Univerzitet u Beogradu
Matematički fakultet

Aleksandar Kartelj

Rešavanje problema minimalne energetske povezanosti u težinskom grafu primenom genetskog algoritma

-master rad-

Beograd
2010.

Mentor: **doc. dr Vladimir Filipović**
Matematički fakultet u Beogradu

Članovi komisije: **prof. dr Dušan Tošić**
Matematički fakultet u Beogradu

doc. dr Nenad Mitić
Matematički fakultet u Beogradu

Datum odbrane: _____

Rešavanje problema minimalne energetske povezanosti u težinskom grafu primenom genetskog algoritma

Rezime

U ovom radu je prikazan genetski algoritam (GA) za rešavanje grafovskog problema minimalne energetske povezanosti (SMET). Prostor za primenu ovakvog algoritma je vrlo širok, prevashodno u mrežama bežičnih senzora. Dokazano je da ovaj problem pripada grupi NP kompletnih problema. Uzimajući u obzir polja primene rešavača ovakvog problema, neegzaktni načini rešavanja, sa suboptimalnim rešenjima, predstavljaju značajan doprinos u realnim aplikacijama. U literaturi se spominju i približni algoritmi zasnovani na rešavanju problema minimalnog povezujućeg stabla sa garantovanim koeficijentom kvaliteta, no postoje i egzaktni metodi koji su bazirani na različitim formulacijama linearnog programiranja i pretrazi sa odsecanjem. Predstavljeni su rezultati koji predstavljaju optimalna rešenja u slučajevima uporedivih manjih dimenzija, ali su predstavljena i rešenja za slučajeve dosta većih dimenzija.

Abstract

In this work genetic algorihm for solving graph problem of minimal strong energy topology (SMET) is presented. The area of using this algorithm is very wide, before all in the networks of wireless sensors. It has been proven that this problem belongs to the group of NP complete problems. Taking in consideration the field of application of the solver for this kind of problem, non exact ways of solving it with suboptimal solutions give significant benefit in real applications. In literature non exact algorithms with guaranteed coefficient of quality usually based on minimal cost spanning tree (MST) are mentioned, but also exact methods, based of the different formulations of linear programming and branch and cut search, exist. In this work the results giving optimal solutions in comparable cases of lower dimensions and also solutions for higher dimensions (without proving exactness) are presented.

SADRŽAJ

1. UVOD

1.1 Grafovi

Za teoriju grafova se može reći da beleži svoje početke još od 1736 kada je Ojler (Euler) razmatrao problem Keninsberških mostova (eng. Königsberg bridge problem): Da li postoji put koji prolazi kroz svaki od sedam mostova tačno jednom? Ojler je pokazao da problem ima negativno rešenje, zbog toga što su čvorovi u pridruženom grafu neparnog stepena [18].

U matematici graf predstavlja abstraktnu reprezentaciju skupa objekata među kojima su neki od njih povezani. Te objekte nazivamo čvorovi, a veze među njima ivice ili grane grafa. Obično, grafički prikaz grafa se sastoji od tačkica za čvorove i linija za ivice. Ivice mogu biti usmerene (asimetrični graf) ili neusmerene (simetrični graf). Na primer. ako se na nekoj zabavi osoba A rukuje sa osobom B, to podrazumeva i da se osoba B rukovala sa osobom A, dakle u tom slučaju se radi o neusmerenoj vezi. Sa druge strane ako linija nosi značenje poznavanja druge osobe, onda to što osoba A poznaje osobu B nužno ne uzrokuje i obrnuto značenje, pa se ovde radi o usmerenom grafu.

Može se naslutiti već iz pomenutog primera da je područje primene grafova vrlo široko, i da se mnogi realni problemi i situacije mogu predstaviti gravofski na primer: putevi, avionske, željezničke, računarske i druge mreže, razni odnosi između ljudi kao što su porodična stabla, sklopljeni brakovi i slično.

Neka je V konačan skup čvorova i obeležimo sa:

$$E(V) = \{\{u,v\} \mid u,v \in V, u \neq v\} \tag{1.1}$$

skup podskupova veličine 2 od V sa različitim elementima.

Definicija 1.1 Par *G=(V,E) gde je* $E \subseteq E(V)$ se naziva *graf* (neusmereni) nad V. Elementi iz V se nazivaju čvorovi grafa, a elementi iz E ivice grafa.

Alternativno graf možemo definisati i na sledeći način:

Definicija 1.2 *Graf* $G = (V, E)$ je uređeni par koji se sastoji od konačnog skupa čvorova V i binarne relacije E nad V.

Definicija 1.3 *Digraf* $G = (V, E)$ ili usmereni graf je uređeni par gde je V konačan skup čvorova i $E \subseteq E(V)$, sa tom razlikom da je *E(V)* definisano kao:

$$E(V) = \{(u,v) \mid u,v \in V, u \neq v\} \tag{1.2}$$

Definicija 1.4 Graf K_r je *r-kompletan* ako važi *E=E(V)*, tj. za svaki par čvorova postoji grana i $|V|=r$.

Definicija 1.5 *Stepen nekog čvora v digrafa G=(V,E)* se definiše dvojako

- izlazni stepen: $d^+(v) = |\{u \mid (v,u) \in E\}|$
- ulazni stepen: $d^-(v) = |\{u \mid (u,v) \in E\}|$ i važi sledeće tvrđenje:

$$\sum_{v \in V} d^+(v) = \sum_{v \in V} d^-(v) = |E| \tag{1.3}$$

Definicija 1.6 *Stepen nekog čvora v grafa G=(V,E)* se definiše kao:

$d(v) = |\{\{u,v\} \mid \{u,v\} \in E, u \neq v\}|$, i važi sledeće tvrđenje:

$$\frac{1}{2}\sum_{v \in V} d(v) = |E| \tag{1.4}$$

.

Definicija 1.7 *Put dužine l* između čvorova v_o *i* v_1 je niz čvorova $A = [v_0, v_1, v_2, ..., v_l]$ u grafu G ako važi:

$$(v_{i-1}, v_i) \in E \quad za\ i = 1, 2, ..., l$$

Defincija 1.8 *Povezani graf* je graf u kojem između bilo koja dva čvora postoji put koji ih spaja. Sa druge, strane za nepovezani graf postoji bar jedan par čvorova između kojih ne postoji put, tj. postoji više odvojenih povezanih delova i njih nazivamo *komponente povezanosti.*

Definicija 1.9 *Strogo povezani graf* se odnosi specijalno na slučaj digrafa i predstavlja digraf u kojem između svaka dva čvora postoji put u oba smera.

Definicija 1.10 *Težinski graf (mreža)* je uređeni par (G, ω), gde je *G=(V,E),* a ω je funkcija koja svakoj grani $e = (u,v),\ e \in E$ dodeljuje realnu vrednost $\omega(e)$.

Definicija 1.11 *Minimalno razapinjuće stablo (eng MST – minimum spanning tree)* za dati težinski graf (G, ω) je stablo *T* koje sadrži sve čvorove datog grafa i ima minimalnu vrednost za:

$$\sum_{e \in T} \omega(e)$$

Definicija 1.12 *Artikulacioni čvor* je čvor čijim izbacivanjem graf postaje nepovezan (eng. cut point). Skup čvorova $V' \subset V$, $|V'| = k$, takvih da je $G \setminus V'$ nepovezan je *k-čvorni rez* grafa *G*.

Definicija 1.13 *Most* je grana čijim izbacivanjem graf postaje nepovezan. Skup grana $E' \subset E$, takav da je graf $G \setminus E'$ nepovezan ili trivijalan je *granski rez.*

1.2 Bežične senzorske mreže

Bežična senzorska mreža - BSM (eng. wireless sensor network –WSN) [38], [25] se sastoji od geografski distribuiranih autonomnih senzora koji koopeartivno prate neke fizičke ili prostorne uslove kao što su temperatura, zvuk, vibracije, pritisak i druge. Razvoj bežičnih senzorskih mreža je bio motivisan razvojem vojnih mehanizama kao što je npr. nadgledanje bojnog polja. Danas se koriste kako za civilne potrebe, tako i za industrijske procese nadgledanja i kontrolisanja mašina, prostorne potrebe nadgledanja parametara zagađenja prirode, kontrolu saobraćaja i mnoge druge.

Osnovna jedinica senzorske mreže, senzor obično je opremljen odgovarajućim mernim instrumentom i radio prijemnikom za komunikaciju sa ostatkom mreže. Pored navedenog, senzor obično poseduje i mali programabilni mikrokontroler i naravno energetski izvor, obično bateriju. Problematika koja je aktivna u kontekstu bežičnih senzorskih mreža obuhvata optimizaciju ukupne utrošene energije, memorije, računskih resursa i protoka podataka. Naravno, sve ovo je potrebno izvesti uz odgovarajuća ograničenja prostornog rasporeda, dostupnih energetskih, memorijskih i računskih resursa, verovatnoću kvara senzora, stepen mobilnosti, dinamičnosti mreže, heterogenosti senzorskih uređaja, ponašanja u mrežama velikih dimenzija itd.

Hardverske mogućnosti ovih uređaja su u skladu sa njihovom cenom. S'obzirom na broj uređaja koji se obično koristi u izgradnji bežične senzorske mreže, prirodno se nameće potreba ekonomske dostupnosti uređaja. Tako da su osnovni zahtevi da uređaji budu što jeftiniji, ali i što manjih dimenzija.

Softverske mogućnosti obično adresiraju optimizacione probleme uštede resursa. Energija je svakako najznačajniji resurs svakog čvora BSM. Problemu se može pristupati kako pojedinačno, gde su uštede obično konstante i zasnivaju se na minimiziranju troškova pristupa i aktiviranja uređaja, tako i globalno – odabirom prave topologije, dobrim protokolima komunikacije itd. Pored maksimizacije trajanja baterije, interesantna polja optimizacije su i pitanja robusnosti, tolerancije prema padu rada uređaja, kao i mogućnosti dinamičkog konfigurisanja.

Algoritmi na kojima se zasniva rešavanje problema navedenih pragmatičkih potreba su obično abstrahovani, tj. predstavljeni metajezikom mreže, a to su naravno grafovi. Pored toga algoritmi moraju omogućiti svojstva distribuiranog i paralelnog rada u nekim slučajevima. Postoje verifikacioni softverski alati tzv. simulatori mreže koji omogućavaju proveru i vizuelizaciju problema i rešenja na intuitivan način. Neki od poznatijih simulatora su TOSSIM, COOJA, ns-2, J-Sim i drugi.

1.3 Optimizacija

U matematici i računarstvu *optimizacija (matematičko programiranje)* predstavlja proces odabira najboljeg elementa iz skupa mogućih elemenata, gde su ti elementi obično moguća rešenja nekog problema. U najjednostavnijem slučaju ovo znači rešavanje problema traženja minimuma ili maksimuma neke realne funkcije sistematskom pretragom vrednosti realnih ili celobrojnih argumenata te funkcije.

Formalnije, optimizacioni problem A je uređena četvorka (*I,f,m,g),* gde je:

- *I* skup svih instanci odnosno ulaznih argumenata problema
- za datu instancu $x \in I$, *f(x)* je skup svih mogućih rešenja
- za datu instancu *x* i neko njeno moguće rešenje *y, m(x,y)* označava meru za *y,* i ona je obično pozitivan realan broj
- *g* je funkcija cilja čiji se ekstremum traži (minimum ili maksimum)

Cilj je dakle za zadatu instancu *x* pronaći *optimalno rešenje*, tj. moguće rešenje *y* sa merom

$$m(x, y) = g\{m(x, y') \mid y' \in f(x)\}.$$

Pošto je često teško naći optimum na egzaktan način, algoritmi se obično konstruišu tako da pronalaze rešenja bliska optimalnim.

1.3.1 Istorijat i oblasti

Prva optimizaciona tehnika, poznata pod nazivom metod najstrmijeg spusta (eng. steepest descent) vezuje se još za Gausa (Gauss). Poseban značaj optimizacione tehnike su počele da dobijaju pojavom linearnog programiranja, koje je izumeo Džordž Danzih (George Dantzig) četrdesetih godina prošlog veka. Termin programiranje u ovom kontekstu se ne vezuje za programiranje na računarima, već zato što se tehnika koristila u okviru vojnog *programa* Amerike za vreme drugog svetskog rata koji je imao za cilj da matematičkim modelom planiranja smanji troškove ratovanja i poveća gubitke protivnika.

Matematičko programiranje ima veliki niz podoblasti:

- *Konveksno programiranje* proučava optimizacione probleme kod kojih je funkcija cilja konveksna, a ograničenja, ukoliko postoje, formiraju konveksan skup. Najjednostavnija forma ovog tipa problema je problem *Linearnog programiranja (LP)*, kod kojeg je funkcija cilja linearna, a skup ograničenja je definisan korišćenjem samo linearnih jednačina i nejednačina.
- *Celobrojno programiranje* proučava linearne programe kod kojih neke ili sve promenljive imaju ograničenje rada nad skupom celih brojeva. Ovim se gubi svojstvo konveksnosti i generalno je ovakve probleme mnogo teže rešiti.
- *Kvadraturno programiranje* dozvoljava da u funkciji cilja figurišu kvadrirani izrazi, dok skup ograničenja ostaje linearan. Za određene kvadraturne forme problem ostaje konveksan.
- *Nelinearno programiranje* posmatra opšti slučaj u kojem bilo funkcija cilja bilo ograničenja poseduju nelinearne segmente. Ovo može, a ne mora biti konveksan program, no konveksnost je svakako poželjno svojstvo.
- *Stohastičko programiranje* koristi i slučajne brojeve u ograničenjima ili parametrima problema.
- *Kombinatorna optimizacija* se ograničava na slučajeve u kojima je skup mogućih rešenja diskretan ili se može svesti na diskretan.

- *Beskonačno dimenziona optimizacija* proučava slučajeve kada je skup svih mogućih rešenja podskup nekog beskonačno dimenzionog prostora.
- i druge.

1.3.2 Kombinatorna optimizacija

Kao što je gore navedeno, kombinatorna optimizacija predstavlja podoblast optimizacije kod koje su problemi definisani nad diskretnim skupom mogućih rešenja ili se na njega mogu svesti. Ova oblast je usko povezana sa *operacionim istraživanjima* i *teorijom složenosti algoritama.*

Neki od primera problema kombinatorne optimizacije su:

- Problem trgovačkog putnika (eng. TSP)
- Minimalno razapinjuće stablo (eng. MST)
- Problem ranca
- i drugi.

Formalno, problem ovog tipa se može predstaviti na sledeći način:

Problem: Neka je dat konačan diskretan skup S *(*dopustivi skup) i funkcija f (funkcija cilja) koja elemente iz skupa S slika u skup realnih brojeva R *(* $f : S \to R$*).* Naći minimum (maksimum) funkcije f na skupu S, tj. rešiti:

$$\min_{x \in S} f(x)$$

Rešenje x' takvo da

$$\min(f(x')) = m = \min_{x \in S} f(x),$$

nazivamo optimalnim rešenjem problema kombinatorne optimizacije.

Diskretnost dopustivog skupa implicira postojanje egzaktnog eksponencijalnog algoritma.

Problem: Problem trgovačkog putnika je grafovski problem obilaska svih čvorova (gradova) grafa na najjeftiniji mogući način.

Ovo je očigledno problem kombinatorne optimizacije jer je skup dopustivih rešenja diskretan i odgovara broju permutacija redosleda čvorova (gradova), a funkcija cilja je suma svih grana između uzastopnih čvorova u grafu. Međutim, jasno je da ovaj algoritam nije efikasan, i šta više pripada klasi NP kompletnih problema o kojima će biti reči u narednom paragrafu.

1.4 NP kompletni problemi

U teoriji složenosti algoritama NP kompletan je oznaka za klasu problema koja imaju sledeća dva svojstva:

- Problem je nedetermnistički polinomski (NP), što podrazumeva da se za predloženo rešenje u polinomskom vremenu može utvrditi (verifikovati) da li ono pripada ili ne pripada skupu rešenja datog problema. Za deterministički polinomski problem (P) sa druge strane, u polinomskom vremenu moguće je i pronaći rešenje, a ne samo izvršiti njegovu verifikaciju.
- Svaki problem koji pripada klasi NP se može svesti na ovaj problem u polinomskom vremenu.

Iako se verifikacija rešenja može izvršiti relativno brzo, ne zna se da li postoji efikasan način pronalaženja rešenja, jer vremenska složenost u do sada predloženim algoritmima za ovu vrstu problema, raste eksponencijalno sa povećanjem dimenzije ulaza. Pitanje da li postoji efikasan algoritam za neki NP kompletan problem je jedno od osnovnih nerešenih pitanja današnjice u oblasti matematike i računarstva. Ukoliko bi odgovor na ovo pitanje bio pozitivan to bi impliciralo jednakost klasa determinističkih i nedeterminističkih polinomskih problema tj. P=NP [13].

1.4.1 Tehnike rešavanja NP kompletnih problema

Rešavanje NP kompletnih problema se obično zasniva na približnim tehnikama, osim u slučajevima manjih dimenzija kada se mogu iskoristiti egzaktni algoritmi. Osnovni pristupi rešavanja su:

- *Aproksimacija*: umesto traženja optimalnog rešenja, traži se "skoro" optimalno, što često dovodi do smanjenja prostora pretrage mogućih rešenja .
- *Randomizacija*: zasniva se na korišćenju slučajnih brojeva radi postizanja boljeg prosečnog vremena izvršavanja, ali dopušta da algoritam ne uspe sa unapred zadatom verovatnoćom.
- *Parametrizacija*: koristi činjenicu da često postoji efikasan algoritam u slučaju da su neki od parametara problema fiksirani.
- *Heruistike:* algoritmi koji rade "dobro" u velikom broju slučajeva, ali za koje ne postoji dokaz da su uvek efikasni i da proizvode "dobra" rešenja. Obično se koriste tzv. metaheuristike.

1.5 Metaheuristike

Metaheuristike [23] predstavljaju metode koje optimizuju problem iterativnim poboljšavanjem potencijalnog rešenja uzimajući u obzir zadatu meru kvaliteta. Pripadaju grupi približnih (aproksimativnih) algoritama, pored konstruktivnih metoda koji počinju od praznog skupa i rešenje konstruišu u koracima. Za razliku od heuristika, metaheuristike prave malo pretpostavki ili čak ne prave nikakve pretpostavke o prirodi problema i imaju mogućnost pretrage velikog broja potencijalnih rešenja. Međutim, metaheuristika ne garantuje da će optimalno rešenje biti nađeno. Ove metode se koriste za rešavanje problema kombinatorne optimizacije. Neke od najpoznatijih metaheuristika za rešavanje ovog tipa problema su simulirano kaljenje [27], genetski algoritam [26], mravlje kolonije [17] i drugi. Područje primene metaheuristika su takođe i problemi definisani nad realnim domenima, gde su klasični metodi na primer najstrmiji spust (eng. steepest descent) računski veoma zahtevni.

Metaheuristike mogu biti zasnovane na lokalnoj pretrazi odnosno usmeravanju (eng. trajectory based) i na populaciji (eng. population based).

1.5.1 Metode zasnovane na lokalnoj pretrazi

Osnovna lokalna pretraga *(eng. basic local search)* koristi početno, često slučajno odabrano rešenje i iterativno ga poboljšava dok god je poboljšanje moguće (samo u smeru poboljšanja). Ovo je vrlo bazična forma algoritma koji se zaustavlja čim pronađe lokalni optimum. To nije dobra osobina, ali sa druge strane, predstavlja polaznu osnovu za druge metaheuristike zasnovane na usmeravanju. U literaturi ova metoda se još naziva i metoda penjanja uz brdo (hill climbing method).

Simulirano kaljenje *(eng. simulated annealing)* [27] koristi činjenicu poznatu iz metalurških procesa obrade metala da se postepenim hlađenjem obezbeđuje bolja konfiguracija kristalne rešetke, tj. konfiguracija koja ima manju unutrašnju energiju. Osnova algoritma je lokalna pretraga sa tom razlikom da su dozvoljeni i pomeraji u pravcu bez poboljšanja sa nekom verovatnoćom. Metod poseduje parametar temperature koji su iterativno smanjuje, a izvršavanje metoda se završava ili kada se dosegne stanje iz kojeg se ne može preći u drugo stanje manje energije ili dok parametar temperature ne postane nula.

```
s←InicijalnoRešenje()
T←T0
while uslovi zaustavljanja nisu ispunjeni do
        s'←IzaberiSlučajno(N(s))
        if f(s') < f(s) then
                s←s'
        else
                prihvati s' sa verovatnoćom p(T,s',s)
        endif
        Ažuriraj(T)
endwhile
```

Pri čemu je $p(T,s',s) = e^{-\frac{f(s')-f(s)}{T}}$, a $T_{k+1} = \alpha T_k, 0 < \alpha < 1$.

Jednostavna tabu pretraga *(eng. simple tabu search)* [24] je metaheuristika koja se zasniva na principima lokalnog pretraživanja, a dodatno je da poseduje memoriju koja sadrži informacije o ranijim pretragama (tabu lista), tako da se ne prolazi ponovo kroz neperspektivne „delove“. Postoje razna poboljšanja pre svega vezana za način organizovanja memorije. Osnovni algoritam je prikazan sledećim pseudokodom:

```
s←InicijalnoRešenje()
TabuLista←∅
while uslovi zaustavljanja nisu ispunjeni do
```

```
    s←OdaberiNajboljeOd(N(s)\TabuLista)
    Ažuriraj(TabuLista)
end while
```

1.5.2 Populacione metode

Populacione metode se zasnivaju na evoluciji skupa tačaka u prostoru pretrage. Neke od njih su inspirisane prirodnim procesima, kao što su prirodna evolucija, socijalno ponašanje insekata itd.

Evolucioni algoritmi *(eng. evolutionary algorithms)* [42] simuliraju pojednostavljeni proces evolucije nad tačkama u prostoru koji za cilj ima pronalaženje najboljih tačaka tj. globalnih optimuma. Ovoj grupi algoritama pripadaju evoluciono programiranje, evolucione strategije i genetski algoritmi. U nastavku je dat pseudokod ove metode:

```
P←InicijalnaPopulacija()
Evoluiraj(P)
while uslovi zaustavljanja nisu ispunjeni do
    P'←Rekombinuj(P)
    P''←Mutiraj(P')
    Evoluiraj(P'')
    P←Odaberi(P''∪P)
end while
```

Optimizacija mravljim kolonijama *(eng. ant colony optimization)* [14] je nastala posmatranjem socijalnog ponašanja mrava, tj. konkretno njihovog načina traženja najkraćeg puta do hrane. Dok se kreću mravi ispuštaju supstancu zvanu feromon. Prilikom odlučivanja kojim putem da krenu, mravi sa većom verovatnoćom krenu onim putem gde osećaju jaču dozu feromona, što konačno dovodi do jasno utvrđene optimalne putanje kretanja.

Ostale populacione metaheuristike obuhvataju čitav niz prirodno modeliranih tehnika kao što su (eng. harmony opitimization) harmonijska optimizacija zasnovana na kompozitorskom procesu traženja najbolje melodije, optimizacija rojevima insekata (eng. particle swarm optimization) kod koje je bitan način pomeranja roja u prostoru pretrage i druge.

1. 6 Genetski algoritam

Genetski algoritam je razvio Džon Holand (eng. John Holland) šezdesetih godina prošlog veka . Originalni cilj razvoja genetskog algoritma nije bila praktična primena zarad rešavanja nekog specifičnog problema, već formalna studija o evoluciji i adaptaciji u prirodi i načinima na koje je tu logiku moguće ubaciti u računarstvo. U svojoj knjizi iz 1975. [26], prikazan je genetski algoritam kao abstrakcija biološke evolucije. Da bismo detaljnije opisali osnove genetskog algoritma, uvešćemo deo biološko računarske terminologije vezane za temu biološke evolucije.

Svi živi organizmi se sastoje od ćelija, i svaka ćelija sadrži skup hromozoma, koji su izgrađeni od lanaca DNK (eng. DNA - Deoxyribonucleic acid) koji kodiraju informacije o odgovarajućem organizmu. Hromozom dalje može da se deli u gene, koji su zapravo funkcionalni blokovi DNK, gde svaki gen predstavlja odgovarajuće svojstvo organizma. Različite mogućnosti za svako svojstvo, npr. različita boja za oči se nazivaju genski aleli. Svaki gen je postavljen na određenoj lokaciji hromozoma. Tokom reprodukcije dešava se ukrštanje gena: oni bivaju razmenjeni između parova roditeljskih hromozoma i daju gene potomka. Potomstvo dalje postaje subjekat mutacije, gde su pojedinačni segmenti DNK skloni promenama. "Prilagođenost" ili "dobrota" (eng. fitness) organizma se može kvantifikovati kao verovatnoća da će organizam opstati i dobiti mogućnost da se reprodukuje i ostavi svoje potomstvo u narednoj generaciji [35]. Skup svih hromozoma u datom momentu se naziva populacija. Evolucija se postiže mutacijama i ukrštanjem između različitih jedinki, pri čemu najbolje jedinke imaju pravo da učestvuju u kreiranju naredne generacije. U računarstvu, genetski algoritmi se koriste za pronalaženje „dobrih" rešenja iz velikog skupa mogućih. Od tehničkih aspekata ističemo način kodiranja rešenja tj. hromozoma, inicijalnu populaciju, operatore mutacije i ukrštanja, tzv. funkciju prilagođenosti (funkcija „dobrote" jedinke) i operator selkecije za odabir onih koji će preživeti i ostaviti potomstvo u idućoj generaciji. Osnovni genetski algoritam je dat sledećim pseudokodom:

```
P←InicijalnaPopulacija()
while uslovi zaustavljanja nisu ispunjeni do
        foreach jedinka in P do
                pi←funkcija_cilja()
        end foreach
        IzračunajFunkcijuPrilagodođenosti()
        Selekcija()
        Ukrštanje()
        Mutacija()
end while
```

1.6.1 Kodiranje

Kao što je opisano, osnova prirodne genetike je hromozom. Prilikom projektovanja ove ideje na genetske algoritme, potrebno je razmišljati o mapiranju svake jedinke u okviru populacije u nešto nalik hromozomu. Uobičajeni način da se ovo uradi je korišćenjem niza bitova [35]. Na taj način svaka lokacija u okviru hromozoma predstavlja gen, a aleli odgovaraju vrednostima jedan ili nula. Prednost ovog kodiranja je jednostavnost i ono se često koristi kod problema kombinatorne optimizacije. Na primer, ukoliko želimo da dobijemo sumu brojeva što bližu vrednosti x, sumiranjem brojeva od jedan do dvadeset, tako da zbir brojeva bude što manji, možemo koristiti hromozom od dvadeset bitova, gde je svaki broj predstavljen odgovarajućom lokacijom u okviru hromozoma. Tada alel sa vrednošću jedan, na odgovarajućoj lokaciji, znači da broj ulazi u sumu. Drugi način je da koristimo niz prirodnih brojeva. Takvo kodiranje je dobro za rad sa permutacionim problemima, tj. problema kod kojih je bitan redosled, a ne samo pripadnost skupu rešenja, na primer, problem trgovačkog putnika (eng. travelling salesman person – TSP) [36]. Broj načina kodiranja nije ograničen sa ove dve konvencije - ovo su samo dve najčešće korišćenje i najpogodnije za prikaz. Kodiranje može biti i dijametralno suprotno zamišljeno, ali uvek mora zadržati mogućnosti primene ukrštanja i mutacije.

1.6.2 Mutacije

Mutacija je način kreiranja novih individua iz populacije pravljenjem manjih promena alela na slučajnim lokacijma već postojećih individua (hromozoma). U slučaju binarnog kodiranja, osnovna mutacija podrazumeva promenu 0 u 1 i obrnuto. Na primer, možemo imati niz bitova 010100, i mutirati njegovu treću lokaciju pri čemu ćemo dobiti 011100. Kada niz sadrži prirorodne brojeve, mutacija može podrazumevati zamenu njihovih vrednosti na dve lokacije, dakle jednostavna transpozicija. No, bez obzira na to kako je mutacija zamišljena, dobijeni rezultat uvek mora biti legitimna individua, tj. mora biti "neko" rešenje posmatranog problema. Za svaku mutaciju uvek postoji definisana verovatnoća primene, tzv. koeficijent mutacije (eng. mutation rate) [35]. Kao i u prirodi, mutacije su neželjene u najvećem broju slučajeva, tako da su koeficijenti mutacije uglavnom dosta niski. O tačnom stepenu mutacije treba dobro razmisliti, jer prevelike ili premale verovatnoće primene mutacije mogu rezultirati problemima. Ako je ona prevelika, algoritam će bez jasnog fokusa pretraživati prostor rešenja, u suprotnom ako je premala, populacija će iz generacije u generaciju ostajati vrlo slična, jer ne postoji dovoljno razlika među individuama, što obično dovodi do lokalnog optimuma.

1.6.3 Ukrštanje

Operator ukrštanja se primenjuje na dva hromozoma roditelja i kreira dva potpuno nova hromozoma, tj. njihovo potomostvo, koje sadrži kombinovana svojstva roditelja. Kao i mutacije, ukrštanje se primenjuje na slučajno odabranim lokacijama hromozoma, tzv. tačkama prekida. Operator ukrštanja vrši transpoziciju podnizova pre i posle odabrane lokacije [35, 36].
Na primer, predpostavimo da imamo hromozome $x_1x_2x_3...x_n$ i $y_1y_2y_3...y_n$, i odabrali smo lokaciju k, k<n za tačku prekida. Potomstvo će u tom slučaju predstavljati dva hromozoma $x_1x_2..x_ky_{k+1}..y_n$ i $y_1y_2..y_kx_{k+1}...x_n$. Vodeći se ovim primerom, lako je zamisliti ukrštanje sa dve, tri, u opštem slučaju m tačaka prekida.
Kao i mutacija, operator ukštanja ima koeficijent ukrštanja, koji predstavlja verovatnoću da će se primeniti na nekom hromozomu. Međutim, postoje bitne razlike između ova dva koeficijenta. Najpre, koeficijent ukrštanja je u pozitivnoj korelaciji sa funkcijom prilagođenosti. Drugo, koeficijent ukrštanja ne odgovara uvek verovatnoći ukrštanja. Na primer. u slučaju ukrštanja sa više tačaka prekida, imamo dva parametra: verovatnoću da će se operator uopšte primeniti na datoj jedinki, i ako se primenjuje, koje će biti tačke prekida [35].
U [35] i [39] se smatra da operator selekcije (odabira) uvek bira roditelje, te na taj način svi hromozomi postaju subjekt ukrštanja. Operator ukrštanja zatim utvrđuje da li će se ukrštanje i primeniti, ili će potomci biti kopije roditelja. Autor u [36], sa druge strane, smatra da ukrštanje treba da se izvede nakon prethodnog odabira nove generacije. Hromozomi koji postaju subjekt ukrštanja se slučajno uparuju, a potomstvo koje se dobija na ovaj način ne proizvodi duplikate. Oba pristupa zamenjuju roditelje rezultujućim potomstvom.

1.6.4 Funkcija prilagođenosti

U cilju određivanja koliko je koja jedinka u okviru populacije dobra, definiše se funkcija prilagođenosti. Uobičajena primena genetskog algoritma je optimizacija neke funkcije. Nažalost, optimizacioni problemi su retko kad jednostavni i genetski algoritam je, pre svega, orijentisan ka optimizaciji kompleksnih funkcija više promenljivih i tzv. funkcija nad ne-numeričkim podacima, za koje ne postoje efikasna egzaktna numerička rešenja. Prirodno, što je kompleksniji problem, komplikovanija je i funkcija prilagođenosti. Kada se radi o numeričkim podacima, funkcija prilagođenosti može biti lako utvrđena iz samog optimizacionog problema. Najkomplikovanije funkcije prilagođenosti su one potrebne za evaluaciju ne-numeričkih podataka, kod kojih se mora tražiti adekvatna metrika za evaluaciju rešenja.

1.6.5 Operator selekcije

Kako broj jedinki u okviru populacije ukrštanjem raste, operator selekcije mora da se pobrine da veličina ipak ostane ujednačena. Ovaj operator određuje jedinke koje će ostati u sledećoj generaciji i obično se definiše tako da one jedinke, koje imaju bolju prilagođenost, imaju veće šanse. Najjednostavniji način definisanja operatora selekcije je čisto elitistički pristup, kod kojeg se biraju jedinke sa najvećom vrednošću funkcije prilagođenosti. Ovaj pristup je jednostavan za shvatanje i implementaciju. Međutim, ovo nije uvek najbolji izbor jer često može da uzrokuje "zaglavljivanje" u lokalnom optimumu. Drugi, češće korišćeni metod, je zasnovan na selekciji koja je proporcionalna prilagođenosti, ali sa dozom slučajnosti. Može se implementirati korišćenjem "ruletskog točka" [5, 35, 36, 39], gde svaka jedinka dobija svoj odsečak na točku veličine koja je propocionalna funkciji prilagođenosti. Na ovaj način, jedinke sa većom vrednošću funkcije prilagđenosti imaju veće šanse da budu izabrane. Ipak, često se koristi i kombinacija ova dva metoda, prema kojem određeni broj najboljih jedinki garantovano prelazi u narednu generaciju zarad ubrzavanja algoritma, dok ostali birane gore predloženim probabilističkim metodom, kako bi se osigurala raznolikost populacije. Jedna od popularnijih tehnika je i turnirska selekcija [7,19,40], kod koje se populacija deli u grupe od po n jedinki, koje se zatim "nadmeću" za prelazak u narednu generaciju. Ponekad je problem odabrati odgovarajući broj n, te postoji i metod tzv. fino gradirane turnirske selekcije, koji omogućava da u proseku taj broj ne bude ceo [20,21, 22].

1.6.6 Kriterijum zaustavljanja

Genetski algoritam ne pretražuje prostor rešenja sistematski kao što je to slučaj kod potpune enumeracije, te je potrebno definisati kriterijum zaustavljanja. Neki od poznatijih kriterijuma su: maksimalni broj generacija, maksimalni broj ponavljanja najbolje jedinke, stepen sličnosti jedinki, ograničeno vreme izvršavanja, dostizanje optimalnog rešenja (pod uslovom da je unapred poznato) i druge.

1.6.7 Ostali aspekti genetskog algoritma

Izvršavanje genetskog algoritma podrazumeva i niz drugih, pre svega implementacionih pitanja, kao što je način promene parametara (veličina početne populacije, koeficijent mutacije, ukrštanja itd.) i tu se ističu dva osnovna pristupa:

- fiksna promena parametara – podrazumeva podešavanje svih parametara pre početka izvršavanja algoritma.
- adaptivna promena – gde algoritam sam menja parametre u zavisnosti od prethodne uspešnosti. Detaljnije o temi adaptivnog podešavanja parametara genetskog algoritma može se pogledati u [8,3,4].

Pored ovoga važnost u izvršavanju genetskog algoritma imaju i drugi aspekti, kao što su npr. paralelizacija i keširanje [29, 30].

2. SMET

Aktuelni napredak na polju tehnologije mikročipova, bežičnih komunikacija i digitalne elektronike, doveli su do razvoja minijaturnih, jeftinih i multifunkcionalnih senzora, koji komuniciraju bez problema na malim udaljenostima. Upotreba ovakvih uređaja je u ekspanziji, a potencijalna područja praktične primene su, kao što je ranije navedeno, velike: nadgledanje okoline, kontrola brojnih prirodnih fenomena (zemljotresi, klizišta, padavine itd), vojne namene itd. U [1] se može pogledati detaljnije o mogućim primenama.

Bitno svojstvo bežične senzorske mreže je niska potrošnja energije. Postoje generalno najmanje dva grafovska problema čije rešavanje može dovesti do smanjenja troškova napajanja. Problem nalaženja energetski efikasnih ruta za komunikaciju u zadatoj mreži senzora je od izuzetnog značaja, no u ovom radu se analizira jedan konceptualno drugačiji problem, a to je problem dodeljivanja energije svakom čvoru mreže, tako da graf bude strogo povezan *(Definicija 1.9)*, a ukupna energija celog komunikacijskog sistema minimizovana (eng. general strong minimum energy topology – SMET) [2].

2.1 Formulacija problema

Definicija 2.1 SMET: Za dati skup senzora u ravni, odrediti vrednost energije koju treba dodeliti svakom senzoru, tako da između svakog uređenog para čvorova senzora postoji najmanje jedan usmereni put i ukupna potrošnja energije je minimalna [2].

Energija prenosa (transimisije) između senzora (čvorova) obeleženog indeksom i i čvora sa indeksom j se obično definiše kao:

$$f_{i,j} = f(d_{i,j}) = t_j (d_{i,j})^{\alpha} \tag{2.1}$$

gde $d_{i,j}$ predstavlja meru udaljenosti između čvorova, t_j prag osetljivosti (eng. detection sensitivity treshold) senzora j, tj. potrebna vrednost jačine signala dovoljna da ga senzor j detektuje. Kada je u pitanju prag osetljivosti, u literaturi se obično podrazumeva da je on jednak za sve senzore te se često njegova vrednost normalizuje na 1. Ovo ima za posledicu da problem, koji posmatramo, postaje simetričan. Konstanta α je vezana za stepen gubitka energije signala u zavisnosti od povećanja udaljenosti (eng. path loss), i ona obično ima vrednost 2 ili 4 [11, 12]. Ako posmatramo čvor i i njemu dodeljenu energiju z_i, za svaki čvor j za koji važi $f_{i,j} \le z_i$, kažemo da pripada oblasti signala čvora i, tj. moguće je poslati signal sa čvora i na čvor j. Neka je $G_d = (V, E)$ kompletan digraf *(Definicije 1.3, 1.4),* gde je V skup čvorova (senzora), a E je skup svih usmerenih grana. Grafovska formulacija SMET-a je sledeća:

Definicija 2.2 SMET: Za dati kompletan digraf G_d i funkciju energije $f_{i,j}$ odrediti pridružene vrednosti energije u svakom od čvorova grafa $\{z_1, z_2, ..., z_n\}$, tako da je podgraf dat sa

$G_d = (V, E')$, gde je $E' = \{f_{i,j} \mid f_{i,j} \le z_i, i, j \in V\}$ strogo povezan i ukupna energija data sa $\sum_{k \in N} z_k$ minimalna.

Iako ova formulacija problema deluje prilično jednostavno, pokazuje se da za praktične namene, npr. svođenje na problem celobrojnog programiranja, postoji varijanta sličnog problema, koji se definiše bez uvođenja novih promenljivih z_i. Taj problem može konceptualno drugačije da se interpretira, ali mu je matematička formualcija dosta slična, zapravo SMET predstavlja specijalizaciju tog problema. Radi se o tzv. SCSSP (eng. strongly connected spanning subgraph problem).

Definicija 2.3 SCSSP: Za dati kompletan digraf G_d i nenegativni skup grana $W = \{w_{i,j} \mid i, j \in V\}$ odrediti minimalni razapinjući strogo povezani podgraf $G_d = (V, E')$, takav da je $\sum_{i \in N} \max\{w_{i,j} \mid (i, j) \in E'\}$ minimalno.

2.1 Dosadašnji rezultati

U [9] se razmatra SMET i takođe iznosi dokaz o NP kompletnosti problema. U istom radu je pokazano da pod pretpostavkom simetričnih podataka, približni algoritam zasnovan na minimalnom razapinjućem stablu, daje u najlošijem slučaju dva puta lošije rešenje nego što je optimalno. U literaturi se prikazuju i dva polinomska heuristička algoritma bazirana na varijantama minimalnog razapinjućeg stabla [12, 34]. U [44] se predlaže i memetički algoritam, koji predstavlja kombinaciju lokalne pretrage i genetskog algoritma. Rezultat dobijen ovom metodom je bolji od rešenja koje daje približni algoritam zasnovan na minimalnom razapinjućem stablu. U skorije vreme, pojavio se i pristup zasnovan na kombinaciji linearnog programiranja i grananja sa odsecanjem. U [2] se iznosi nekoliko formulacija problema, kao i algoritam koji koristi grananje sa odsecanjem u kombinaciji sa LP (linearno programiranje) rešenjima u svakom čvoru drveta pretrage. Na osnovu ovih rešenja se vrši odsecanje neperspektivnih delova stabla.

2.1 Formulacija zasnovana na celobrojnom programiranju

Razmotrićemo sada jednu relativno jednostavnu formulaciju problema SCSSP, baziranu na celobrojnom programiranju. Neka je *A* skup binarnih promenljivih $x_{i,j}$, takvih da $x_{i,j} = 1$ ako je grana *(i, j)* odabrana u skup grana razapinjućeg podgrafa, inačne $x_{i,j} = 0$. Neka je z_i težina koja odgovara čvoru *i.* Ako se vratimo na SMET problem, z_i očigledno odgovara vrednosti energije koja je dodeljena čvoru *i.* Dalje, za dati skup čvorova *S,* takav da $S \ne E, i\ S \ne \varnothing$, neka je $(S, \overline{S})$ skup grana *(i, j),* koje povezuju čvorove iz skupa *S* sa skupom $E \setminus S$. Formulacija celobrojnog programiranja se definiše, u ovom slučaju, glasi [2]:

$$
\begin{aligned}
&\min \sum_{i=1}^{|V|} z_i, \\
&z_i \geq w_{i,j} x_{i,j} \quad \forall (i,j) \in E', \forall i \in E \\
&\sum_{(i,j)\in(S,\bar{S})} x_{i,j} \geq 1 \quad \forall S \subset E, S \neq E, S \neq \varnothing \\
&x_{i,j} \in \{0,1\} \quad \forall (i,j) \in E' \\
&z_i \geq 0 \quad \forall i \in E
\end{aligned}
\tag{2.2}
$$

gde je funkcija cilja zbir težina svih čvorova (tj. u slučaju SMET-a ukupna potrošnja energije za napajanje senzora). Prvi skup ograničenja definiše težine svakog čvora na taj način da zbir svih izlaznih grana mora biti manji ili jednak kapacitetu čvora (ne može se poslati signal senzoru koji nije u opsegu). Optimalno rešenje, očigledno, odgovara situaciji kada je $z_i = w_{i,j} x_{i,j}$. Kada je u pitanju drugi skup ograničenja, on je značajan jer osigurava da je podgraf strogo povezan. Kao što se vidi iz formule (2.2), za svaku podelu početnog skupa čvorova na dva disjunktna, postoji veza između bar dva čvora tako da prvi pripada jednom skupu, a drugi čvor drugom skupu. Korišćenjem svojstva tranzitivnosti, može se izvesti konstruktivan dokaz o tome da će, pod tim uslovima postojati, usmereni put između svaka dva čvora. I konačno, tu su i uslovi celobrojnosti,promenljivih $x_{i,j}$ i nenegativnosti težine čvora, tj. energije senzora.

3. GENETSKI ALGORITAM

U ovom poglavlju će biti predstavljen genetski algoritam za rešavanje problema određivanja minimalne energetske povezanosti u težinskom grafu – SMET. Implementacija ovog algoritma je izvršena na programskom jeziku C, korišćenjem prethodno razvijene programske platforme (pogledati [31]).

3.1 Kodiranje

U slučaju problema kombinatorne optimizacije, odabir načina kodiranja je obično vrlo jasan. Binarno kodiranje odgovara ovim tipovima problema, jer je generalna problematika vezana za odabir podskupa nekih elemenata. Potrebno je da odabrani podskup ulaznog skupa daje dobro rešenje. Prirodno se nameće binarno kodiranje, jer se ono upravo zasniva na skupu indikatorskih promenljivih, gde svaka određuje da li element pripada ili ne pripada rešenju problema. U našem problemu određivanja minimalne energetske povezanosti u težinskom grafu, iz formulacije (2.2) se može prirodno odrediti i način kodiranja. Dakle, prema formulaciji problema SCSSP, imamo skup A indikatorskih binarnih promenljivih $x_{i,j}$, takvih da $x_{i,j} = 1$ ako grana *(i, j)* pripada rešenju problema, 0 inače. Kodiranje će odgovarati ovoj formulaciji, tj. veličina hromozoma je jednaka broju grana, a vrednost će biti 1 ili 0 na odgovarajućoj lokaciji u hromozomu, u zavisnosti od toga da li je grana odabrana ili ne.

3.2 Funkcija cilja

Da bismo izračunali funkciju cilja, potrebno je da napravimo nekoliko koraka:

1. Pročitamo genetski kod (hromozom)
2. Prevedemo genetski kod u graf-baziranu strukturu
3. Proveravamo povezanost grafa Tarjanovim algoritmom
4. Ukoliko je graf povezan, računamo vrednost funkcije, inače jedinku proglašavamo za nevalidnu.

U poslednjem koraku računamo vrednost funkcije cilja, koja predstavlja zbir težina dodeljenih svakom čvoru u grafu. Kako mi rešavamo problem SCSSP-a, ono što dobijamo neće biti vrednosti težina, već težine izlaznih grana iz svakog od čvorova. Tako da je potrebno naći maksimum od svih tih izlaznih grana i to proglasiti za težinu čvora.

$$\sum_{i=1}^{n} \max\{f_{i,j} \mid x_{i,j} = 1\} \tag{3.1}$$

Vrlo je bitno primetiti da, u slučaju da graf nije povezan, što nam svakako po uslovima problema ne odgovara, jedinku elimišemo iz konkurencije za sledeću generaciju.

3.3 Tarjanov algoritam za proveranje povezanosti

U trećem koraku izračunavanja funkcije cilja potrebno je odrediti broj komponenti povezanosti mapirane grafovske strukture, tj. utvrditi da li je hromozom koji je kodira validan za dalje učestvovanje u izvršavanju genetskog algoritma.

Tarjanov algoritam [43], nazvan po Robertu Tarjanu, koji ga je i osmislio, predstavlja grafovski algoritam za pronalaženje strogo povezanih komponenti grafa *(Definicija 1.9),* kao i broj artikulacionih tačaka u grafu *(Definicija 1.12).* Osnovna ideja algoritma je pretraga u dubinu od nekog početnog čvora. Obilaskom čvorova formiraće se stabla, a broj tih stabala će odgovarati broju komponenti povezanosti.

Naivni pristup ovom problemu bi se zasnivao na pretrazi u dubinu počev od svakog čvora grafa, sa ciljem da se obiđe svaki put ceo skup čvorova. To bi impliciralo da je svaki čvor dostupan iz svakog drugog, što je upravo svojstvo stroge povezanosti. Međutim, složenost ovog algoritma, s'obzirom na složenost algoritma pretrage u dubinu koja iznosi $O(|V|+|E|)$, je $O(|V|(|V|+|E|))$, što je vrlo neefikasno. Sada će biti opisan algoritam složenosti $O(|V|+|E|)$, koji radi isto to.

Čvorovi se stavljaju na stek redosledom po kojem bivaju posećeni. Kada se pretraga vrati iz podstabla, čvorovi se skidaju sa steka, i određuje se da li je čvor koren strogo povezane komponente. Ako je čvor koren strogo povezane komponente, onda taj čvor i svi koji su skinuti sa steka pre njega se uklanjaju iz prostora pretrage i formiraju strogo povezanu komponentu. Pretraga se nastavlja od sledećeg slobodnog čvora, ukoliko je neki ostao, u potrazi za drugim komponentama povezanosti.

Suština algoritma je određivanje da li je čvor koren strogo povezane komponente. Da bi se ovo odredilo, svaki čvor dobija tzv. indeks dubine `v.index`, koji broji čvorove u redosledu posećivanja. Pored toga, svaki čvor dobija i vrednost `v.lowlink`, koji zadovoljava jednakost `v.lowlink=min{v'.index},` gde je v' dostižan iz v. Odavde sledi da je v koren strogo povezane komponente akko je `v.lowlink=v.index.` Vrednost za `v.lowlink` se izračunava tokom pretrage u dubinu i može se uvek po potrebi koristiti. U narednom delu teksta je prikazan pseudokod ovog algoritma.

```
Ulazt: graf  G = (V, E)
index←0
S←∅
foreach v in V do
        if v.index nije definisan then
                tarjan(v)
        endif
end foreach
```

```
procedure tarjan(v)
v.index←index
v.lowlink←index
index←index+1
S.push(v)
foreach (v,v') in E do
        if v'.index nije definisan then
                tarjan(v')
                v.lowlink←min(v.lowlink, v'.lowlink)
        else if v'∈S
                v.lowlink←min(v.lowlink, v'.index)
        endif
end foreach
if v.lowlink =v.index then
repeat
        v'=S.pop()
        print v'
until v'=v
```

U ovoj varijanti algoritma, izlaz je predstavljen ispisom pojedinačnih komponenti povezanosti, tj. čvorova koji pripadaju svakoj od njih. Za potrebe problema rešavanja SMET-a, potrebno je utvrditi da je broj tih komponenti povezanosti jednak 1, što predstavlja blagu modifikaciju gore predloženog algoritma. U slučaju da je broj komponenti povezanosti veći od 1, genetski kod jedinke, koja je predstavljala ovaj graf, proglašavamo za nevalidnu i eliminišemo iz konkurencije za selekciju.

3.4 Operatori GA

3.4.1 Selekcija

GA koristi fino gradiranu turnirsku selekciju (skr. FGTS) [19] koja je varijanta turnirske selekcije sa tom razlikom što srednja vrednost veličine turnirske grupe $N_{compavg}$ nije celobrojna, već se može zadati unapred nekom realnom vrednošću. U implementaciji ovog algoritma korišćena je vrednost $N_{compavg}$ =5.4.

3.4.2 Ukrštanje

Nakon procesa selekcije, vrši se uparivanje roditeljskog genetskog materijala i davanje potomstva. Ranije je napomenuto da je ideja ukrštanja bazirana na razmeni, tj. kombinovanju genetskog materijala roditelja. U ovoj implementaciji je vršena jednostavna razmena levo i desno od jedne odabrane tačke ukrštanja (jednopoziciono ukrštanje). Kada su u pitanju parametri

ukrštanja, koristi se stepen ukrštanja, koji opisuje sa kojom verovatnoćom se realizuje ukrštanje za formirane parove roditelja. Ovde je korišćena vrednost $P_{crossprob}$= 0.85, što znači da se u 85% slučajeva za formirane parove ukrštanje izvrši, a u 15% se ne vrši selekcija, već se odmah prelazi na operator mutacije.

3.4.3 Mutacija

Primenjena je mutacija sa zaleđenim bitovima. To znači da, ukoliko se u okviru cele populacije pojavljuju jedinke se fiksiranom vrednošću na nekom bitu, onda se primenjuje drugačija strategija mutacije. Naime, u tim situacijama jasno je da se prostor pretrage smanjuje za celu jednu dimenziju, te je potrebno odreagovati radikalnije i vratiti prostor pretrage u okvire normale. Stoga se mutacija, u slučaju tzv. zaleđenih bitova, vrši sa verovatnoćom od 1/n, gde je n dužina genetskog koda. Inače, pod normalnim okolnostima kada bit nije zaleđen, koristi se koeficijent 0.4/n.

3.5 Ostali aspekti GA

3.5.1 Veličina populacije i strategija zamene generacija

Pokazuje se da je optimalna veličina populacije povezana sa dimenzijom problema. U slučaju problema manjih dimenzija, korišćene su i veličine populacija od nekoliko desetina jedinki, međutim, čim su se dimenzije povećane, došlo se do neke „magične“ brojke od N=150 jedinki, koja daje dobre rezultate. Politika zamene je bazirana na elitističkom pristupu, gde je najboljih N_{elit} =100 jedinki u svakoj generaciji zadržano, što ostavlja prostor za stvaranje najviše N-N_{elit}=50 jedinki. Ovo omogućava da se u svakoj generaciji ne vrše sva ta intenzivna izračunavanja ponovo, nego se elitne jedinke samo prepisuju zajedno sa njihovim već izračunatim svojstvima. Ova metoda se naziva još i stacionarna elitistička strategija [30,31].

3.5.2 Keširanje

Keširanje nije obavezan segmenat genetskog algoritma, ali je svakako izuzetno koristan. Ideja je jednostavna: tokom računanja vrednosti funkcije cilja jasno je da jedinki sa nekom kombinacijom bitova odgovara tačno jedna vrednost funkcije cilja, bez obzira kada se ta jedinka pojavi (prvoj, petoj, desetoj generaciji). Pokazuje se da su jedinke sa istim genetskim kodovima dosta česta pojava, a za izračunavanje njihove funkcije cilja, podsetimo se, u ovom konkretnom problemu potrebno je uraditi niz zahtevnih operacija: izvršiti mapiranje strukture genetskog koda na grafovsku strukturu, izvršiti Tarjanov algoritam provere stroge povezanosti tog grafa i izvršiti sumiranje maksimalnih izlaznih grana za svaki od čvorova dobijenog grafa. Tek tako dobijena vrednost, u slučaju da je graf povezan, postaje validna vrednost funkcije cilja. Keširanje je proces pravljenja heš tabele gde su ključevi CRC vrednosti genetskog koda neke jedinke, a vrednost u tabeli je vrednost funkcije cilja. Koristi se tehnika poslednjeg najmanje korišćenog elementa

(eng. least recently used). Veličina keš memorije, u ovoj implementaciji, je ograničena na N_{cache}=5000 jedinki.

3.5.3 Kriterijum zaustavljanja

Program se može zaustaviti na nekoliko načina:

- Korisnik je prekinuo izvršavanje - ovde naravno ne postoji nikakva garancija kvaliteta poslednjeg rešenja
- Najbolja jedinka se nije promenila $N_{maxbest}$ uzastopnih generacija
- Najbolja vrednost se nije promenila N_{maxval} uzastopnih generacija
- Broj generacija je prešao N_{maxgen}
- Sličnost između jedinki u okviru jedne generacije je prevelika, preko P_{maxsim}

3.6 Strukture podataka

Program je modularan i omogućava razdvajanje dela podataka, koji služe za definisanje problema, od dela za optimizaciju. Tačnije, postoje dve paralelne grupe struktura, jedna za predstavljanje logike koja je prilagođena samom problemu, i druga koja vrši optimizaciju i generalne je namene. Osnovna optimizaciona struktura je `gastruct`:

```
typedef struct ga_struct
{
  item_struct *pop[MAXNITM];
  int codelen;
  int nitem;
  int iitem;
  int currentgen;
  item_struct *kbest;
  int type;
  unsigned int rnd;
  unsigned int seed;
  f_struct f;
  file_struct file;
  fitness_struct fitness;
  select_struct select;
  cross_struct cross;
  mut_struct mut;
  newgener_struct newgener;
  finish_struct finish;
  cache_struct cache;
} ga_struct;
```

Ukratko, namena bitnijih stavki iz ove strukture je sledeća:

- `pop[MAXNITM]` - informacije o jedinkama, sa njihovim genetskim kodovima i drugim vrednostima
- `codelen` - dužina genetskog koda

- `nitem` - broj jedinki u okviru populacije
- `iitem` – redni broj tekuće jedinke
- `currentgen` – tekuća generacija
- `kbest` – najboljih k jedinki
- `type` – tip optimizacije: minimizacija ili maksimizacija
- `rnd` – na početku slučajna populacija
- `seed` – osnova za generisanje slučajnih brojeva (eng. random seed)
- `f` – sadrži razne funkcije za učitavanje, inicijalizaciju, mapiranje iz grafovske u genetsku strukturu, izračunavanje funkcije cilja, ispisivanja rezultata, generisanje izveštaja i slično.
- `file` – pokazivači na razne pomoćne datoteke
- `fitness` – funkcija prilagođenosti
- `select` – informacije i parametri selekcije
- `cross` – informacije i parametri ukrštanja
- `mut` – informacije i parametri mutacije
- `newgener` – politika zamene generacija
- `finish` – kriterijum zaustavljanja
- `cache` – struktura za keširanje

Druga ključna struktura podataka se, kao što je već pomenuto, tiče definisanja samog optimizacionog problema.

```
typedef struct
{
    common_struct com;
    int nv;
    int nv;
    node_struct * v;
    edge_struct *e;
    double obj;
} problem_struct;
```

gde je:

- `nv` – broj čvorova u grafu
- `ne` – broj grana
- `v` – pokazivač na niz čvorova
- `e` – pokazivač na niz sa granama
- `obj` – vrednost funkcije cilja

Konačno, tu su i dve strukture prirodno nastale iz opisa problema: struktura čvora i grane grafa.

```
typedef struct node_struct
{
      int i;
      double z;
      int ns;
      int *s;
```

```
      edge_struct **pokS;
      int index;
    int lowlink;
    int isOnStack;
} node_struct;
```

Pri tome je:

- `i` – indeks čvora
- `z` – dodeljena težina u datom čvoru (energetska snaga datog senzora)
- `ns` – broj čvorova u koje može da se pređe iz ovog čvora
- `s` – indeksi tih čvorova
- `pokS` – niz sa pokazivačima na te čvorove
- `index, lowlink, isOnStack` – pomoćne promenljive u realizaciji Tarjanovog algoritma

```
typedef struct edge_struct
{
      int start;
      int end;
      char belong;
      double f;
} edge_struct;
```

Gde su:

- `start` – indeks čvora koji predstavlja početak ove grane
- `end` – indeks čvora koji predstavlja
- `belong` – da li grana pripada rešenju
- `f` – težina grane

4. EKSPERIMENTALNI REZULTATI

U ovom poglavlju će biti predstavljena serija rezultata koji su nastali kao posledica izvršenih eksperimenata vezanih za SMET. Izvršena su paralelna testiranja dva nezavisna mehanizma, sa jedne strane CPLEX rešavač predviđen da rešava probleme linearnog i kvadraturnog programiranja, i sa druge strane predloženi genetski algoritam. Genetski algoritam, kao što je i ranije naglašeno, implementiran je u programskom jeziku C i kompajliran na platformi Visual Studio 2008. Kada je u pitanju CPLEX, korišćena je verzija 12.1. Ona se inkorporira u vidu dinamičke biblioteke u okviru C baziranog projekta i koristi kroz javno dostupan API. Svi eksperimenti su izvršeni na PC-ju sa Intel procesorom na radnom taktu 2GHz pod Windows XP operativnim sistemom. Vremena izvršavanja su predstavljena u sekundama.

4.1 Test instance

Generisana su dva skupa slučajnih instanci težinskih grafova. Kako je bežična mreža uvek kompletna mreža, sve generisane instance su kompletni grafovi. Prvi skup predstavlja simetrične instance, kod kojih ulazna i izlazna grana između dva senzora ima istu težinu. To praktično znači da je energija transmisije signala u oba pravca ista i da kao što je to već pomenuto u poglavlju 2.1, važi pretpostavka da su pragovi osetljivosti svih senzora normalizovani. U okviru simetričnih instanci postoji 29 različitih dimenzija problema, najmanja je 10 čvorova, a najveća 150, pri čemu je svaka veća za pet čvorova od prethodne. Treba imati na umu da broj čvorova, sam po sebi, nije adekvatna mera ulazne dimenzije koja daje intuitivan osećaj o veličini problema. Kako se radi o kompletnim grafovima, ukupan broj grana, u slučaju n čvorova je:

$$2\frac{n!}{2!(n-2)!} \tag{4.1}$$

Tako, na primer, za n=10, dobijamo 90 grana, a u slučaju n=150 čak 22350 grana. Kako je osnova genetskog algoritma „naslonjena“ na proveru stroge povezanosti grafa, vidi se koliki značaj čak i najmanja dobit po pitanju performansi ovog segmenta algoritma ima na celokupnu složenost. Podsećamo da Tarjanov algoritam, koji je ovde primenjen, ima složenost $O(|V+E|)$. Raspored čvorova je određen slučajnim odabirom u okviru mreže celobrojnih tačaka [0,..,1000]x[0,...,1000], a težina grana između njih je određena po formuli $(d_{i,j})^2$, gde je $d_{i,j}$ udaljenost između dva čvora. U asimetričnom slučaju, parametar praga osetljivosti senzora, koji prima signal, nije normalizovan te se težina od čvora i ka čvoru j računa po formuli $t_j(d_{i,j})^2$, gde za t_j biramo neku realnu slučajnu vrednost iz opsega [1, 2].

4.2 Rezultati na simetričnim instancama

Jedan od važnijih segmenata rada sa optimizacionim algoritmima je svakako i verifikacija njihove korektnosti. No, u slučaju NP kompletnih problema svakako nije moguće izvršiti potpunu proveru korektnosti, ali je moguće to uraditi za manje dimenzije ulaznih veličina. Primena CPLEX-a je iz tog razloga bitna, kako bi se utvrdilo da predloženi GA daje optimalna rešenja za probleme manjih dimenzija. Samim tim sa većom sigurnošću „verujemo“ da će se pokazati dobro i kada su u pitanju veće dimenzije. Rešavanje problema datog kroz formulaciju zasnovanu na celobrojnom programiranju (2.2) pomoću CPLEX-a, u poređenju sa rezultatima genetskog algoritma na malim dimenzijama, je prikazano u tabeli 1 za dimenziju problema $|V|=10$, gde su data CPLEX rešenja optimalna i u tabeli 2 za dimenziju $|V|=15$, gde su rešenja uslovno optimalna, jer je CPLEX vremenski ograničen na 600 sekundi. U slučajevima malih dimenzija (10 i 15 čvorova), rešenje je traženo u 20 iteracija po svakoj instanci, a za veće dimenzije u 5 iteracija.

Instance	|V|	|E|	opt. CPLEX	GA reš.	t CPLEX	t GA
SMET_sim_t10_167	10	90	2808	**2808**	5,72	4,92
SMET_sim_t10_288	10	90	2576	**2576**	8,90	3,19
SMET_sim_t10_409	10	90	2439	**2439**	8,12	4,85
SMET_sim_t10_530	10	90	2243	**2243**	9,67	2,95
SMET_sim_t10_652	10	90	2562	**2562**	7,60	8,98
SMET_sim_t10_704	10	90	2568	**2568**	4,76	3,14
SMET_sim_t10_773	10	90	2507	**2507**	5,42	2,54
SMET_sim_t10_825	10	90	2050	**2050**	8,03	2,78
SMET_sim_t10_894	10	90	2305	**2305**	11,15	2,60
SMET_sim_t10_946	10	90	3182	**3182**	9,79	3,27

Tabela 1

instanca	|V|	|E|	usl. opt. CPLEX	GA reš.	t CPLEX	t GA
SMET_sim_t15_128	15	210	3274	3186	604,45	5,71
SMET_sim_t15_249	15	210	3418	3416	604,81	6,91
SMET_sim_t15_370	15	210	3906	3714	605,42	14,99
SMET_sim_t15_492	15	210	3142	3061	604,90	14,79
SMET_sim_t15_613	15	210	3407	3392	604,75	14,07
SMET_sim_t15_734	15	210	3189	3029	604,76	10,02

SMET_sim_t15_786	15	210	3140	3144	604,55	12,58
SMET_sim_t15_855	15	210	3206	3184	604,76	10,81
SMET_sim_t15_907	15	210	2845	2770	605,14	10,36
SMET_sim_t15_976	15	210	2739	2748	604,23	9,84

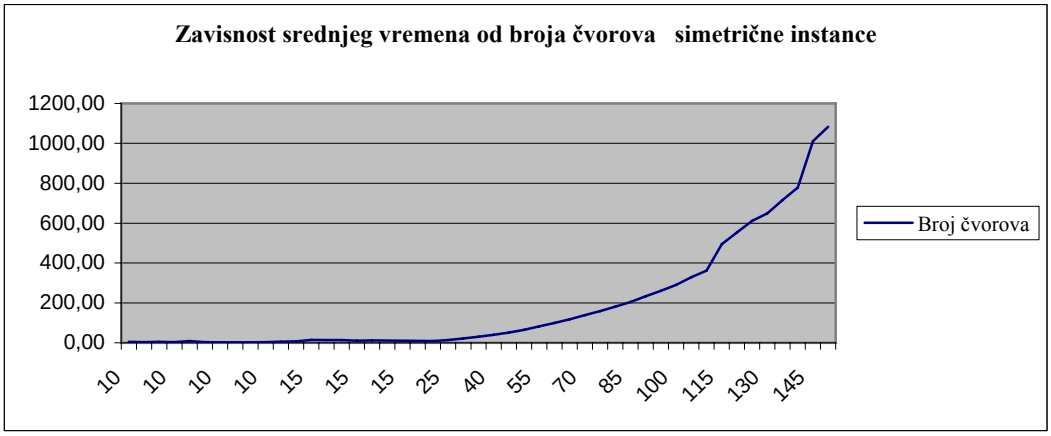
Tabela 2

Dakle, može se zaključiti da se genetski algoritam pokazao dosta dobro kod slučajeva manjih dimenzija. Kod dimenzija sa 20 čvorova, tj. 380 grana program CPLEX nije uspevao u doglednom vremenu da izvrši izračunavanje, te se ono prekidalo i za veće dimenzije nema provere optimalnosti. U tabeli 3 su dati rezultati za instance velikih dimenzija.

instanca	**|V|**	**|E|**	**opt. CPLEX**	**GA reš.**	**t CPLEX**	**t GA**
SMET_sim_t20_139	20	380	-	3452	-	8,80
SMET_sim_t25_169	25	600	-	3592	-	14,49
SMET_sim_t30_199	30	870	-	5031	-	22,05
SMET_sim_t35_592	35	1190	-	5499	-	30,91
SMET_sim_t40_154	40	1560	-	5314	-	40,79
SMET_sim_t45_478	45	1980	-	7056	-	52,12
SMET_sim_t50_872	50	2450	-	9506	-	65,98
SMET_sim_t55_365	55	2970	-	14338	-	82,84
SMET_sim_t60_758	60	3540	-	20482	-	100,20
SMET_sim_t65_251	65	4160	-	25685	-	118,68
SMET_sim_t70_644	70	4830	-	32816	-	139,51
SMET_sim_t75_138	75	5550	-	41310	-	159,00
SMET_sim_t80_531	80	6320	-	48929	-	182,00
SMET_sim_t85_924	85	7140	-	53125	-	205,03
SMET_sim_t90_417	90	8010	-	56150	-	233,59
SMET_sim_t95_810	95	8930	-	64377	-	260,88
SMET_sim_t100_304	100	9900	-	70570	-	290,81
SMET_sim_t105_697	105	10920	-	75360	-	328,51
SMET_sim_t110_190	110	11990	-	77702	-	361,57
SMET_sim_t115_583	115	13110	-	85969	-	493,76
SMET_sim_t120_977	120	14280	-	92110	-	552,44
SMET_sim_t125_470	125	15500	-	91528	-	611,06
SMET_sim_t130_863	130	16770	-	105252	-	649,08
SMET_sim_t135_356	135	18090	-	107350	-	715,28
SMET_sim_t140_749	140	19460	-	103376	-	777,69
SMET_sim_t145_243	145	20880	-	117689	-	1010,02
SMET_sim_t150_636	150	22350	-	125755	-	1081,47

Tabela 3

Na slici 1 se može videti zavisnost srednjeg vremena od broja čvorova, tj. grana.



Slika 1

4.3 Rezultati na asimetričnim instancama

Rezultati na malim asimetričnim instancama su dati u tabelama 4 i 5.

instanca	|V|	|E|	opt. CPLEX	GA reš.	t CPLEX	t GA
SMET_asim_t10_167	10	90	4084	**4084**	11,28	2,31
SMET_asim_t10_288	10	90	3576	**3576**	8,36	2,04
SMET_asim_t10_409	10	90	3340	**3340**	5,69	2,08
SMET_asim_t10_530	10	90	3435	**3435**	5,45	2,23
SMET_asim_t10_652	10	90	3641	**3641**	3,67	1,94
SMET_asim_t10_704	10	90	3851	**3851**	10,23	2,06
SMET_asim_t10_773	10	90	3583	**3583**	4,89	2,09
SMET_asim_t10_825	10	90	2573	**2573**	5,84	2,11
SMET_asim_t10_894	10	90	3028	**3028**	6,61	2,18
SMET_asim_t10_946	10	90	5452	**5452**	28,27	2,30

Tabela 4

instanca	|V|	|E|	usl.opt. CPLEX	GA reš.	T CPLEX	t GA
SMET_asim_t15_128	15	210	4391	4347	604,97	4,97
SMET_asim_t15_249	15	210	4578	4758	605,38	4,82
SMET_asim_t15_370	15	210	5926	5688	610,97	4,71

SMET_asim_t15_492	15	210	4114	4114	606,47	4,54
SMET_asim_t15_613	15	210	4747	4676	605,25	4,85
SMET_asim_t15_734	15	210	4410	4356	604,48	4,14
SMET_asim_t15_786	15	210	4635	4327	605,16	4,45
SMET_asim_t15_855	15	210	4371	4254	604,984	4,86
SMET_asim_t15_907	15	210	3779	3779	604,39	5,17
SMET_asim_t15_976	15	210	4594	4179	604,81	4,94

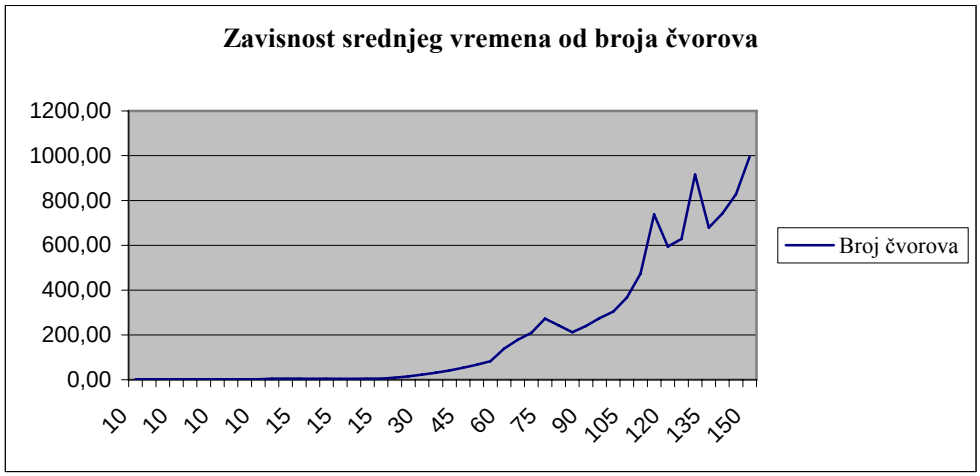
Tabela 5

U tabeli 6 su prikazani rezultati na asimetričnim instancama većih dimenzija.

instanca	|V|	|E|	opt. CPLEX	GA reš.	t CPLEX	t GA
SMET_asim_t20_952	20	380	-	1890	-	9,09
SMET_asim_t25_986	25	600	-	2065	-	14,95
SMET_asim_t30_121	30	870	-	2692	-	23,14
SMET_asim_t35_155	35	1190	-	2889	-	31,65
SMET_asim_t40_190	40	1560	-	3323	-	41,60
SMET_asim_t45_225	45	1980	-	4232	-	53,32
SMET_asim_t50_259	50	2450	-	4877	-	66,86
SMET_asim_t55_294	55	2970	-	8411	-	82,71
SMET_asim_t60_328	60	3540	-	11147	-	138,62
SMET_asim_t65_363	65	4160	-	17591	-	178,04
SMET_asim_t70_397	70	4830	-	21383	-	209,01
SMET_asim_t75_432	75	5550	-	24116	-	272,51
SMET_asim_t80_466	80	6320	-	28410	-	242,44
SMET_asim_t85_501	85	7140	-	29808	-	211,79
SMET_asim_t90_535	90	8010	-	37158	-	239,95
SMET_asim_t95_570	95	8930	-	39584	-	274,89
SMET_asim_t100_605	100	9900	-	42710	-	304,00
SMET_asim_t105_639	105	10920	-	49363	-	367,60
SMET_asim_t110_674	110	11990	-	54373	-	473,46
SMET_asim_t115_708	115	13110	-	59689	-	737,44
SMET_asim_t120_743	120	14280	-	61082	-	595,08
SMET_asim_t125_777	125	15500	-	65365	-	628,50
SMET_asim_t130_812	130	16770	-	70690	-	915,61
SMET_asim_t135_846	135	18090	-	73625	-	679,54
SMET_asim_t140_881	140	19460	-	74706	-	741,99
SMET_asim_t145_915	145	20880	-	80840	-	828,34
SMET_asim_t150_950	150	22350	-	91076	-	995,16

Tabela 6

Na slici 2 se može videti zavisnost srednjeg vremena izvršavanja od broja čvorova u slučaju asimetričnih instanci.


Zavisnost srednjeg vremena od broja čvorova
1200,00
1000,00
800,00
600,00
400,00
200,00
0,00
10
10
10
10
15
15
15
30
45
60
75
90
105
120
135
150
Broj čvorova

Slika 2

5. ZAKLJUČAK

U ovom radu prikazana je implementacija genetskog algoritma za rešavanje problema određivanja minimalne energetske povezanosti u težinskom grafu, koji se može primeniti na problem minimazije energetskih troškova u bežičnim senzorskim mrežama. Data je matematička formulacija, opis problema, opis implementacije genetskog algoritma, kao i prikaz dobijenih rezultata.

5.1 Pregled primenjenih metoda i budući rad

U prvom delu implementacije je razmatran što pogodniji algoritam za proveru ograničenja povezanosti grafa i implementiran je trenutno najefikasniji algoritam, koji to ograničenje proverava u linearnoj složenosti $O(|V|+|E|)$ u odnosu na ulaznu dimenziju problema. Kasnije je, primenom genetskog algoritma, rešen problem nalaženja optimalnih rešenja u slučajevima manjih dimenzija, kao i nalaženje rešenja za probleme velikih dimenzija. Korišćeno je binarno kodiranje koje se pokazalao da prirodno odgovara diskretnoj formulaciji problema. Osnovni genetski operatori su implementirani na način koji je u skladu sa načinom kodiranja i koji čuva korektnost jedinki u svakoj generaciji. Takođe se pokazalo da fino graduirana turnirska selekcija daje najbolje rezultate u kontekstu ovog problema. Pored toga, od bitnijih elemenata GA implementirano je jednopoziciono ukrštanje i osnovna mutacija sa zaleđenim bitovima, direktan prelazak najboljih jedinki (elitizam) u narednu generaciju. Od sporednih aspekata implementirano je i keširanje, koje je značajno ubrzalo vreme izvršavanja algoritma.

Dalje unapređivanje rezultata i proširivanje moguće je uraditi kroz neki od narednih koncepata:

- Paralelizacija predloženog genetskog algoritma kako bi se izvršavao i na multiprocesorskom računaru
- Kombinovanje (hibridizacija) algoritma sa drugim egzaktnim ili heurističkim metodama
- Rešavanje sličnih problema opisanim tehnikama
- Rešavanje opisanog problema sa složenijim modelom funkcije cilja, što odgovara drugačijim merama prenosa signala i drugačijim svojstvima senzorskih prijemnika.
- Robusnije ponašanje algoritma, u smislu pravljenja modela koji je sposoban da reaguje na otkaze senzora

5.2 Naučni doprinos rada

Doprinos ovog rada ogleda se u sledećim aspektima:

- Prvi put je korišćen genetski algoritam za rešavanje problema minimalne energetske povezanosti težinskog grafa prema opisanoj formulaciji.
- Za razmatrani SMET problem uvedena je odgovarajuća funkcija cilja, pa je na taj način bilo moguće koristiti binarno kodiranje, gde je sačuvana korektnost jedinki i omogućena efikasna implementacija

- Primenjen je najefikasniji trenutno poznati algoritam za proveravanje povezanosti grafa, što je omogućilo efikasnu konvergenciju
- Dobijeni su rezultati na instancama velikih dimenzija, koji su uporedivi sa rezultatima dobijenim korišćenjem drugih neegzaktnih algoritama, kao što je npr. kombinovana metoda pretrage sa odsecanjem i linearnog programiranja [2]
- Dobijeni su rezultati na instancama dimenzija koje nisu do sada razmatrane u literaturi, te se ne može govoriti o njihovoj uporedivosti

Rezultati rada na ovom problemu su u fazi pripreme za slanje nacionalnim i internacionalnim naučnim časopisima.

6. LITERATURA

[1] Akyildiz, I., Su, W., Sankarasubramaniam, Y., Cayirci, E., 2002. Wireless sensor networks: *A survey. Computer Networks 38, 393–422.*

[2] Y.P. Aneja, R. Chandrasekaran, Xiangyong Li, K.P.K. Nair, A branch-and-cut algorithm for the strong minimum energy topology in wireless sensor networks, *European Journal of Operational Research (2009)*

[3] Bëck T., "Self-adaptation in Genetic Algorithms", in: *Proceedings of the First European Conference on Artificial Life, MIT Press (1992).*

[4] Bëck T., "Optimal Mutation Rates in Genetic Search", in: *Proceedings of the Fifth International Conference on Genetic Algorithms, Morgan Kaufmann, San Mateo, Calif., pp. 2-8 (1993).*

[5] Bäck, Thomas, Evolutionary Algorithms in Theory and Practice *(1996), p. 120, Oxford Univ. Press*

[6] Bellman, R.E. 1957. Dynamic Programming. *Princeton University Press, Princeton, NJ. Republished 2003: Dover*

[7] T. Blickle, Evolving compact solutions in genetic programming: a case study In: H. Voigt, W. Ebeling, I. Rechenberg, and H. Schwefel (eds.), Parallel Problem Solving from Nature IV, *Proceedings of the International Conference on Evolutionary Computation, LNCS 1141, 564-573, 1996, Springer.*

[8] Bramlette M.F., "Initialisation, mutation and selection methods in genetic algorithms for function optimization", in: *Proceedings of the Fourth International Conference on Genetic Algorithms, Morgan Kaufmann, San Mateo, Calif., pp. 100-107 (1991).*

[9] Chen, W.-T., Huang, N.-F., 1989. The strongly connecting problem on multihop packet radio networks. *IEEE Transaction on Communications 37, 293– 295.*

[10] Cheng, C., Hu, T., 1991. Ancestor tree for arbitrary multi-terminal cut functions.
Annals of Operations Research 33, 199–213.

[11] Cheng, X., Narahari, B., Simha, R., Cheng, X., Liu, D., 2003. Strong minimum energy topology in wireless sensor networks: NP-completeness and heuristics. *IEEE*
Transactions on Mobile Computing 2, 248–256.

[12] Cheng, M.X., Cardei, M., Sun, J., Cheng, X., Wang, L., Xu, Y., Du, D.-Z., 2004. Topology control of ad hoc wireless networks for energy efficiency. *IEEE Transactions on Computers 53, 1629–1635.*

[13] Cook, S.A. (1971). "The complexity of theorem proving procedures". Proceedings, Third *Annual ACM Symposium on the Theory of Computing, ACM, New York. pp. 151–158. doi:10.1145/800157.805047.*

[14] A. Colorni, M. Dorigo et V. Maniezzo, Distributed Optimization by Ant Colonies, *actes de la première conférence européenne sur la vie artificielle, Paris, France, Elsevier Publishing, 134-142, 1991.*

[15] Cormen, T.H.; Leiserson, C.E., Rivest, R.L.; Stein, C. (2001). Introduction to Algorithms (2nd ed.). *MIT Press and McGraw-Hill. Chapter 34: NP–Completeness, pp. 966–1021.*

[16] Dantzig G. B, and Mukund N. Thapa. 1997. Linear programming 1: Introduction. *Springer-Verlag.*

[17] Dorigo, M. (1992). Optimization, Learning and Natural Algorithms *(Phd Thesis). Politecnico di Milano, Italie.*

[18] Euler, Solutio Problematis ad geometriam situs pertinentis, *Commentarii Academiae Scientiarum Imperialis Petropolitanae 8 (1736), pp. 128-140*

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

[20] Filipović V., Kratica J., Tošić D., Ljubić I., "Fine Grained Tournament Selection for the Simple Plant Location Problem", *Proceedings of the 5th Online World Conference on Soft Computing Methods in Industrial Applications - WSC5, pp. 152-158, (2000).*

[21] Filipović V., Tošić D., Kratica J., "Experimental Results in Applying of Fine Grained Tournament Selection", *Proceedings of the 10th Congress of Yugoslav Mathematicians, pp. 331-336, Belgrade, 21.-24.01. (2001).*

[22] Filipović, V. "Fine-Grained Tournament Selection Operator in Genetic Algorithms", *Computing and Informatics 22(2), 143-161.(2003).*

[23] Glover, F. (1986). "Future Paths for Integer Programming and Links to Artificial Intelligence". *Computers and Operations Research 13 (5): 533-549.*

[24] Glover, F. and M. Laguna. (1997). Tabu Search. *Kluwer, Norwell, MA.*

[25] Thomas Haenselmann (2006-04-05). Sensornetworks. GFDL Wireless Sensor Network textbook. Retrieved 2006-08-29.

[26] Holland, J.H. (1975). Adaptation in Natural and Artificial Systems. *University of Michigan Press.*

[27] Kirkpatrick, S.; Gelatt Jr., C.D.; Vecchi, M.P. (1983). "Optimization by Simulated Annealing". *Science 220 (4598): 671-680.*

[28] Kratica J.,Filipović V., Tošić D., "Solving the uncapacitated Warehouse Location problem by SGA Eith Add-Heuristic", *XV ECPD International Conference on Material Handling and Warehousing, University of Belgrade, Faculty of Mechanical Engineering, Materials Handing Institute, Belgrade (1998).*

[29] Kratica J., "Improving Performances of the Genetic Algorithm by Caching", *Computers and Artificial Inteligence, Vol. 18, No. 3, pp.271-283 (1999).*

[30] Kratica J., "Paralelizacija genetskih algoritama za rešavanje nekih NPkompletnih problema", *Doktorska disertacija, Matematički fakultet, Beograd (2000).*

[31] Kratica J., Tošić D., Filipović V., Ljubić I., "Solving the Simple Plant Location Problem by Genetic Algorithm", *RAIRO Operations Research, Vol. 73, No. 1, pp.127-142 (2001)*

[32] Kratica J., Tošić D., Filipović V., Ljubić. I., ”A genetic algorithm for the uncapacitated network design problem”, *Soft Computing in Industry – Recent Applications, Engineering series, pp. 329-338. Springer (2002).*

[33] Kratica J., Ljubić I., Tošić D., ”A genetic algorithm for the index selection problem”, *Springer Lecture Notes in Computer Science, Vol. 2611, pp. 281-291 (2003).*

[34] Lloyd, E., Liu, R., Marathe, M., Ramanathan, R., Ravi, S., 2002. Algorithmic aspects of topology control problems for ad hoc networks. In: *Proceedings of ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc’2002), Lausanne, Switzerland, June 9–11, pp. 123–134.*

[35] Mitchell, Melanie, (1996), An Introduction to Genetic Algorithms, *MIT Press, Cambridge, MA.*

[36] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolutionary Programs. *Springer-Verlag, 1992.*

[37] Papadimitiou, C. (1994). Computational Complexity (1st ed.). *Addison Wesley. Chapter 9 (NP-complete problems), pp. 181–218.*

[38] Römer, Kay; Friedemann Mattern (December 2004), "The Design Space of Wireless Sensor Networks", *IEEE Wireless Communications 11 (6): 54–61, doi:10.1109/MWC.2004.1368897*

[39] C. R. Reeves, Modern Heuristic Techniques for Combinatorial Problems. *McGraw-Hill Book Company, 1995.*

[40] O. Seng, M. Bauyer, M. Biehl and G. Pache, Search-based improvement of subsystem decomposition, In: *GECCO 2005: Proceedings of the Genetic and Evolutionary Computation Conference, 2005, 1045 – 1051.*

[41] O. Seng, J. Stammel and D. Burkhart, Search-based determination of refactorings for improving the class structure of object-oriented systems, In: *GECCO 2006: Proceedings of the Genetic and Evolutionary Computation Conference, 2006, 1909–1916*.

[42] Ingo Rechenberg (1971): Evolutionsstrategie - Optimierung technischer Systeme nach Prinzipien der biologischen Evolution (PhD thesis). *Reprinted by Fromman-Holzboog (1973).*

[43] Tarjan Robert: Depth-first search and linear graph algorithms. In: *SIAM Journal on Computing. Vol. 1 (1972), No. 2, P. 146-160.*

[44] Yang, K., Konstantinidis, A., Chen, H., Zhang, Q., 2007. Energy-aware topology control for wireless sensor networks using memetic algorithms. *Computer Communications 30, 2753–2764.*