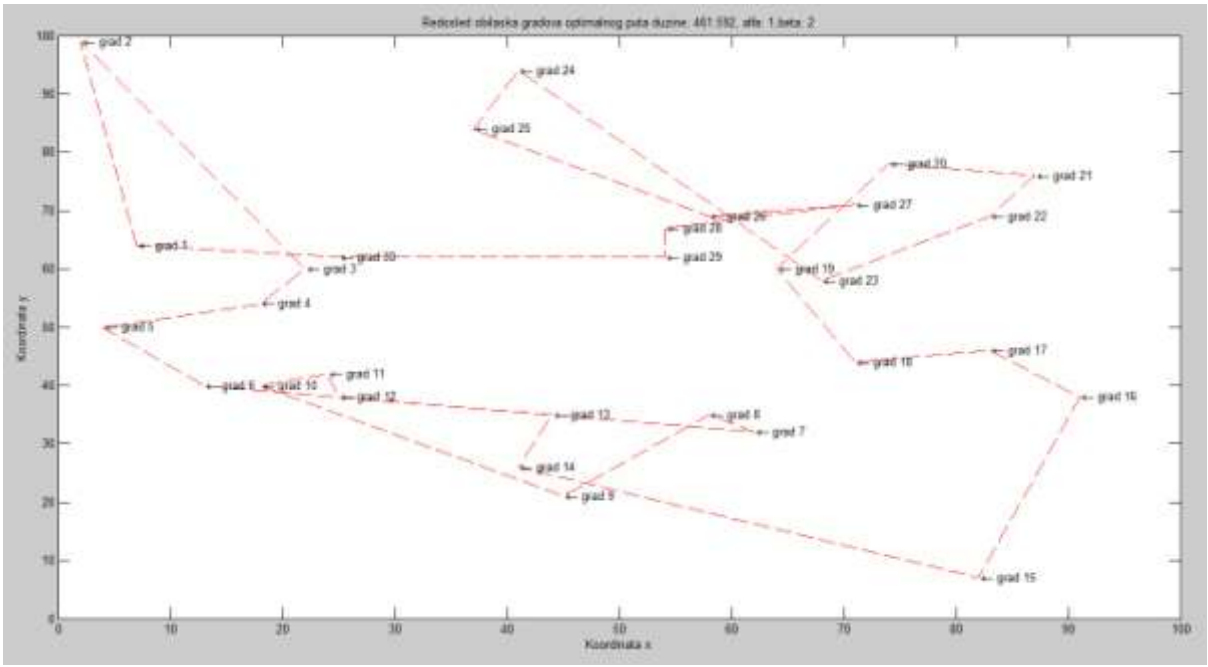
Na sledećim slikama sam prikazao uobičajeno rešenje za trenutni algoritam. Parametri su podešeni na sledeći način: alfa=1, beta=2, ro=0.5, tau_0=1. Prva slika prikazuje prosečnu dužinu putanje mrava u odnosu na broj iteracije i minimalnu dužinu putanje svih mrava u odnosu na broj iteracije (slike 3 i 4).

Na slikama 5 i 6 je prikazano rešenje istog algoritma za drugačije odabrane parametre: alfa=2, beta=1, dok je sve drugo isto.

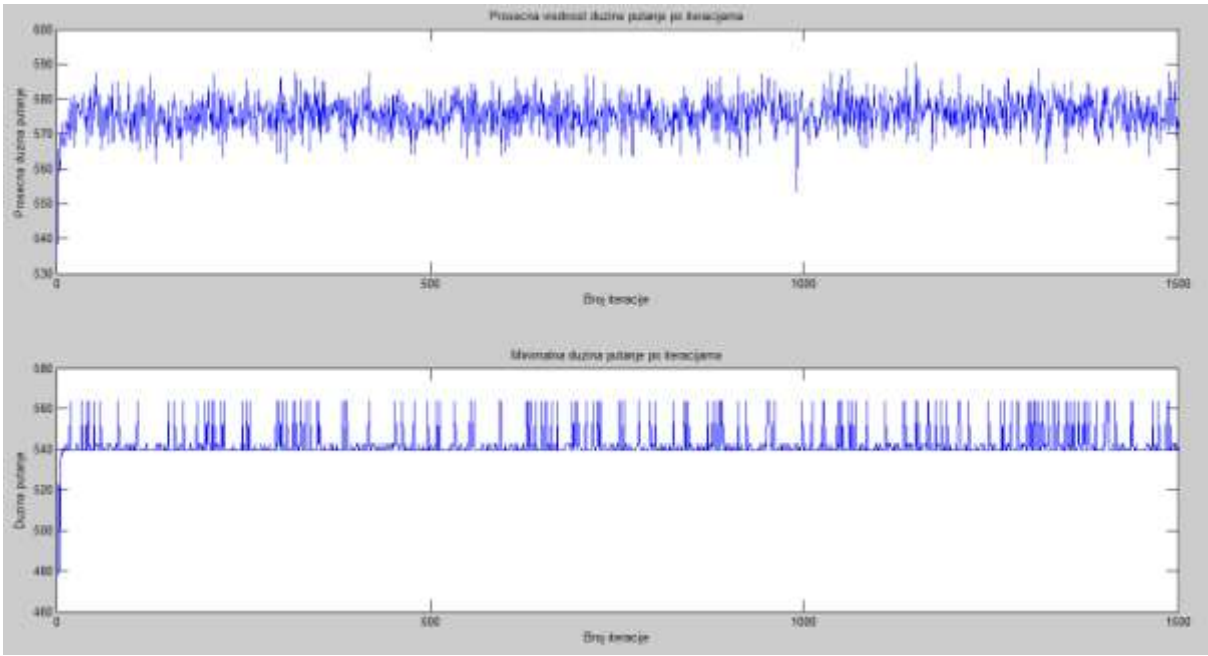


Slika 3: Prvi primer loše konvergencije rešenja za AS.

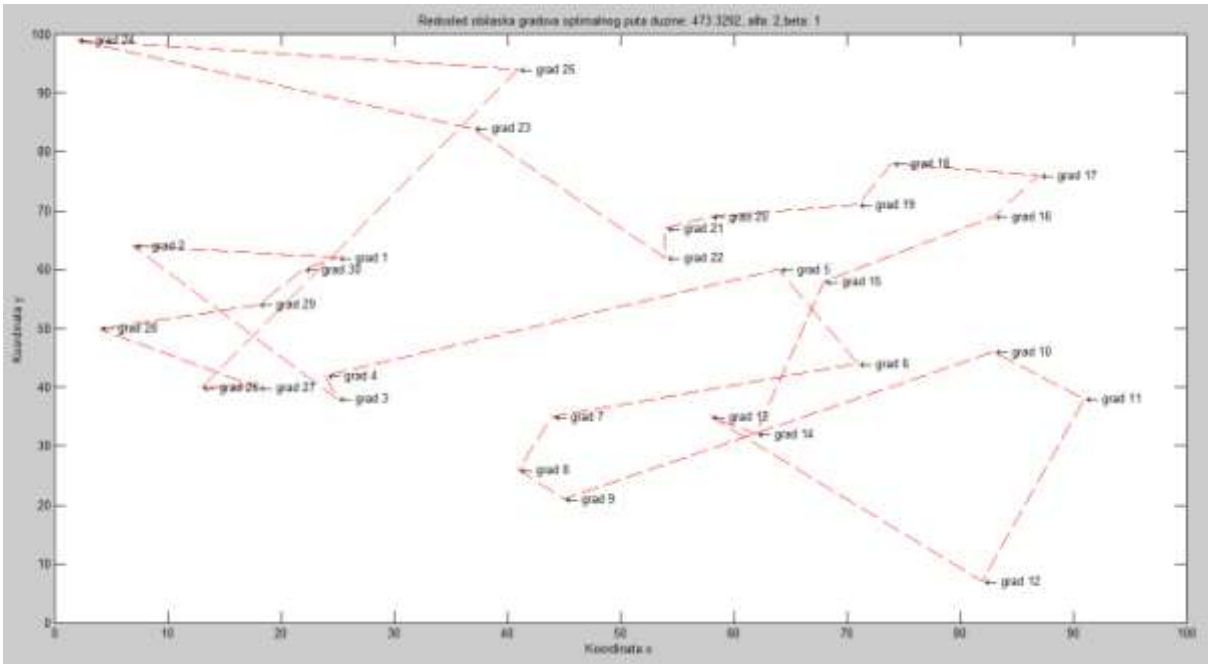
Minimalna putanja i njena dužina za isti slučaj je prikazana na sledećoj slici (slika 4).



Slika 4: Prvi primer loše minimalne putanje (optimalni put) za AS.



Slika 5: Drugi primer loše konvergencije rešenja za AS.



Slika 6: Drugi primer loše minimalne putanje (optimalni put) za AS.

Pretpostavljam da niste zadovoljni prikazanom konvergencijom i rešenjima. Pokušavao sam na razne načine da podesim parametre kako bi dobio bolje rešenje ali to sa ovim algoritmom jednostavno nije moguće ili ja ne znam to da izvedem. U svakom slučaju, razvio sam još jednu verziju koda koja mi je na kraju dala rešenje sa kojim sam bio zadovoljan.

Dok sam pisao prvi algoritam intuitivno mi je bilo jasno da se algoritam može unaprediti na nekolino načina, međutim, nisam hteo da odstupam od teorijske forme rešavanja AS problema. Na kraju sam, ipak, nezadovoljan performansama projektovanog sistema bio prisiljen da dodam još nekoliko linija koda. Te izmene sam opisao u sledećem poglavlju.

6. Unapređeni AS ili ACS?

Razlike između novog i prethodnog algoritma nisu velike. U ovom, kako sam ga ja nazvao, unapređenom AS-u sam uveo i lokalno ažuriranje feromona (ne samo globalno) kao i ažuriranje feromona mrava koji je napravio najbolju putanju. Evo o čemu se radi.

Lokalno ažuriranje feromona sam uveo da bi ojačao količinu feromona na putevima koje su mravi prešli. Ovo sam uradio jer sam primetio da ojačanje predviđeno globalnim ažuriranjem feromona obrađeno u teoriji za AS sa početka ovog teksta nije dovoljno (feromoni se dodaju u malim količinama). Rezultat ovakvog globalnog ažuriranja je takav da feromoni nisu dovoljno jaki da privuku mrave na "utabane staze". Mogao sam i pojačati uticaj feromona na pronalaženje rešenja povećavajući faktor alfa, ali to takođe nije dalo zadovoljavajuće rešenje.

Lokalno ažuriranje sam izveo na sledeći način.

```
% Odabir sledeceg grada u koji mrav ide
    max_p=max(p); % Trazim maksimalnu verovatnocu
    for i=1:br_grd
        if (p(1,i)==max_p)
            % Odabran je sledeci grad (grad sa najvecom verovatnocom)
            sled_grd=i;
            % Azuriram informacije o putanji
            putanja(m,n)=sled_grd;
            % Azuriram duzinu putanje
            duz_putanje(m,1)=duz_putanje(m,1)+d(ppm,sled_grd);
           →% Uvodim lokalno azuriranje feromona
           →tau(ppm,sled_grd)=(1-ro)*tau(ppm,sled_grd)+ro*(1/duz_putanje(m,1));
            % Azuriram pocetnu poziciju mrava
            ppm=sled_grd;
        end
    end
```

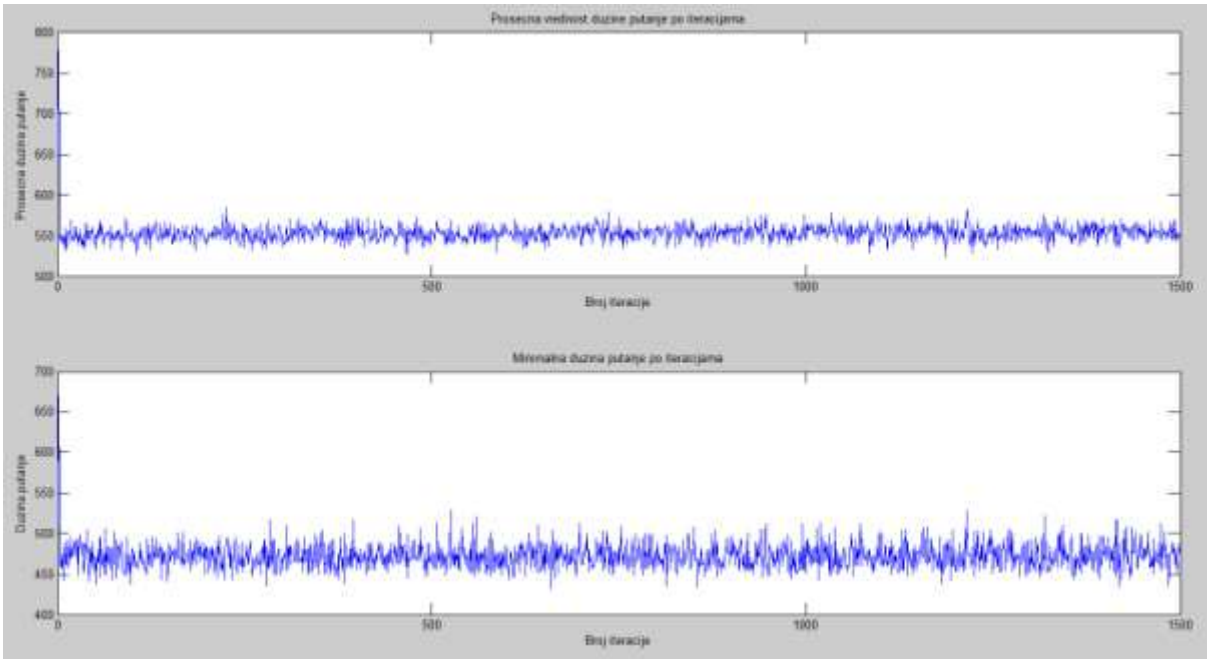
Dakle, dodatnu liniju koda sam ubacio u deo koda koji se odnosi na biranje sledećeg grada. Jedinu razliku predstavlja linija za lokalno ažuriranje feromona. Izraz koji sam ubacio kaže sledeće: na postojeću vrednost feromona na datom putu, pomnoženom sa (1-ro), sabrao sam recipročnu vrednost pređenog puta koju sam pomnožio sa parametrom ro.

Još jedan dodatak prvobitnom algoritmu je ojačanje feromona na putanji mrava koji je napravio najbolju rutu (optimalni put). Odnosno, ojačao sam puteve koji pripadaju putanji lokalnog minimuma na sledeći način.

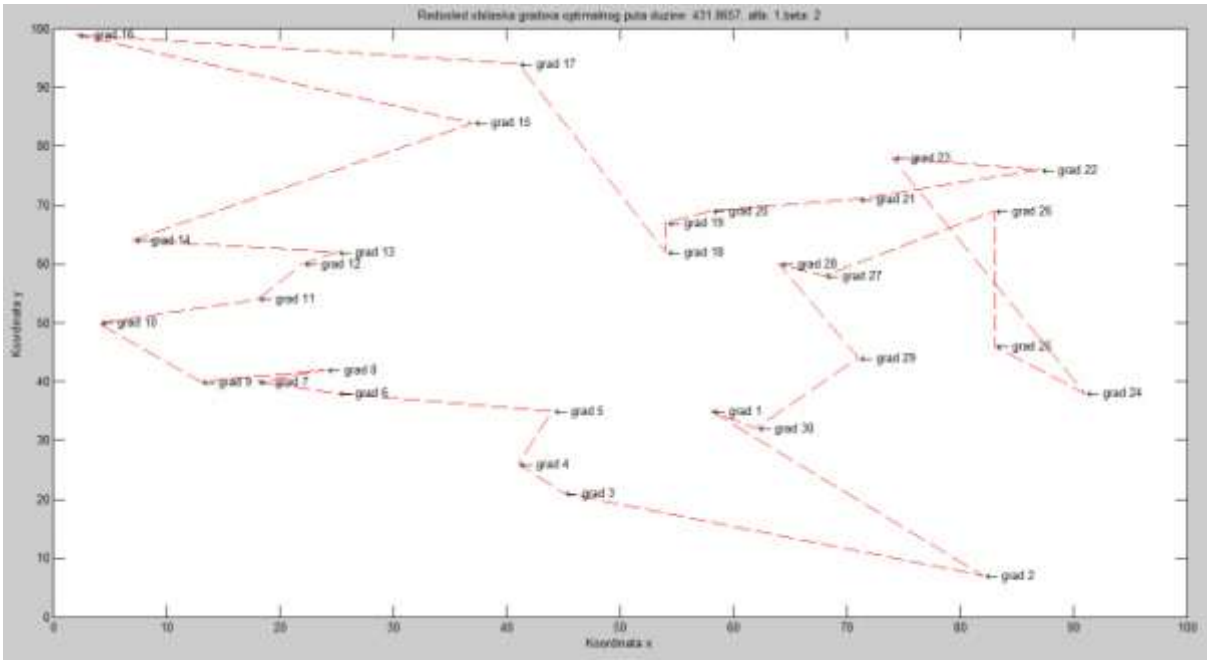
```
% Dodajem dodatne feromone na putanju lokalnog minimuma
    for i=1:br_grd
        pom=tau(lok_min(a,i),lok_min(a,i+1));
        tau(lok_min(a,i),lok_min(a,i+1))=(1-ro)*pom;
    end
```

Sa ovim izmenama moj AS algoritam neodoljivo podseća na ACS algoritam. Sa uvedenim izmenama moj algoritam sada pokazuje mnogo bolju konvergenciju rešenja i bolji optimalni put sa parametrima podešenim na isti način kao u do sada prikazanim primerima (slike 3,4,5 i 6).

Na slikama 7 i 8 sam prikazao rešenje prvog primera (za parametre alfa=1 i beta=2), dok sam na slikama 9 i 10 prikazao rešenje drugog primera (za parametre alfa=2 i beta=1).

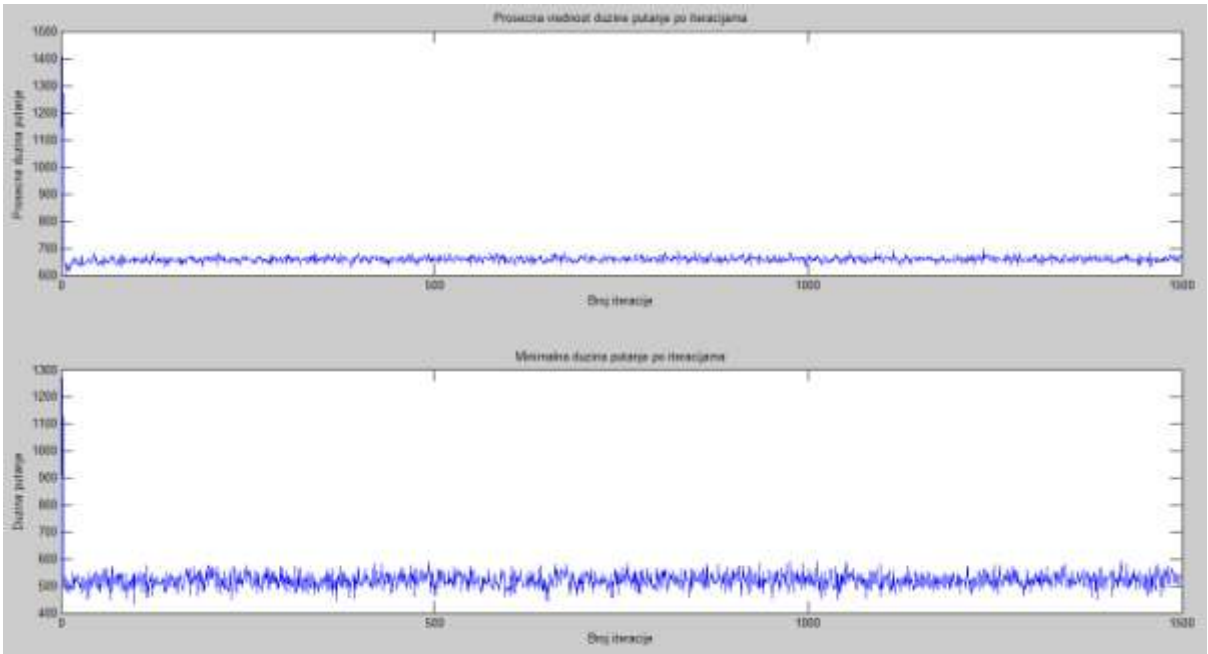


Slika 7: Prvi primer dobre konvergencije rešenja za izmenjeni AS.

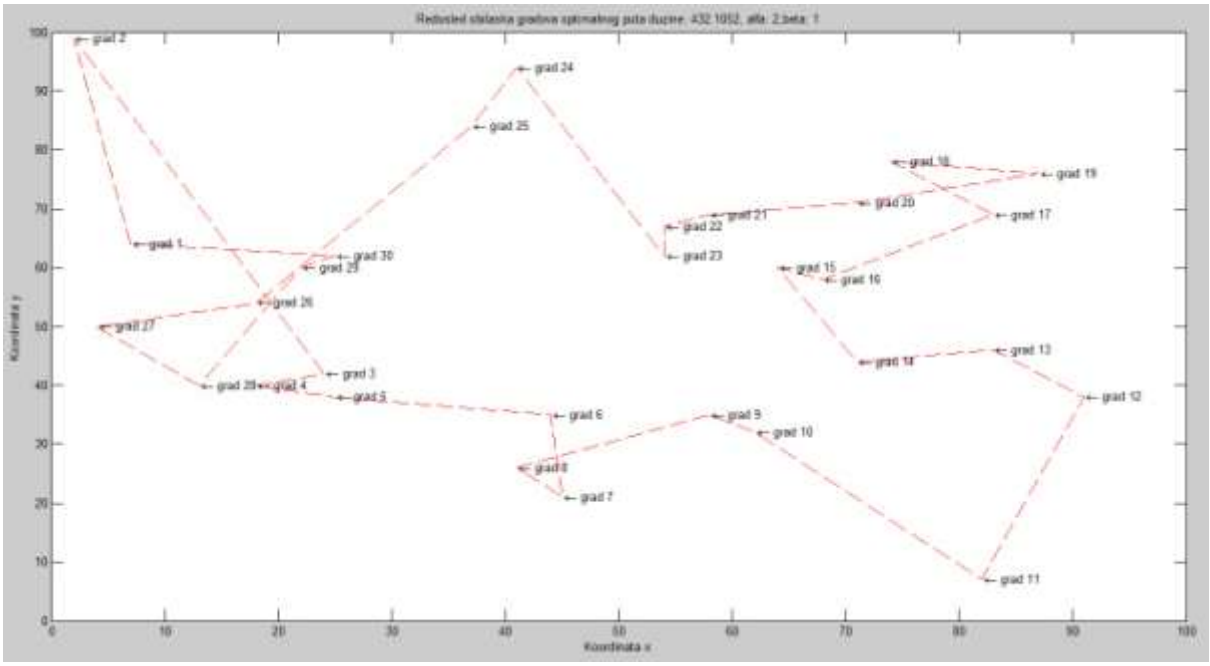


Slika 8: Prvi primer dobre minimalne putanje (optimalni put) za izmenjeni AS.

Drugi primer, za parametre podešene na alfa=2, beta=1.



Slika 9: Drugi primer dobre konvergencije rešenja za izmenjeni AS.



Slika 10: Drugi primer dobre minimalne putanje (optimalni put) za izmenjeni AS.

Algoritam pokazuje bolju konvergenciju i bolji optimalni put u drugom slučaju, tj. kada je alfa=2 i beta=1. Po teoriji ovaj slučaj (*Oliver30*) je granični slučaj za upotrebu AS-a, odnosno, upotreba AS-a se preporučuje za manji broj gradova (do 30) dok se za veće skupove problema preporučuje ACS.

Slučaj koji sam ja analizirao je pokazao da bolje ponašanje ispoljava sistem projektovan ACS-om (iako moj ACS nije kompletan). Međutim, iako je konvergencija bila loša u slučaju AS-a ipak sam i sa tim algoritmom dobijao dobre putanje za optimalan put (putanje malih dužina ukupnog pređenog puta). AS sam morao pokretati više puta kako bi našao dobro rešenje, dok se konvergencija od slučaja do slučaja nije mnogo poboljšavala (izgledala je manje-više isto).

Sa druge strane, primetio sam da je ACS algoritam manje robustan, pa tako manje promene parametara izazivaju velike promene u odzivu sistema pa je samim tim oblast podešavanja parametara za

poboljšanje rešenja manja nego kod AS-a. To ipak i nije velika mana jer i pored manjeg opsega za podešavanje parametara ovaj algoritam ipak pokazuje bolje ponašanje ”na svim frontovima”.

7. Poslednja modifikacija algoritma

Posle ubačenih dodatnih linija koda u moj originalni algoritam za AS dobijao sam dobra rešenja za optimalni put i donekle i za konvergenciju rezultata. Ipak, prikazani grafici konvergencije me nisu zadovoljili. Koliko god da sam pokušavao sa trenutnim algoritmom da se usmerim na eksploataciju (korišćenje akumuliranih znanja), uticaj eksploracije je bio jači nego što bi ja to želeo.

Da bi zadovoljio i svoja očekivanja vezana za konvergenciju ubacio sam još jedan dodatni izraz u algoritam, čime je algoritam koji sam napisao postao u potpunosti ACS.

U trenutni algoritam sam ubacio pseudo-nasumično proporcionalno pravilo koje sam objasnio u teoriji na početku ovog dokumenta. Modifikacija nije bila velika, trebalo mi je vrlo malo vremena da napišem, ovih dodatnih, nekoliko linija koda.

Odabiranje sledećeg grada sam smestio u zasebnu funkciju koju sam nazvao *biranje_acs*, ta funkcija je realizovana na sledeći način.

```
function [sled_grd]=biranje_acs(p,k,i)
% Odabran je sledeci grad (grad sa najvecom verovatnocom)
sled_grd=i;
q=rand(1);
suma=0;
for i=1:k
    suma=suma+p(1,i);
    if q<=suma
        sled_grd=i;
        break
    end
end
```

A samu funkciju sam u postojeći algoritam ubacio na sledeće mesto.

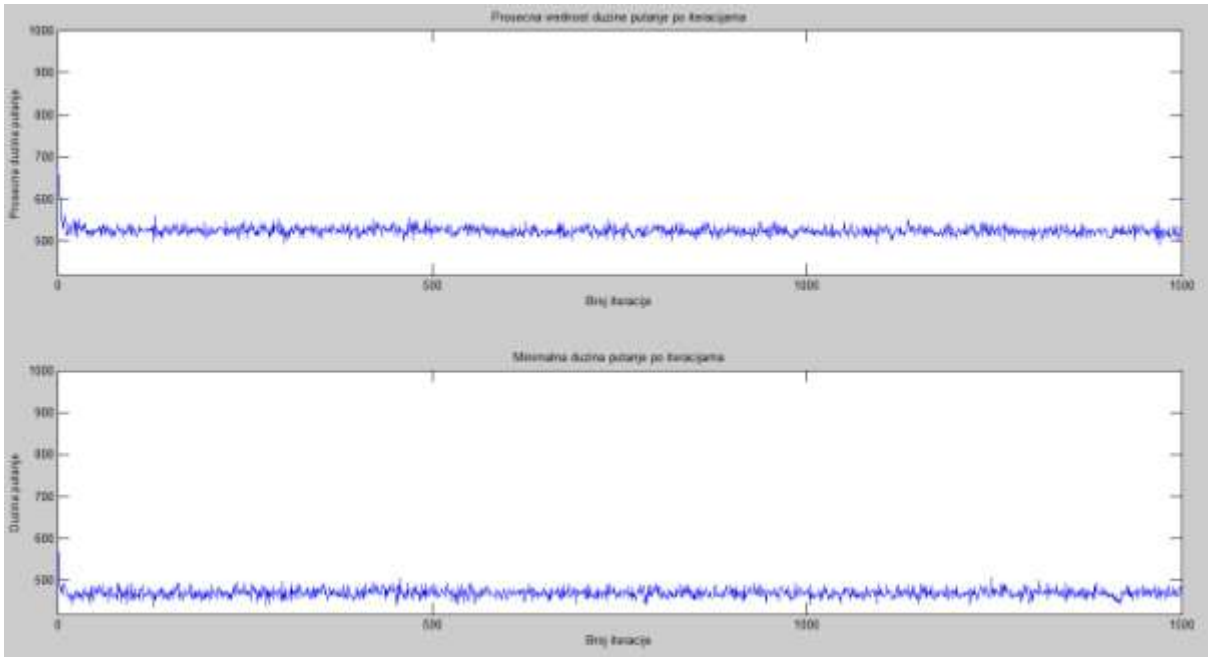
```
% Odabir sledeceg grada u koji mrav ide
max_p=max(p); % Trazim maksimalnu verovatnocu
for i=1:br_grd
    if (p(1,i)==max_p)
       →% Pozivam funkciju za biranje sledeceg grada
       →[sled_grd]=biranje_acs(p,k,i);
        % Azuriram informacije o putanji
        putanja(m,n)=sled_grd;
        % Azuriram duzinu putanje
        duz_putanje(m,1)=duz_putanje(m,1)+d(ppm,sled_grd);
        % Azuriram pocetnu poziciju mrava
        ppm=sled_grd;
    end
end
```

8. Kvalitetno rešenje TSP-a

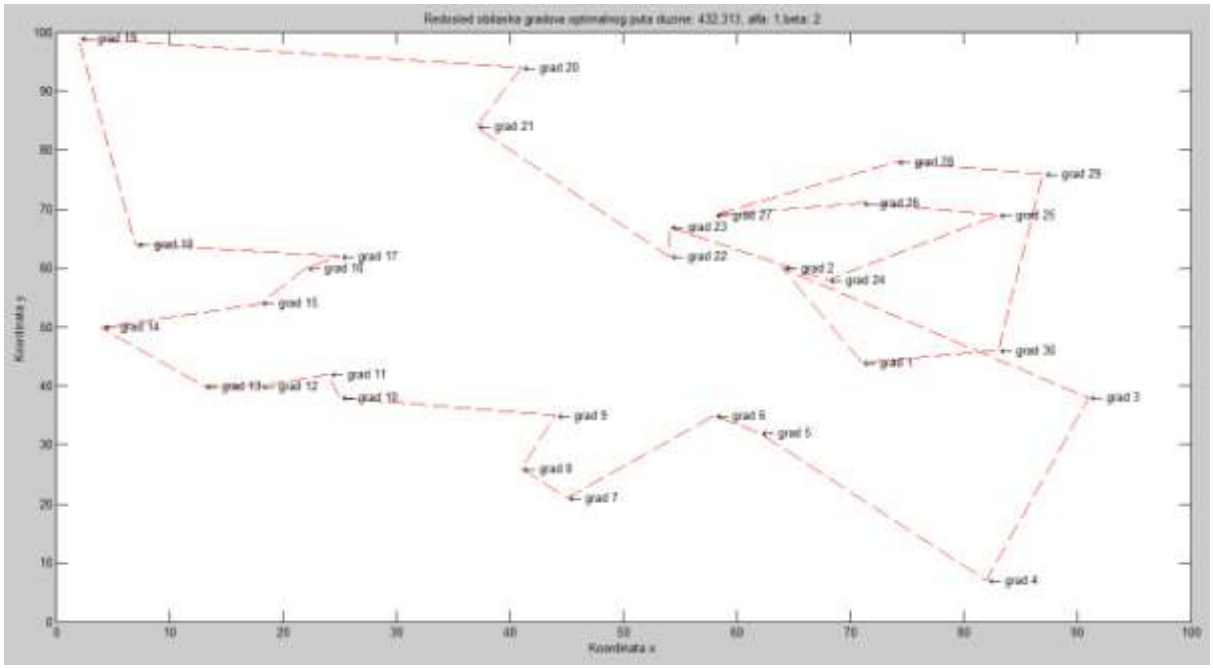
Prikazaću slike koje sam dobio kao rešenje problema trgovačkog putnika za parametre podešene na isti način kao i u prethodnim primerima. Ovo sam uradio kako bi upoređivanje kvaliteta rešenja bilo jednostavnije.

Optimalni put u naredna dva primera je znatno bolji (kraći) od optimalnog puta dobijanog dosadašnjim algoritmom.

Na prve dve slike sam prikazao izlaz sistema za alfa=1 i beta=2 dok su svi ostali parametri podešeni na iste vrednosti kao i do sada.

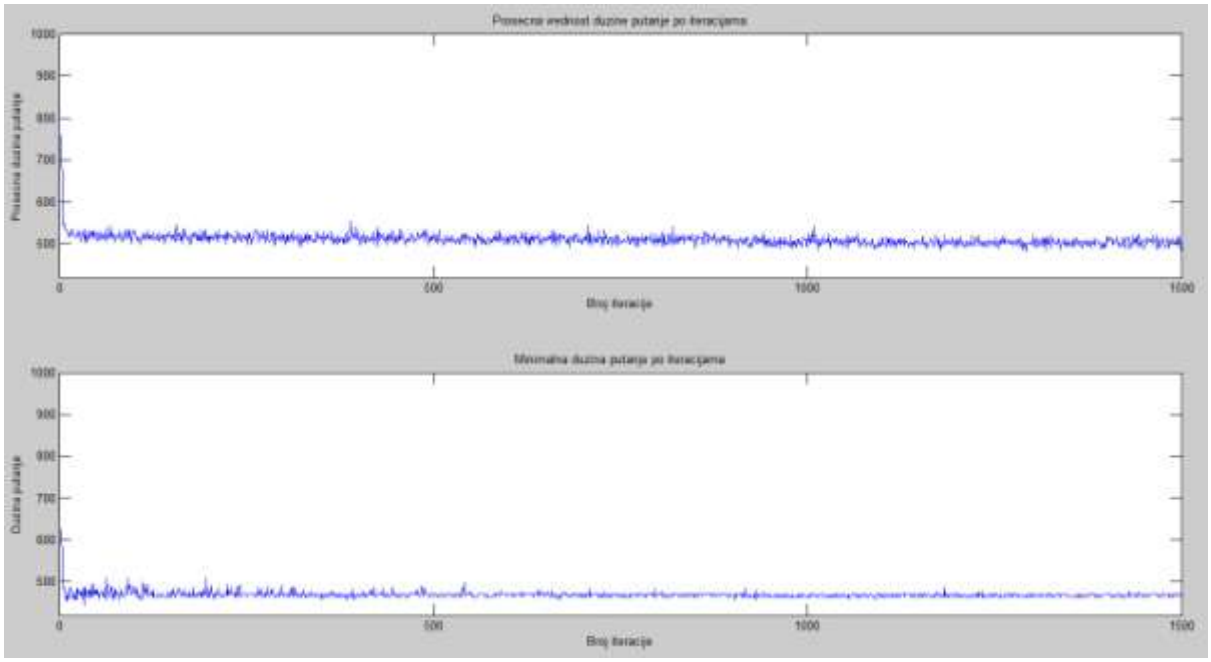


Slika 11: Prvi primer konvergencije rešenja za ACS.

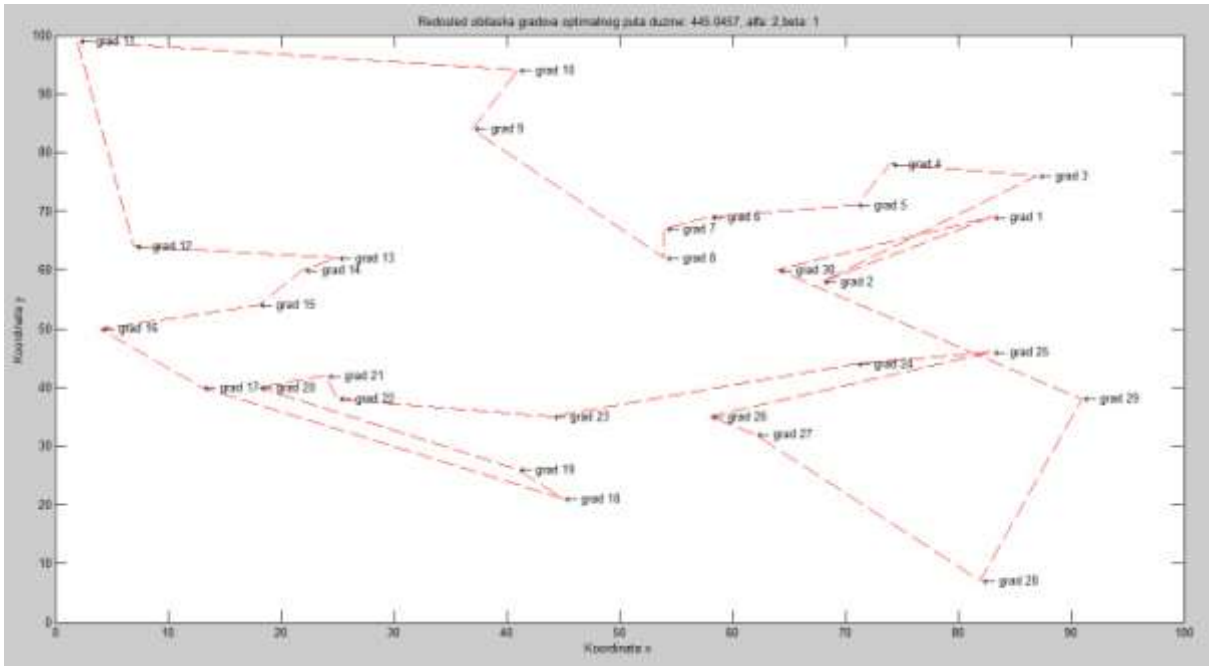


Slika 12: Prvi primer optimalne putanje za ACS.

Iako se na prvi pogled čini da rešenja i dalje osciluju u velikoj meri to ipak nije tako. Skala na kojoj postoje odstupanja od minimalnog puta je mnogo manja nego u prethodnim slučajevima. Situacija je još jasnija u sledećem primeru gde su parametri podešeni drugačije (alfa=2, beta=1).



Slika 13: Drugi primer konvergencije rešenja za ACS.



Slika 14: Drugi primer optimalne putanje za ACS.

Na poslednja dva primera možemo videti uticaj odnosa eksploracije i eksploatacije na rešenje. Naime, u prvom primeru eksploracija ima veći uticaj pa zbog toga vrednosti minimalne putanje u odnosu na broj iteracije dosta variraju u zavisnosti od broja iteracije. Ipak, rešenje dobijeno na ovaj način je bolje nego rešenje dobijeno u narednom primeru gde je sistem podešen tako da veći značaj daje eksploataciji postojećih znanja. Tada su oscilacije dužina minimalnih putanja manje ali je rešenje lošije.

Ovaj odnos između eksploracije i eksploatacije je suptilan ali rešenja mogu biti bitno različita što daje potporu mojoj tvrdnji da je ACS manje robustan nego AS.

Bitno je napomenuti i da se pokazuje da ACS algoritam radi veoma efikasno i bez lokalnog ažuriranja feromona i bez ažruriranja feromona lokalno najboljeg optimalnog puta. Upravo su rezultati koje sam ja dobio dokaz ove tvrdnje (u poslednja dva primera nisam koristio malopre pomenute linije koda).

Kompletan kod za AS i ACS sam izložio u dodatku A i dodatku B, respektivno.

Dodatak A
Ant System algoritam

```
%% Domaci zadatak 2 - 06.09.2010. Autor: Ognjen Zelenbabic
% Problem: Resavanje TSP-a pomocu Ant System algoritma
clc
clear all
close all

%% Definisem pocetne podatke

% Koordinate gradova
x=[...
    54 54 37 41  2  7 25 22 18  4 13 18 24 25 44 ...
    41 45 58 62 82 91 83 71 64 68 83 87 74 71 58];
y=[...
    67 62 84 94 99 64 62 60 54 50 40 40 42 38 35 ...
    26 21 35 32  7 38 46 44 60 58 69 76 78 71 69];

br_grd=length(x); % Broj gradova (ili cvorova)
br_mrava=15; % Broj mrava u koloniji
alfa=1; % Znacaj uticaja feromona
beta=2; % Znacaj uticaja udaljenosti
tau=(ones(br_grd,br_grd)-eye(br_grd)); % Pocetna kolicina feromona na svim granama
ro=0.5;
br_iter=1500; % Broj iteracija
delta_tau=zeros(br_grd,br_grd);
min_duz_putanje=zeros(br_iter,1); % Minimalna duzina putanje date iteracije
lok_min=zeros(br_iter,br_grd+1); % Lokalni minimum za datu iteraciju
pros_duz_putanje=zeros(br_iter,1); % Prosecna duzina putanje date iteracije

% Racunam rastojanja izmedju svih gradova i zelju za granom (ni)
d=zeros(br_grd,br_grd);
ni=zeros(br_grd,br_grd);
for i=1:br_grd
    for j=1:br_grd
        d(i,j)=sqrt((x(j)-x(i))^2+(y(j)-y(i))^2);
        if (d(i,j)==0)
            ni(i,j)=0;
        else
            ni(i,j)=1/d(i,j);
        end
    end
end

%% Ciklus mrava
for a=1:br_iter
    putanja=zeros(br_mrava,br_grd+1); % Pamtim putanju mrava (ruta)
    duz_putanje=zeros(br_mrava,1); % Pamtim duzinu putanje mrava

    for m=1:br_mrava
        % Svaki mrav nasumicno zauzima pocetnu poziciju
        ppm=fix(1+br_grd*rand(1)); % Pocetna Pozicija Mrava
        putanja(m,1)=ppm; % Pocetna pozicija mrava u putanji
        putanja(m,br_grd+1)=putanja(m,1); % Poslednja pozicija mrava u putanji

        % Pravim matricu preostalih gradova koje mrav mora da poseti (sa nulama)
        ostali_grd_0=zeros(1,br_grd);
        k=0; % Pomocni brojac
        for i=1:br_grd
            if (i~=ppm)
                k=k+1;
                ostali_grd_0(1,k)=i;
            end
        end

        % Pravim matricu preostalih gradova koje mrav mora da poseti (bez nula)
        ostali_grd=zeros(1,k);
        for i=1:k
            ostali_grd(1,i)=ostali_grd_0(1,i);
        end

        for n=2:br_grd
            % Verovatnoca da mrav poseti sledeci grad
```

```
            suma=0;
            for i=1:k % Broj preostalih gradova da se posete
                suma=suma+(tau(ppm,ostali_grd(i)).^alfa).*(ni(ppm,ostali_grd(i)).^beta);
            end

            % Verovatnoca da mrav poseti jedan od preostalih gradova
            p=zeros(1,br_grd);
            for i=1:k
                privr=((tau(ppm,ostali_grd(i)).^alfa).*(ni(ppm,ostali_grd(i)).^beta));
                p(1,ostali_grd(i))=privr/suma;
            end

            % Odabir sledeceg grada u koji mrav ide
            max_p=max(p); % Trazim maksimalnu verovatnocu
            for i=1:br_grd
                if (p(1,i)==max_p)
                    % Odabran je sledeci grad (grad sa najvecom verovatnocom)
                    sled_grd=i;
                    % Azuriram informacije o putanji
                    putanja(m,n)=sled_grd;
                    % Azuriram duzinu putanje
                    duz_putanje(m,1)=duz_putanje(m,1)+d(ppm,sled_grd);
                    % Azuriram pocetnu poziciju mrava
                    ppm=sled_grd;
                end
            end

            % Azuriram listu preostalih gradova koje mrav ima da poseti
            ostali_grd_1=zeros(1,k-1);
            broj=k;
            k=0; % Pomocni brojac
            for i=1:broj
                if (ostali_grd(1,i)~=sled_grd)
                    k=k+1;
                    ostali_grd_1(1,k)=ostali_grd(1,i);
                end
            end
            ostali_grd=zeros(1,k);
            for i=1:k
                ostali_grd(1,i)=ostali_grd_1(1,i);
            end
        end
    end
    for m=1:br_mrava
        % Primenjujem pravilo globalnog azuriranja feromona
        for i=1:br_grd
            for j=1:br_grd
                for f=1:br_grd
                    if (i==putanja(m,f) && j==putanja(m,f+1))
                        delta_tau(i,j)=delta_tau(i,j)+1/duz_putanje(m,1);
                        delta_tau(j,i)=delta_tau(j,i)+1/duz_putanje(m,1);
                    end
                end
            end
        end
    end

    % Azuriram duzinu putanje koju je svaki mrav presao
    for m=1:br_mrava
        duz_putanje(m,1)=duz_putanje(m,1)+d(putanja(m,1),putanja(m,br_grd));
    end

    % Azuriram kolicinu feromona na svim granama
    for i=1:br_grd
        for j=1:br_grd
            tau(i,j)=(1-ro)*tau(i,j)+delta_tau(i,j);
            tau(j,i)=(1-ro)*tau(j,i)+delta_tau(j,i);
        end
    end

    % Trazim lokalni minimum
    min_duz_putanje(a,1)=min(duz_putanje);
    for m=1:br_mrava
        if (duz_putanje(m,1)==min_duz_putanje(a,1))
            lok_min(a,:)=putanja(m,:);
```

```
        end
    end

    % Prosecna vrednost duzine putanje
    suma=0;
    for i=1:br_mrava
        suma=suma+duz_putanje(i,1);
    end
    pros_duz_putanje(a,1)=suma/br_mrava;
end

%% Iscrtavanje resenja

% Trazim globalni minimum
globalni_minimum=zeros(1,br_grd+1);
duzina_optimalnog_puta=min(min_duz_putanje);
for i=1:br_iter
    if (duzina_optimalnog_puta==min_duz_putanje(i,1))
        globalni_minimum(1,:)=lok_min(i,:);
    end
end

% Iscrtavam minimalnu i prosecnu duzinu putanje po iteracijama
figure(1)
subplot(211)
plot(1:br_iter,pros_duz_putanje);
title('Prosecna vrednost duzine putanje po iteracijama');
xlabel('Broj iteracije');
ylabel('Prosecna duzina putanje');
axis([0 br_iter 420 1000]);

subplot(212)
plot(1:br_iter,min_duz_putanje)
title('Minimalna duzina putanje po iteracijama');
xlabel('Broj iteracije');
ylabel('Duzina putanje');
axis([0 br_iter 420 1000]);

% Iscrtavam raspored gradova
figure(2)
%subplot(211)
plot(x,y,'.b');
title('Raspored gradova (brojevima je oznacen redosled koordinata)');
xlabel('Koordinata x');
ylabel('Koordinata y');
for i=1:br_grd
    text(x(i)+0.2,y(i)+0.2,[' ',num2str(i)]);
end

% Crtam optimalni put
figure(3)
%subplot(212)
xd=zeros(1,br_grd+1);
yd=zeros(1,br_grd+1);
for i=1:br_grd
    for j=1:br_grd
        if (globalni_minimum(1,j)==i)
            xd(1,i)=x(1,j);
            yd(1,i)=y(1,j);
        end
    end
end
xd(1,br_grd+1)=xd(1,1);
yd(1,br_grd+1)=yd(1,1);

plot(xd,yd,'--r');

for i=1:br_grd
    text(xd(i)+0.2,yd(i)+0.2,['\leftarrow grad ',num2str(i)]);
end

title(['Redosled obilaska gradova optimalnog puta duzine: ',num2str(duzina_optimalnog_puta),',
alfa: ',num2str(alfa),',beta: ',num2str(beta)]);
xlabel('Koordinata x');
ylabel('Koordinata y');
```

Dodatak B
Ant Colony System algoritam

```
%% Domaci zadatak 2 - 06.09.2010. Autor: Ognjen Zelenbabic
% Problem: Resavanje TSP-a pomocu Ant Colony System algoritma
clc
clear all
close all

%% Definisem pocetne podatke

% Koordinate gradova
x=[...
    54 54 37 41  2  7 25 22 18  4 13 18 24 25 44 ...
    41 45 58 62 82 91 83 71 64 68 83 87 74 71 58];
y=[...
    67 62 84 94 99 64 62 60 54 50 40 40 42 38 35 ...
    26 21 35 32  7 38 46 44 60 58 69 76 78 71 69];

br_grd=length(x); % Broj gradova (ili cvorova)
br_mrava=15; % Broj mrava u koloniji
alfa=2; % Znacaj uticaja feromona
beta=1; % Znacaj uticaja udaljenosti
tau=(ones(br_grd,br_grd)-eye(br_grd)); % Pocetna kolicina feromona na svim granama
ro=0.5;
br_iter=1500; % Broj iteracija
delta_tau=zeros(br_grd,br_grd);
min_duz_putanje=zeros(br_iter,1); % Minimalna duzina putanje date iteracije
lok_min=zeros(br_iter,br_grd+1); % Lokalni minimum za datu iteraciju
pros_duz_putanje=zeros(br_iter,1); % Prosecna duzina putanje date iteracije

% Racunam rastojanja izmedju svih gradova i zelju za granom (ni)
d=zeros(br_grd,br_grd);
ni=zeros(br_grd,br_grd);
for i=1:br_grd
    for j=1:br_grd
        d(i,j)=sqrt((x(j)-x(i))^2+(y(j)-y(i))^2);
        if (d(i,j)==0)
            ni(i,j)=0;
        else
            ni(i,j)=1/d(i,j);
        end
    end
end

%% Ciklus mrava
for a=1:br_iter
    putanja=zeros(br_mrava,br_grd+1); % Pamtim putanju mrava (ruta)
    duz_putanje=zeros(br_mrava,1); % Pamtim duzinu putanje mrava

    for m=1:br_mrava
        % Svaki mrav nasumicno zauzima pocetnu poziciju
        ppm=fix(1+br_grd*rand(1)); % Pocetna Pozicija Mrava
        putanja(m,1)=ppm; % Pocetna pozicija mrava u putanji
        putanja(m,br_grd+1)=putanja(m,1); % Poslednja pozicija mrava u putanji

        % Pravim matricu preostalih gradova koje mrav mora da poseti (sa nulama)
        ostali_grd_0=zeros(1,br_grd);
        k=0; % Pomocni brojac
        for i=1:br_grd
            if (i~=ppm)
                k=k+1;
                ostali_grd_0(1,k)=i;
            end
        end

        % Pravim matricu preostalih gradova koje mrav mora da poseti (bez nula)
        ostali_grd=zeros(1,k);
        for i=1:k
            ostali_grd(1,i)=ostali_grd_0(1,i);
        end

        for n=2:br_grd
            % Verovatnoca da mrav poseti sledeci grad
```

```
        suma=0;
        for i=1:k % Broj preostalih gradova da se posete
            suma=suma+(tau(ppm,ostali_grd(i)).^alfa).*(ni(ppm,ostali_grd(i)).^beta);
        end

        % Verovatnoca da mrav poseti jedan od preostalih gradova
        p=zeros(1,br_grd);
        for i=1:k
            privr=((tau(ppm,ostali_grd(i)).^alfa).*(ni(ppm,ostali_grd(i)).^beta));
            p(1,ostali_grd(i))=privr/suma;
        end

        % Odabir sledeceg grada u koji mrav ide
        max_p=max(p); % Trazim maksimalnu verovatnocu
        for i=1:br_grd
            if (p(1,i)==max_p)
                % Pozivam funkciju za biranje sledeceg grada
                [sled_grd]=biranje_acs(p,k,i);
                % Azuriram informacije o putanji
                putanja(m,n)=sled_grd;
                % Azuriram duzinu putanje
                duz_putanje(m,1)=duz_putanje(m,1)+d(ppm,sled_grd);
                % Uvodim lokalno azuriranje feromona
                tau(ppm,sled_grd)=(1-ro)*tau(ppm,sled_grd)+ro*(1/duz_putanje(m,1));
                % Azuriram pocetnu poziciju mrava
                ppm=sled_grd;
            end
        end

        % Azuriram listu preostalih gradova koje mrav ima da poseti
        ostali_grd_1=zeros(1,k-1);
        broj=k;
        k=0; % Pomocni brojac
        for i=1:broj
            if (ostali_grd(1,i)~=sled_grd)
                k=k+1;
                ostali_grd_1(1,k)=ostali_grd(1,i);
            end
        end
        ostali_grd=zeros(1,k);
        for i=1:k
            ostali_grd(1,i)=ostali_grd_1(1,i);
        end
    end
end
for m=1:br_mrava
    % Primenjujem pravilo globalnog azuriranja feromona
    for i=1:br_grd
        for j=1:br_grd
            for f=1:br_grd
                if (i==putanja(m,f) && j==putanja(m,f+1))
                    delta_tau(i,j)=delta_tau(i,j)+1/duz_putanje(m,1);
                    delta_tau(j,i)=delta_tau(j,i)+1/duz_putanje(m,1);
                end
            end
        end
    end
end


% Azuriram duzinu putanje koju je svaki mrav presao
for m=1:br_mrava
    duz_putanje(m,1)=duz_putanje(m,1)+d(putanja(m,1),putanja(m,br_grd));
end

% Azuriram kolicinu feromona na svim granama
for i=1:br_grd
    for j=1:br_grd
        tau(i,j)=(1-ro)*tau(i,j)+delta_tau(i,j);
        tau(j,i)=(1-ro)*tau(j,i)+delta_tau(j,i);
    end
end

% Trazim lokalni minimum
min_duz_putanje(a,1)=min(duz_putanje);
```

```
    for m=1:br_mrava
        if (duz_putanje(m,1)==min_duz_putanje(a,1))
            lok_min(a,:)=putanja(m,:);
        end
    end

    % Dodajem dodatne feromone na putanju lokalnog minimuma
    for i=1:br_grd
        pom=tau(lok_min(a,i),lok_min(a,i+1));
        tau(lok_min(a,i),lok_min(a,i+1))=(1-ro)*pom;
    end

    % Prosecna vrednost duzine putanje
    suma=0;
    for i=1:br_mrava
        suma=suma+duz_putanje(i,1);
    end
    pros_duz_putanje(a,1)=suma/br_mrava;
end

%% Iscrtavanje resenja

% Trazim globalni minimum
globalni_minimum=zeros(1,br_grd+1);
duzina_optimalnog_puta=min(min_duz_putanje);
for i=1:br_iter
    if (duzina_optimalnog_puta==min_duz_putanje(i,1))
        globalni_minimum(1,:)=lok_min(i,:);
    end
end

% Iscrtavam minimalnu i prosecnu duzinu putanje po iteracijama
figure(1)
subplot(211)
plot(1:br_iter,pros_duz_putanje);
title('Prosecna vrednost duzine putanje po iteracijama');
xlabel('Broj iteracije');
ylabel('Prosecna duzina putanje');
axis([0 br_iter 420 1000]);

subplot(212)
plot(1:br_iter,min_duz_putanje)
title('Minimalna duzina putanje po iteracijama');
xlabel('Broj iteracije');
ylabel('Duzina putanje');
axis([0 br_iter 420 1000]);

% Iscrtavam raspored gradova
figure(2)
%subplot(211)
plot(x,y,'.b');
title('Raspored gradova (brojevima je oznacen redosled koordinata)');
xlabel('Koordinata x');
ylabel('Koordinata y');
for i=1:br_grd
    text(x(i)+0.2,y(i)+0.2,[' ',num2str(i)]);
end

% Crtam optimalni put
figure(3)
%subplot(212)
xd=zeros(1,br_grd+1);
yd=zeros(1,br_grd+1);
for i=1:br_grd
    for j=1:br_grd
        if (globalni_minimum(1,j)==i)
            xd(1,i)=x(1,j);
            yd(1,i)=y(1,j);
        end
    end
end
xd(1,br_grd+1)=xd(1,1);
yd(1,br_grd+1)=yd(1,1);

plot(xd,yd,'--r');
```

```
for i=1:br_grd
    text(xd(i)+0.2,yd(i)+0.2,['\leftarrow grad ',num2str(i)]);
end

title(['Redosled obilaska gradova optimalnog puta duzine: ',num2str(duzina_optimalnog_puta),',
alfa: ',num2str(alfa),',beta: ',num2str(beta)]);
xlabel('Koordinata x');
ylabel('Koordinata y');
```

Funkcija *biranje_acs*.

```
function [sled_grd]=biranje_acs(p,k,i)
% Odabran je sledeci grad (grad sa najvecom verovatnocom)
sled_grd=i;
q=rand(1);
suma=0;
for i=1:k
    suma=suma+p(1,i);
    if q<=suma
        sled_grd=i;
        break
    end
end
```