SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 5389

Optimizacija parametara GPS napada uz pomoć evolucijskog algoritma

Fran Huzjan

Mentor: *prof. dr. sc. Domagoj Jakobović*

Zagreb, lipanj 2018.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
ODBOR ZA ZAVRŠNI RAD MODULA

Zagreb, 15. ožujka 2018.

ZAVRŠNI ZADATAK br. 5389

Pristupnik: **Fran Huzjan (0036496352)**
Studij: Računarstvo
Modul: Računarska znanost

Zadatak: **Optimizacija parametara GPS napada uz pomoć evolucijskog algoritma**

Opis zadatka:

Opisati mehanizam rada GPS sustava pozicioniranja i prikazati ga u obliku optimizacijskog problema. Navesti uobičajene postupke rješavanja problema GPS pozicioniranja. Opisati mogućnosti napada na sustav GPS te uvjete pod kojima su napadi mogući. Ostvariti programski sustav za optimizaciju parametara napada temeljen na evolucijskim algoritmima. Ispitati učinkovitost programskog rješenja na različitim scenarijima napada s obzirom na vremensku složenost i zadana ograničenja. Radu priložiti izvorne tekstove programa, dobivene rezultate uz potrebna objašnjenja i korištenu literaturu.

Zadatak uručen pristupniku: 16. ožujka 2018.
Rok za predaju rada: 15. lipnja 2018.

Mentor:
Prof. dr. sc. Domagoj Jakobović

Djelovođa:
Doc. dr. sc. Tomislav Hrkać

Predsjednik odbora za završni rad modula:
Prof. dr. sc. Siniša Srbljić

Velika zahvala mentoru prof. dr. sc. Domagoju Jakoboviću na ukazanom povjerenju, strpljenju, nesebičnoj pomoći, vodstvu i izdvojenom vremenu pri izradi ovog završnog rada.

Sadržaj

1. Uvod

U ovom završnom radu proučavat će se parametri GPS napada, njihova optimizacija i evolucijski algoritmi. Prikazat će se računanje pozicije primatelja signala, kao i vrijeme napadača, preko genetskog algoritma, u kojem treba poslati vlastiti signal kako bi uspješno odradio GPS napad. U poglavljima su objašnjeni osnovni pojmovi vezani za GPS i GPS napade, evolucijske algoritme uključujući genetski algoritam, određeni dijelovi implementacije rješenja, konkretan primjer problema i njegovo rješenje i grafički rezultati genetskih algoritama i njihova objašnjena.

2. GPS

2.1. Općenito o GPS sustavu

Globalni navigacijski satelitski sustavi (GNSS) su sustavi koji pružaju geoprostorno pozicioniranje na globalnoj razini. Neki od GNSS-a su američki *Global Positioning System* (GPS), ruski GLONASS, Galileo Europske unije (koji je još u inicijalnoj fazi implementacije). Republika Kina proširila je njihov regionalni navigacijski sustava Beidou u GNSS Compas 2015. godine. U okviru ovog završnog rada proučavat ćemo i baviti se američkim GPS-om. GPS omogućuje pozicioniranje, navigaciju i vremenske usluge na globalnoj razini svim korisnicima u svim vremenskim uvjetima bez obzira na vrijeme u danu. GPS-ova mreža se sastoji od 24 satelita koja šalju signale na GPS primatelje. Potrebno je napomenuti kako GPS može funkcionirati s minimalno 4 vidljiva satelita.

Pozicioniranje GPS-om provodi se u trodimenzionalnim (3-D) geocentričnom kartezijskom prostoru, koji je određen X, Y i Z koordinatama. Određivanje tih koordinata dobiva se iz formule :

$$d_i = \sqrt{(x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2}$$

gdje d_i predstavlja udaljenost između satelita i prijemnika, s_i vrijeme u kojem je satelit poslao signal, b greška sata primatelja signala i t_i trenutak u kojem je primljen signal, te se računa kao:

$$d_i = \left(\tilde{t}_i - b - s_i\right) c,\ i = 1, 2, \ldots, n$$

Kako je u svakom signalu satelita zapisano i u kojem je trenutku poslan signal, u formuli s_i, može se izračunati udaljenost satelita od prijemnika zbog pretpostavke da signali putuju brzinom svjetlosti.

Zbog nelinearnosti formule, koja se može riješiti različitim netrivijalnim matematičkim metodama, koristimo genetski algoritam.

2.2. GPS napad

Napad podvale (eng. *spoofing attack*) je situacija u kojoj osoba i/ili program (napadač) zavara "žrtvu" tako da joj pruži lažne podatke.

GPS napad (eng. *GPS spoofing*) je napad u kojem napadač zamaskira pravu GPS informaciju primatelja na željenu pogrešnu. Napadi mogu biti usmjereni na davanje lažnih koordinata ili pravih koordinata, ali s pogrešnim vremenom.

Jedan od češćih GPS napada je takozvani *carry-off attack* kada napadač sinkronizira svoje signale koje šalje sa signalima satelita koje primatelj treba primiti. S tom vrstom napada ću se baviti u ovom završnom radu. Pretpostavlja se da je hvatanje drona u Iranu 2011. godine rezultat takve vrste napada. Studenti sa "Cockrell School of Engineering" iz Texasa uspješno su izveli napad u kontroliranim uvjetima 2013. godine kada su jahtu "White Rose" preusmjerili lažnim GPS signalima [2]. Studenti su bili na jahti i koristili svoju opremu kako bi nadjačali prave GPS signale, te tako varirali smjer kretanja jahte.

Postoje razne vrste prevencije i zaštite od GPS napada. Preporučljiva je instalacija antena tako da nisu vidljive javno, dodavanje senzora ili blokera koji mogu detektirati kada se događa napad. Također se preporučuje korištenje više tipova GPS signala. Namjerno sam pisao preporučuje se, a ne zakonom je propisano jer još nije niti jednom pronađen dokaz koja ukazuje da su se GPS napadi koristili u zlonamjerne svrhe.

3. Evolucijski algoritmi

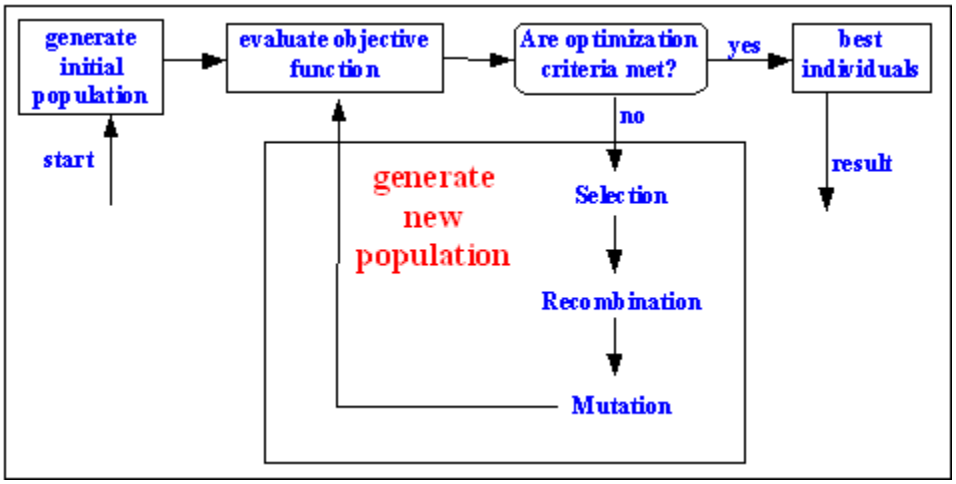
3.1. Općenito o evolucijskim algoritmima

Evolucijski algoritmi (engl. *evolutionary algorithms, evolutionary computing*) su postupci optimiranja, učenja i modeliranja koji se temelje na mehanizmu prirodne evolucije. To su formalni sustavi koji nastoje biti izomorfni s prirodnom evolucijom. Evolucijski algoritmi nastali su iz dviju pobuda: želje za boljim razumijevanjem prirodne evolucije i pokušaja primjene načela prirodne evolucije pri rješavanju različitih zadaća [3].

Za bolje razumijevanje evolucijskih algoritama potrebno je poznavati i shvaćati znanstvenu teoriju evolucije. Sva živa bića imaju DNK – nukleinsku kiselinu koja sadrži upute za biološki razvoj. DNK je genotip jedinke, dok je fenotip posljedica genotipa tj. manifestacija genotipa u izgledu. ponašanju i ostalim svojstvima jedinke. U računarstvu genotip i fenotip imaju isto značenje. Genotip je zapis jedinke u memoriji računala, dok je fenotip samo ponašanje te jedinke. Za razliku od genotipa u biologiji, genotip u računarstvu može biti u raznim zapisima, kao što su polje prirodnih i realnih brojeva, polje bitova, stabala i ostalo.

Temelje se na tome da se matematičke funkcije (ili nekakve druge vrste problema) mogu evoluirati u određenom okruženju, poput i živih bića. Kao okruženje se smatra populacija određenih jedinki (rješenja) navedenog problema. Jedno od najbitnijih svojstava evolucijskog algoritma je da prolazi kroz iteracije, poput pravih živih bića. Iteracija u smislu genetskog algoritma se nazivaju generacija populacija. U ovom završnom radu koristi se podvrsta evolucijskih algoritama, a to je genetski algoritam. Specifično za genetski algoritam je da svakoj jedinki pridodaje određenu dobrotu (eng. *fitness*) kojom se mjeri kvaliteta jedinke. Svaka jedinka u genetskom algoritmu prolazi kroz selekciju gdje se uklanjaju lošije jedinke i preživljavaju bolje, tj. jedinke s pogodnijim iznosom funkcije dobrote. Jedna od najbitnijih značajki

genetskog algoritma su mutacija i križanja. Nakon cijelog procesa očekivano je da se svakom generacijom dobivaju sve bolje i bolje jedinke. Bitna komponenta je i zaustavni kriterij rada genetskog algoritma. Najčešće se kao zaustavni kriterij uzima željeni iznos (minimalan ili maksimalan) funkcije dobrote. Ako se nikad taj iznos ne postigne može se koristiti i broj iteracija.



Slika 3.1.1. Struktura jedne generacije genetskog algoritma

3.2. Selekcija

U prirodnoj evoluciji selekcija se odvija tako da preživljavaju samo one vrste tj. jedinke koje uspiju preživjeti dovoljno dugo da kroz reprodukciju prenesu svoj genetski materijal na sljedeću generaciju. U računarstvu, u okvirima genetskog algoritma, selekcija se odvija pomoću već navedene funkcije koja računa dobrotu (eng. *fitness*) *jedinke.*

3.3. Reprodukcija

Reprodukcija ili križanje (eng. *crossover*) najčešće se misli na spolnu reprodukciju kojom dva roditelja kombiniraju svoja genetska svojstva i stvaraju novu jedinku s kombiniranim svojstvima oba roditelja. U evolucijskim algoritmima smo

slobodniji sa načinima križanja, te ih možemo provesti i bez roditelja tako da samo napravimo mutaciju.

3.4. Mutacija

U biologiji mutacija (eng. *mutation*) je promjena genetske poruke neke jedinke. Može se dogoditi spontano ili radi štete na DNK. Najčešće pojam mutacije vežemo uz negativne promjene, ali ponekad su dolazi i do korisnih mutacija. Upravo zbog toga koristimo mutaciju u evolucijskim algoritmima. Pomoću nje uvodimo sitne promjene u jedinki, te postoji mogućnost pojave boljih jedinki u idućoj generaciji.

3.5. Genetski algoritam

Genetski algoritmi su podskup evolucijskih algoritama. Najčešće su korišteni za pronalazak rješenja optimizacijskog problema ili problema pretrage.

Genetski algoritam provodi se nad populacijom jedinki koje predstavljaju rješenja problema. Svaka jedinka ima skup svojstava koje se mogu mijenjati i mogu biti izvedena na više načina (niz realnih brojeva, niz nula i jedinica, jedan broj, itd.)

Evolucija kao proces je iterativna i počinje sa populacijom nasumično stvorenih jedinki tj. imaju slučajno generirana svojstva. Kako je genetski algoritam podskup evolucijskih algoritama isto se određuje dobrota (eng. *fitness*) svake jedinke koja se računa preko funkcije koja je specifična za svaki problem. U svakoj generaciji preživljavaju jedinke sa najboljom dobrotom, te se njihovi geni rekombiniraju i mutiraju da bi stvorili novu generaciju i taj se postupak ponavlja sve dok se ne nađe rješenje ili dok broj generacija ne postane veći od zadanog. Ključno za genetski algoritam je da se koriste operatori reprodukcije, križanja i mutacije.

```
Genetski_algoritam
{
   t = 0
   generiraj početnu populaciju potencijalnih rješenja P(0);
   sve dok nije zadovoljen uvjet završetka evolucijskog procesa
   {
        t = t + 1;
        selektiraj P'(t) iz P(t-1);
        križaj jedinke iz P'(t) i djecu spremi u P(t);
        mutiraj jedinke iz P(t);
   }
   ispiši rješenje;
}
```

Slika 4.1 Pseudokod genetskog algoritma

Postoji više vrsta selekcije, a u praksi najčešće korištene su jednostavna selekcija (eng. *roulette wheel selection*) i turnirska selekcija (eng. *tournament selection*). Jednostavna selekcija funkcionira tako da izračuna vjerojatnost odabira jedinke tako da se iznos funkcije dobrote jedinke podijeli s ukupnom sumom svih vrijednosti funkcija dobrote. Turnirska selekcija odabire nekoliko jedinki iz populacije i drži "turnir" između njih. Pobjednik turnira radi križanje s drugom jedinkom.

$$p_i = \frac{f_i}{\Sigma_{j=1}^{N} f_j}$$

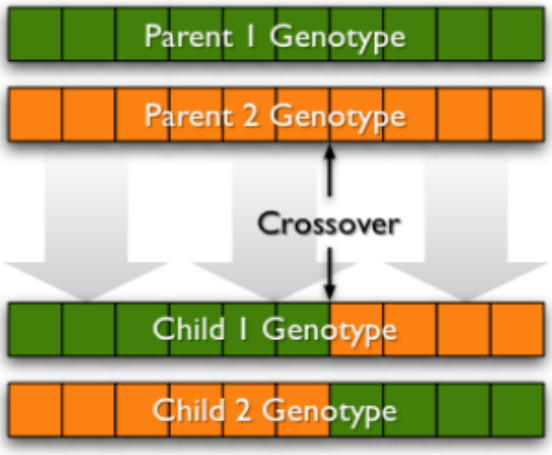
Slika 4.2 Računanje vjerojatnosti za odabir jedinke

odaberi k (veličina turnira) jedinki nasumično iz populacije
odaberi najbolju jedinku iz turnira sa vjerojatnosti p
odaberi drugu najbolju jedinku sa vjerojatnosti p*(1-p)
odaberi treću najbolju jedinku sa vjerojatnosti p*((1-p)^2)
i tako dalje

Slika 4.3 Pseudokod turnirske selekcije

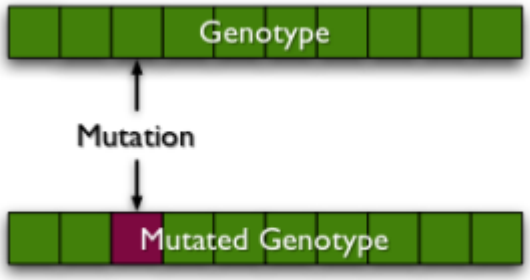
Najčešći oblik križanja je križanje s jednom točkom prekida (eng. *single point crossover*). Križanje s jednom točkom prekida odabire nasumično mjesto gdje će se

genotip jedinke "razrezati", te uzima suprotne dijelove roditelja kao što je prikazano na slici 4.4.



Slika 4.4 Križanje s jednom točkom presjeka

Mutacija služi kao sredstvo sprječavanja konvergencije u genetskom algoritmu. Kao najčešći primjer mutacije u genetskom algoritmu je ako imamo niz bitova, dodjeljuje se vjerojatnost mutacije (oko 1-3 %) koja označava da svaki bit može biti obrnut s tom vjerojatnošću.



Slika 4.5 Jednostavna mutacija

4. Optimizacija parametara GPS napada uz pomoć evolucijskog algoritma

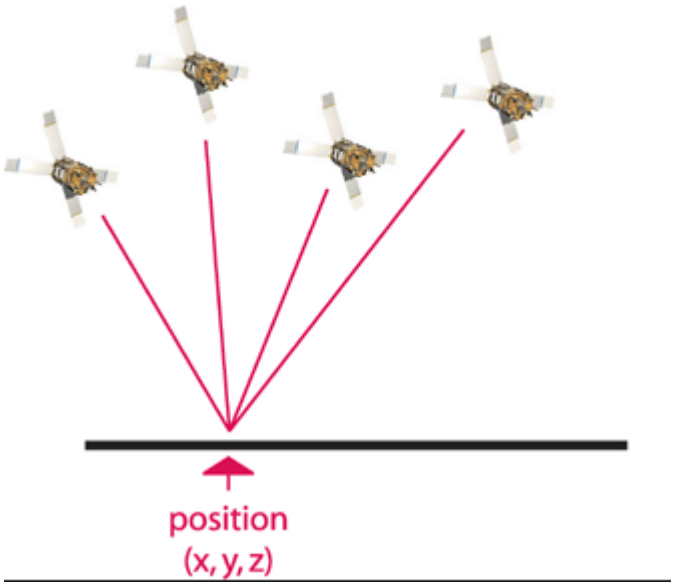
U ovom poglavlju je opisano kako je ostvarena programska potpora za optimizaciju parametara GPS napada genetskim algoritmom na primjeru pronalaska lokacije primatelja i namještanjem vremena signala napadača.

4.1. Ideja rješenja

Ideja ovog završnog rada je ostvariti model koji računa poziciju primatelja signala iz formule za GPS pozicioniranje genetskim algoritmom i odrediti vremena slanja lažnog signala potrebna napadaču koja su sinkronizirana s vremenima signala da bi ih primatelj krivo protumačio. Vremena napadača također se određuju pomoću genetskog algoritma.

Primatelj se nalazi na svojim nepoznatim koordinatama (x, y, z) i ima svoje vrijeme kada prima signal t_i. Sateliti također posjeduju svoje koordinate (x_i, y_i, z_i) i vrijeme u kojem je poslan signal s_i.

Napadač u sebi ima zapisana 4 vremena koja označavaju kada treba poslati signal da bi poremetio ispravnost primatelja.



Slika 4.1.1 Pojednostavljen prikaz GPS sustava

4.2. Implementacija rješenja

U ovom potpoglavlju ću se fokusirati na prikaz implementacije rješenja i strukturi koda i podataka.

4.2.1. Prikaz i implementacija jedinka

Jedinka primatelja signala, u kodu klasa *GaOrganism*, je implementirana kao polje *double* vrijednosti veličine 4. Prve tri vrijednosti polja predstavljaju koordinate primatelja, dok četvrta predstavlja vrijeme u kojem je primio signal. Vrijednosti polja se na početku inicijaliziraju na nasumične vrijednosti. Vrijednosti polja su u preciznosti rješenja sa 0.25 zbog diskretizacije u inicijalizaciji.

```java
public GaOrganism(int fieldSize) {

    int lower = -100;
    int upper = 100;


    this.field = new double[fieldSize];
    for (int i = 0; i < fieldSize; i++) {
        double rand = (int)(Math.random() * (upper-lower+1));
        double result = rand / 4;
        if (Math.random() < 0.5)
            result *= -1;

        this.field[i] =result;
    }

}
```

Slika 4.2.1.1 Implementacija Java koda inicijalizacije jedinke primatelja

Operator križanja kod jedinki primatelja implementirano je već navedenom metodom križanja s jednom točkom presjeka tako da se odredio nasumičan broj od 0 do 3, te se "prepolovilo" na nasumičnom broju i stvorila su se djeca s obje kombinacije genotipa. Zatim se s vjerojatnošću 0.5 uzima prvo, tj. drugo dijete.

```
Križanje {

        rand = nasumičan broj između 0 i 3

        za 0 do rand:
                dijete1[i] = roditelj1[i]
                dijete2[i] = roditelj2[i]

        za rand do veličina_polja_jedinke:
                dijete1[i] = roditelj2[i]
                dijete2[i] = roditelj1[i]

        p = nasumičan broj između 0 i 1
        ako p < 0.5:
                dodaj u listu dijete1
        inače:
                dodaj u listu dijete2
        vrati listu

}
```

Slika 4.2.1.2 Pseudokod za križanje jedinke primatelja

Operator mutiranja kod jedinke primatelja implementiran je tako da na svaki element polja jedinke dodajemo nasumičan broj Gaussove distribucije s vjerojatnošću od 0.015.

```
Mutacija {

        p = nasumičan broj između 0 i 1

        ako p < vjerojatnost mutiranja:
                za svaki element polja dodaj nasumičan broj [0,1] po
        Gaussovoj distribuciji

}
```

Slika 4.2.1.3 Pseudokod za mutaciju jedinke primatelja

Sljedeća jedinka koju smo evoluirali je jedinka vremena napadača (u kodu *AttackerOrganism*). Također se sastoji od polja, ali od 3 elementa. Napravljena je pretpostavka da prvi signal koji šalje je pogođen vremenski s prvim signalom satelita koji želi pogoditi radi napada. Operatori križanja i mutacija su isti kao i kod jedinke primatelja jer su oba iste strukture podataka (polje *double* brojeva).

```
public AttackerOrganism(int fieldSize){
    this.field = new double[3];
    for (int i = 0; i < fieldSize; i++) {
        if(i == 0)
            field[i] = ((int)( Math.random() * 128 * 16 )) / 16;
        else {
            field[i] = ((int)( Math.random() * 128 * 16 )) / 16
            + field[i-1];
        }
    }
}
```

Slika 4.2.1.4 Implementacija Java koda inicijalizacije jedinke napadača

4.2.2. Implementacija funkcije dobrote

Funkcije dobrote služe kako bi se odredila dobrota (eng. *fitness*) jedinke. U ovom završnom radu zbog evolucije dvije vrste jedinka potrebne su nam i dvije funkcije dobrote za svaku vrstu jedinke, te za obje funkcije dobrote tražimo njihove minimalne vrijednosti.

Funkcija dobrote primatelja zapravo je suma kvadriranih greška u računanju izračunate GPS pozicije i stvarne GPS pozicije podijeljena s 4 ($fitness = 0.25 * \sum_{i}^{n} ((\sqrt{((x-x_i)^2+(y-y_i)^2+(y-y_i)^2)}-(t_i-b-s_i)*c)^2), i=1,2,3,4$). U implementaciji funkcije dobrote postoje varijable *recieverTimeError, signalTimeTravel* i *signalRecieveTime*. Varijabla *signalRecieveTime* predstavlja t_i , *signalSendTime* s_i, a *recieverTimeError b*. Zadano parametrom u konfiguracijskoj datoteci možemo birati želimo li evoluirati i zadnji element polja jedinke primatelja, vrijeme. Ako se odlučimo za evoluciju tada u formuli koristimo zadnji element polja jedinke primatelja, a u suprotnom koristimo varijablu *recieverTimeError* koja je učitana iz konfiguracijske datoteke.

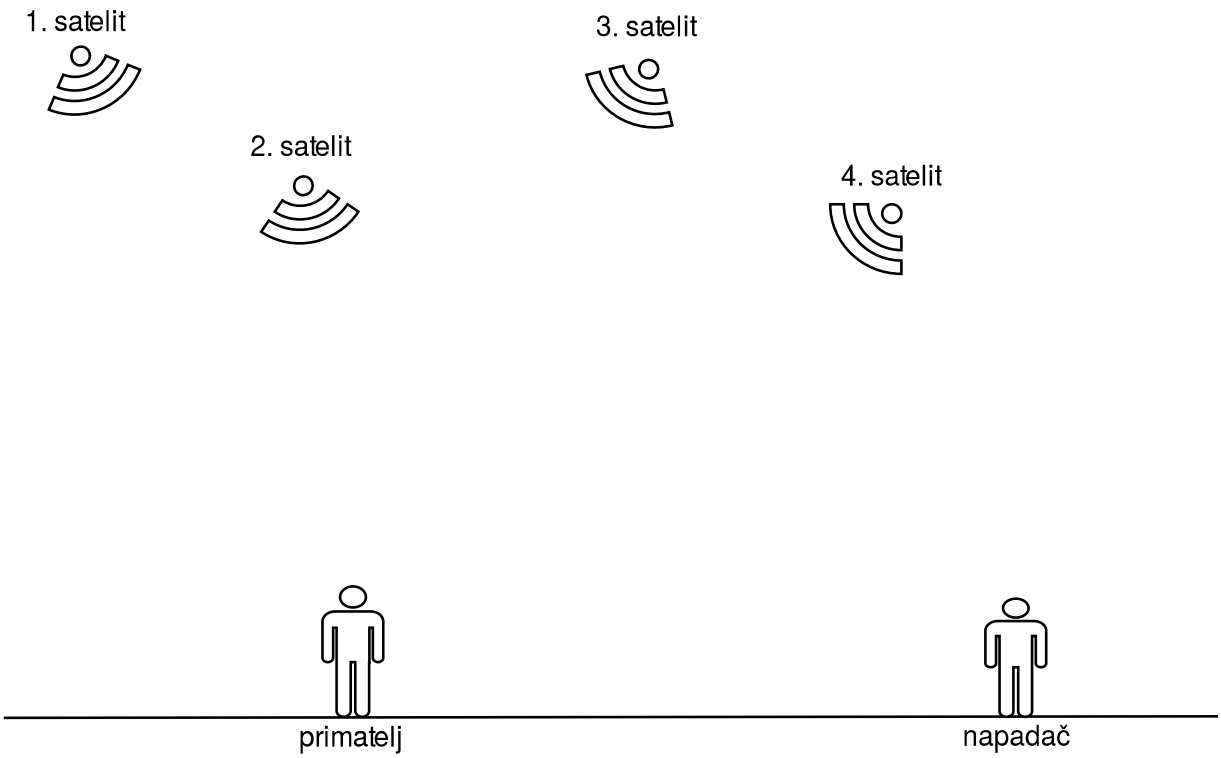
```
za svaki satelit:
    vrijemeSignala[satelit] = udaljenost(signala, primatelj) / c
    t_i[satelit] = s_i + vrijemeSignala + greškaPrimatelja

Za svaki satelit:
    greškaPozicije = (x-x_i)^2 + (y-y_i)^2 + (z-z_i)^2
    ako koristiGreškuPrimatelja:
        VremenskaGreška = (c*(vrijemeSatelita[satelit]) – greškaPrimatelja – s_i))^2
    inače:
        VremenskaGreška = (c*(vrijemeSatelita[satelit]) – b – s_i))^2
```

Slika 4.2.2.1 Pseudokod funkcije dobrote primatelja

Funkcija dobrote napadača zapravo je suma kvadriranih razlika vremena *i*-tog poslanog lažnog signala i razlike vremena poslanog signala satelita i vremena primljenog signala najbolje jedinke primatelja. $fitness = \sum_{i=1}^{3} (v_i - (t_i - s_i)^2)$ Također želimo da vrijednost ove funkcije dobrote bude minimalna.

4.2.3. Primjer

Primatelj se nalazi na koordinatama [-10, 25, -5] i prima signal u trenutku 8. Također postoje pripadajući sateliti s pripadajućim koordinatama : [0, -40, 1000], [300, 0, 1000], [-600, 0, 1000], [0, 700, 1000] i zadano je da svi sateliti šalju signale redom u trenutcima 0, 3, 7 i 2. Prvo se preko prvog genetskog algoritma nađu koordinate primatelja, koje odgovaraju zadanim koordinatama i vrijeme u kojem je signal primljen. Nakon toga genetski algoritam mora izračunati u kojim vremenima napadač mora poslati svoje lažne signale kako bi bili sinkronizirani sa signalima satelita. Algoritam izračunava da signali (pod pretpostavkom da je prvi signal već sinkroniziran) moraju biti poslani u trenutku -5 za sinkronizaciju s drugim satelitom, -1 za drugi satelit i -6 za treći satelit.



Slika 4.2.3.1 Skica konkretnog događaja

4.2.4. Simulacija

Razred *Main* sadrži metodu *main* u kojoj su izvedena oba algoritma i to tako da zapisuju rezultate svakog algoritma u zasebnu tekstualnu datoteku. U metodi prvo se čitaju podaci iz konfiguracijske datoteke. Primjer konfiguracijske datoteke:

0 -40 1000 0 #satelit1

300 0 1000 0 #satelit2

-600 0 1000 0 #satelit3

0 700 1000 0 #satelit4

-10.0 25.0 -5.0 7 #prijemnik

0 #timeError

Prva brojka satelita i prijemnika predstavlja *x* koordinatu, druga *y*, treća *z*, a četvrta vrijeme kada je signal primljen. Zadnji red predstavlja evoluiramo li i vrijeme primatelja ili ne. Ako stoji 0 znaci da se evolucija neće događati, a 1 obratno. Nakon čitanja iz datoteke instanciraju se razredi tipa *Satelite* za primatelja i satelite. Razred *Satelite* u sebi ima atribute *x, y, z* i *t* koji su svi tipa *double.*

Nakon toga algoritam započinje s radom i vrtnjom iteracija. U svakoj iteraciji napravi populaciju, selektira jedinke, križa i mutira ih i izračuna greška zadanih vrijednosti i dobivenih vrijednosti tj. vrijednosti funkcije dobrote. Na kraju svake iteracije algoritma u datoteku se upisuje broj iteracije, najbolja jedinka populacije i njezina vrijednost funkcije dobrote. Nakon izvršavanja određenog broja iteracija (u ovom slučaju 50) ili zadovoljavanja uvjeta prestanka, koji je kada vrijednost funkcije dobrote najbolje jedinke populacije zaokružena je jednaka nuli, dobivamo najbolju jedinku primatelja. Primjer ispisa rezultata u datoteku nakon pronalaska najbolje jedinke:

Iteration	Organism	Fitness
0	[-11.5, 28.25, -3.25, 7.0]	2536.853988759931
1	[-11.25, 25.75, -4.5, 7.0]	818.7480940202629
2	[-10.5, 25.75, -4.75, 7.0]	471.2125690034432

3	[-10.5, 25.75, -5.5, 7.0]	452.6232232658698
4	[-9.75, 24.5, -4.75, 7.0]	233.81156440119452
5	[-9.75, 24.5, -5.0, 7.0]	187.00461899294703
6	[-9.75, 25.0, -5.0, 7.0]	83.33526621530882
7	[-10.0, 25.0, -5.0, 7.0]	4.852375791015358E-8

[-10.0, 25.0, -5.0, 7.0]

Tada drugi algoritam, korišten za pronalazak vremena napadača započinje. Kod drugog algoritma, kao i kod prvog, provodi se selekcija, križanje i mutiranje jedinki populacije. Nakon svake iteracije se u drugu datoteku upisuju broj iteracije, najbolja jedinka i vrijednost funkcije dobrote najbolje jedinke. Algoritam se također vrti određen broj iteracija ili kada najbolja jedinka populacije nema zaokruženu vrijednost funkcije dobrote jednaku nuli.

Iteration	Organism	Fitness
0	[2.0, -17.0, -6.0]	182.0
1	[-10.0, -9.0, -5.0]	17.0
2	[-9.0, -9.0, -6.0]	9.0
3	[-7.0, -9.0, -5.0]	8.0
4	[-7.0, -7.0, -6.0]	1.0
5	[-7.0, -7.0, -6.0]	1.0
6	[-7.0, -7.0, -7.0]	0.0

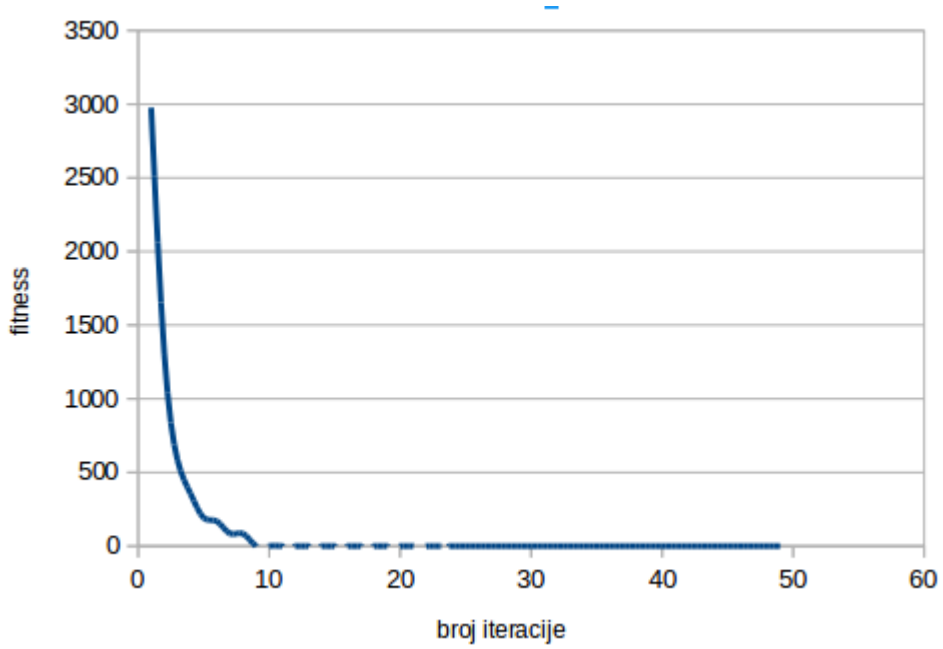
[-7.0, -7.0, -7.0]

5. Rezultati

U nastavku bit će prikazani rezultati prethodno opisanih algoritma na nekim ispitnim problemima. Parametri koji će se mijenjati su veličina populacije u algoritmima.

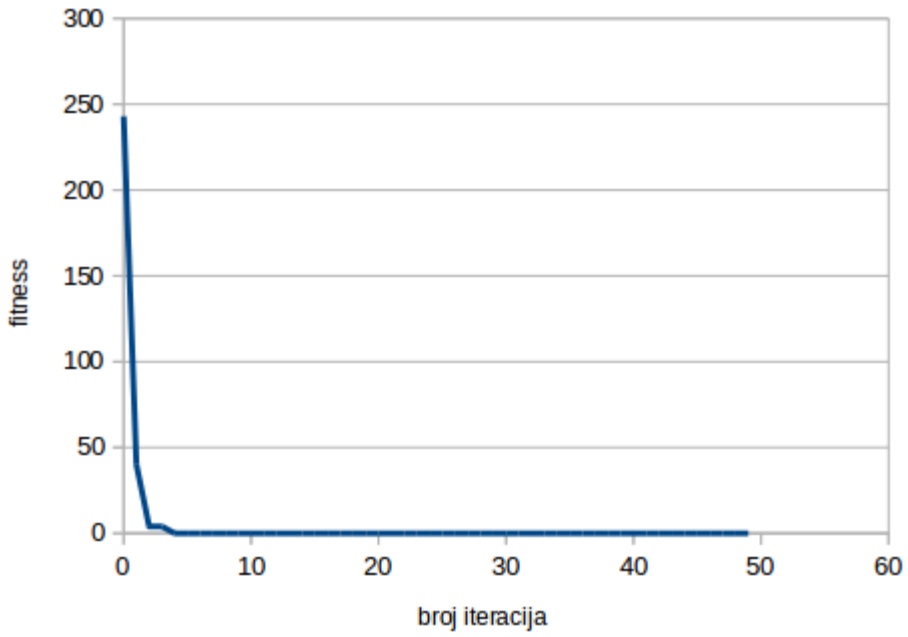
5.1. Kriterij zaustavljanja

Prikazani su slike grafova u kojima se vidi konvergencija vrijednosti funkcije dobrote jedinke primatelja. Metodom pokušaja i pogreške procijenio sam da je Zagreboptimalna veličina populacije 75000 jer tada algoritam u najmanje evaluacija pronalazi rješenje, i nakon 20 ispitanih slučajeva uočeno je da vrijednost konvergira u 0, jer je problem minimizacijski, nakon prosječno 7 generacija tj. 525 tisuća evaluacija. Isprobavajući s veličinama populacije u redu veličine 10^2 najmanja vrijednost funkcije dobrote bila je oko $7*10^7$, dok je u redu veličine 10^3 bila oko 3000. U tih 20 slučajeva u svim slučajevima je našao rješenje tj. niti jednom nije bio potreban maksimalan broj iteracija. Za konfiguraciju kada je veličina populacije 100000 algoritam u prosjeku pronalazi rješenje u 1530000 evaluacija, kada je populacija veličine 75000 u prosjeku 525000, a kada je veličina 50000 u prosječno 1025000 evaluacija. Ako se rješenje ne pronađe uzeto je da je broj iteracija potreban za rješenje 50. Na slici 5.1.1. prikazano je vrijednost funkcije jedinke primatelja najbolje jedinke populacije kroz broj iteracija.



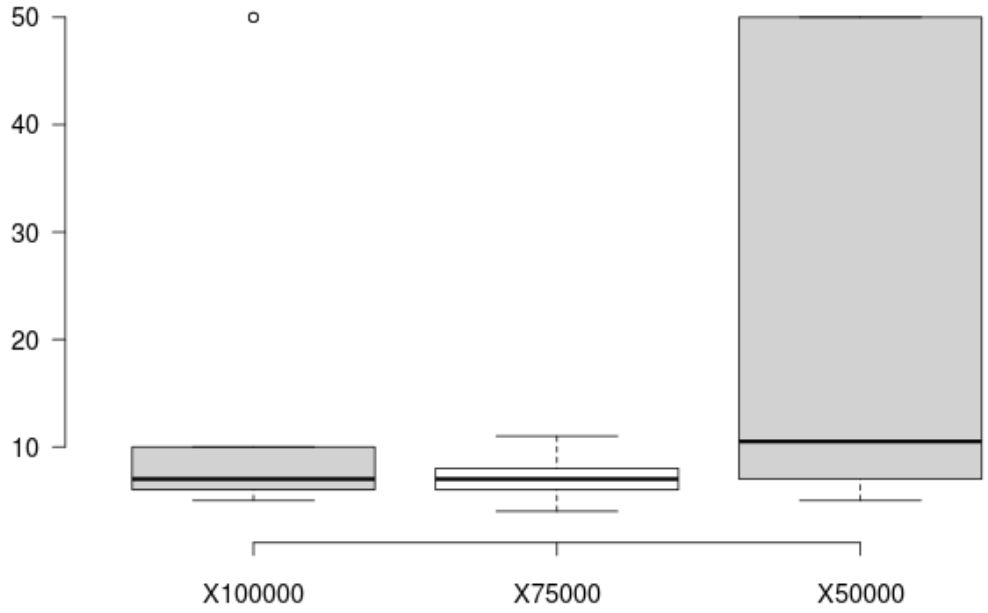
Slika 5.1.1 Graf ovisnosti vrijednosti funkcije dobrote i iteracija algoritma za jedinku primatelja

Nakon prikaza grafova za određivanje jedinke primatelja prikazujemo grafove algoritma za određivanje vremena napadača. Također metodom pokušaja i pogreške uočio sam da algoritam daje rezultate u najmanje evaluacija kada je veličina populacije 50000 i prosječna vrijednost iteracija kada kreće konvergencija je 6.9 što znači da je prosječan broj evaluacija potreban za konvergenciju 345 tisuće. Kada je veličina populacije bila u redu veličine 10^2 vrijednost funkcije dobrote jedinke bila je oko 1500, dok kada je veličina populacije u redu veličine 10^3 bila je oko 800. Na slici 5.1.2 prikazan je vrijednost funkcije dobrote jedinke napadača kroz broj iteracija.

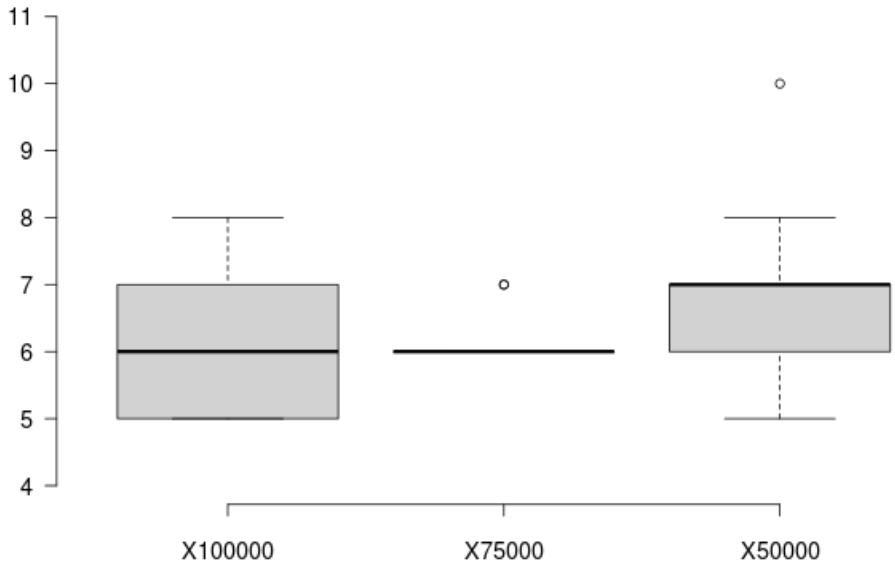


Slika 5.1.2 Graf ovisnosti vrijednosti funkcije dobrote i iteracija algoritma za jedinku napadača

Navedene slike boxplot grafova (slika 5.1.3 i slika 5.1.4), prikazuju prvo pojavljivanje rješenja u algoritmu, točnije kada vrijednost funkcije dobrote budu jednaka 0, u ovisnosti s veličinom populacije. Za svaku veličinu populacije algoritmi su pokrenuti 10 puta. Iz prikazanih slika grafova vidi se kako su metode pokušaja i pogreške zapravo dobro procijenile optimalnu veličinu populacije. Na slikama 5.1.3 i 5.1.4 vidi se kako je veličina populacije 75000 za prvi, i 50000 za drugi algoritam najkonzistentnija i s najmanje odstupanja za pronalazak prvog rješenja problema.



Slika 5.1.3 Boxplot pojavljivanja prve vrijednosti nula funkcije dobrote ovisne o veličini populacije za jedinku primatelja



Slika 5.1.4 Boxplot pojavljivanja prve vrijednosti nula funkcije dobrote ovisne o veličini populacije za jedinku napadača

Konkretna rješenja za navedene pozicije i vremena satelita i veličinu populacije bit će prikazani u sljedećoj tablici.

Veličina populacije (prvog i drugog algoritma)	Pozicija i vremena slanja signala satelita $[x_i, y_i, z_i, t_i]$	Dobiveno rješenje primatelja [x, y, z, t]	Vrijednost funkcije dobrote primatelja	Dobivena vremena napadača $[t_2, t_3, t_4]$	Vrijednost funkcije primatelja napadača
1. 75 000 2. 50 000	1. [0 ,-40, 1000, 0] 2. [300, 0, 1000, 3] 3. [-600, 0, 1000, 7] 4. [0, 700, 1000, 2]	[-10., 25.0, -5.0, 8]	$1.23*10^{-7}$	[-5.0, -6.0, -1.0]	0.0

6. Zaključak

Iako određivanje GPS položaja genetski algoritmom funkcionira i uspješno pronalazi rješenje, moguće je jednadžbu riješiti na "pravi" matematički način i to bi bilo efikasnije i brže. Isto vrijedi i za pronalaženje vremena napadača iako je za taj problem potrebno manje znanje matematike.

Budući da su ovo relativno jednostavni i ne hardverski zahtjevni problemi rješenja se dobiju brzo čak i na osobnim računalima koja su namijenjena za svakodnevno korištenje. Međutim da se radi o nekakvim kompleksnijim i težim problemima vrijeme čekanja da se jedan, a tek dva algoritma izvrše bilo bi preveliko i neučinkovito.

7. Literatura

1. UT Austin Researchers Successfully Spoof an $80 million Yacht at Sea. The University of Texas at Austin.

2. Darko Grundler. Evolucijski Algoritmi. URL hrcak.srce.hr/file/8743

3. Marko Čupić. Evolucijsko računarstvo

4. Marko Čupić. Motivacijski primjer: robot Robby

5. Marko Čupić: Prirodom inspirirani optimizacijski algoritmi. Inačica 2009.

6. Ching-Hung Wu; hung-Ju Chou;Wei-Han Su:Direct transformation of coordinates for GPS positioning using the techniques of genetic programming and symbolic regression

8. Sažetak – abstract

Sažetak

Naslov : Optimizacija parametara GPS napada uz pomoć evolucijskog algoritma

U sklopu ovog završnog rada opisan je mehanizam rada GPS sustava i prikazan je u obliku optimizacijskog problema. Navedeni su uobičajeni postupci rješavanja GPS pozicioniranja, te su i opisani mogućnosti GPS napada i uvjeti kada je napad moguć. Ostvaren je programski sustav za optimizaciju parametara GPS napada koji je temeljen na evolucijskom algoritmu, preciznije genetskom algoritmu. Sustav je ispitan na različitim scenarijima napada s obzirom na vremensku složenost i zadana ograničenja. Dobiveni rezultati prikazani su grafički i zatim analizirani.

Ključne riječi: optimizacija parametara, GPS, GPS napad, GPS pozicioniranje, evolucijski algoritmi, genetski algoritam

Abstract

Title: Optimization of GPS attack parameters with evolutionary algorithm

In this work the mechanism of GPS system is described and presented as an optimization problem. The usual procedures for resolving GPS positioning are mentioned, as well as capabilities of GPS spoofing and conditions when it is possible. A system build for optimization of GPS spoofing parameters was developed. It is based on evolutionary algorithm, more precisely a genetic algorithm. The system was tested at various scenarios of spoof based on computational complexity and given limits. The obtained results were shown graphically and then analyzed.

Keywords: parameter optimization, GPS, GPS spoofing, GPS positioning, evolutionary algorithm, genetic algorithm