# Genetski Algoritmi i Problem Trgovačkog Putnika

Daniel Silađi
Gimnazija "Jovan Jovanović Zmaj"
Novi Sad

6. maj 2016

**Sažetak**

Ovaj rad se bavi primenom genetskih algoritama (GA) na problem trgovačkog putnika (TSP). U prvom delu rada, data je definicija problema trgovačkog putnika, istorijski pregled problema i algoritama za njegovo rešavanje. Takođe su navedeni osnovni pojmovi vezani za genetske algoritme.

Originalni doprinos rada je program koji rešava problem trgovačkog putnika pomoću genetskog algoritma. Algoritam je implementiran u programskom jeziku C++, uz pomoć biblioteka SDL, OpenGL i GALib. U ovom radu je dat i tehnički opis, način korišćenja programa, kao i poređenje sa rezultatima koji se dobijaju sa drugim algoritmima iz literature.

# 1 Uvod

## 1.1 Šta je problem trgovačkog putnika?

Neformalno, problem trgovačkog putnika (engl. Travelling Salesman Problem - TSP) možemo formulisati na sledeći način: Dato je n gradova, poznate su sve udaljenosti među njima; trgovački putnik treba da obiđe sve te gradove i da se na kraju vrati u grad odakle je krenuo, a da pri tome pređe najkraću razdaljinu. Formalna definicija problema trgovačkog putnika[Cormen et al., 2001] glasi:

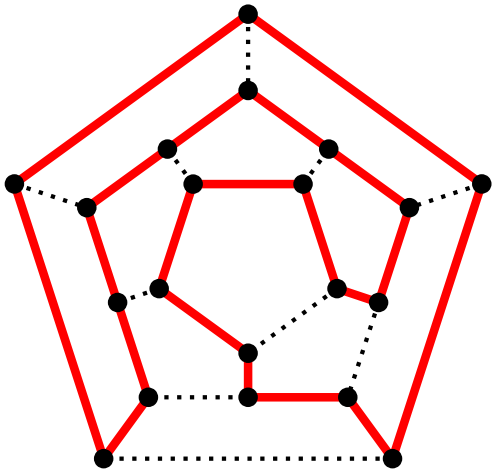
**Definicija 1.1 (Problem trgovačkog putnika-TSP).** Ako je dat kompletan neusmeren težinski graf, sa nenegativnim celobrojnim težinama naći Hamiltonovu putanju sa najmanjom težinom.

Pored optimizacijske varijante ovog problema, postoji i tzv. forma problema odlučivanja:

**Definicija 1.2 (TSP - odlučivanje).** Ako je dat kompletan težinski graf i pozitivan realan broj $L$, treba odrediti da li postoji Hamiltonov put kraći od $L$?

### 1.1.1 Istorija

TSP prvi put spominju irski i britanski matematičari Sir William Rowan Hamilton i Thomas Penyngton Kirkman sredinom XIX veka. Hamilton je 1857. godine izmislio Ikozijansku igru, u kojoj je cilj da se nadje zatvorena putanja koja prolazi kroz (neke) ivice dodekaedra i svako njegovo teme tačno jednom. U današnjoj terminologiji, to bi značilo naći Hamiltonovu putanju u grafu kome su čvorovi temena, a ivice ivice dodekaedra.



Slika 1: Ikozijanska igra

Nakon njih, sledeća osoba koja se ozbiljno bavila TSP-om je bio Karl Menger, 1930-ih godina. On je u svom radu spomenuo trivijalni brute-force algoritam, konstatovao suboptimalnost heuristike „najbližeg komšije" i definisao TSP onako kako se danas definiše. U narednim decenijama, problem je postajao sve više popularan i izučavan od strane mnogih matematičara, informatičara, fizičara,...

### 1.1.2 Algoritmi za rešavanje TSP

Očigledno, TSP se može rešiti brute-force algoritmom(složenosti $\mathcal{O}(n!)$) isprobavajući sve moguće permutacije gradova. Danas nije poznat ni jedan determinističko polinomni algoritam za njegovo rešavanje - TSP spada u klasu NP tvrdih problema. Međutim, Datzig, Fulkerson i Johnson su u svom radu [Dantzig et al., 1954] dali metodu koja se zasniva na celobrojnom programiranju, i iako je još uvek eksponencijalne složenosti, znatno je brža od svih ostalih do sada poznatih metoda [Applegate et al., 2003]. Postoje takođe i aproksimativni algoritmi, npr. genetski algoritmi, Lin-Keringan potezi, tabu pretraživanje, itd.

## 1.2 Šta su genetski algoritmi i čemu služe?

Genetski algoritmi (u daljem tekstu GA) su porodica algoritama inspirisanih Darvinovom teorijom evolucije. Prvi radovi iz ove oblasti su nastali 60-tih godina prošlog veka, ali se u većini izvora kao tvorac ove oblasti uzima John Holland. On je 1975. godine napisao knjigu [Holland, 1992]. GA imaju za cilj rešavanje problema kombinatorne optimizacije, tj problema u kojima se traži minimum ili maksimum neke funkcije . Pošto je prostor pretraživanja (skup rešenja) ponekad prevelik (i njegovo kompletno pretraživanje se ne može izvršiti u nekom doglednom vremenu), a optimalno rešenje nije neophodno (prihvata se i neko približno, suboptimalno rešenje), mogu se koristiti genetski algoritmi.

U GA, svako pojedinačno rešenje je predstavljeno jednom jedinkom, koja sadrži gene, tj delove rešenja. Nad njima se vrše operatori mutacije i ukrštanja (kao mehanizam pretrage) i selekcije (usmerava algoritam ka perspektivnim delovima pretraživačkog prostora).

## 1.3 Osnovni operatori u genetskim algoritmima

**Definicija 1.3. Fitness funkcija** je genetski operator koji svakoj jedinki dodeljuje vrednost $f_i$ koja oslikava kvalitet te jedinke.

Kod TSP, to je ukupna dužina puta predstavljenog tom jedinkom.

Sledeći operator je operator selekcije. U osnovnim crtama, princip rada selekcije je sličan kao u stvarnom životu: cilj je da se odabere genetski materijal koji ce se preneti u narednu generaciju. Kod ovog operatora je bitno sačuvati raznovrsnost, kao i kvalitet genetskog materiala, inače će se dobra rešenja možda zauvek izgubiti.

**Definicija 1.4. Selekcija** je genetski operator koji bira jedinke koje će se ukrštati i/ili preneti u sledeću generaciju.

**Definicija 1.5. Ukrštanje** je genetski operator koji na osnovu 2 date jedinke (roditelja) konstruiše 2 nove jedinke (decu), koje su nastale kombinovanjem genetskog materijala roditelja.

Cilj ovog operatora je da se ukrštanjem dva postojeća rešenja dobiju nova, obično kvalitetnija rešenja.

**Definicija 1.6. Mutacija** genetski operator koji (uglavnom) nasumično mutira genetski materijal date jedinke.

Mutacijom se vraća raznovrsnog genetskog materijala u populaciju. Ipak, prekomerna mutacija može svesti algoritam na nasumičnu pretragu, pa se zato uvodi verovatnoca mutacije, koja je uglavnom manja od 5%.

# 2 Korišćene metode

## 2.1 Konstrukcija početnog rešenja

Za konstrukciju početnog rešenja kojišćen je greedy algoritam koji od zadatog početnog grada gradi put tako što uvek bira grad najbliži poslednjem dodatom gradu. Iako je ovaj pristup suboptimalan, u praksi najčešće daj prilično dobra rešenja koja se od optimalnog retko razlikuju za više od 10%.

## 2.2 Selekcija

U ovom radu su korišćena su dva algoritma za selekciju:

*Napomena* 2.1. Promenljiva $f_i$ predstavlja vrednost fitness-funkcije $i$-te jedinke u populaciji, $p_i$ verovatnoću da bas $i$-ta jedinka bude odabrana prilikom selekcije, a $r_i$ rang $i$-te jedinke, tako da najkvalitetnija jedinka (gledajući fitness) ima rang $n$, a najnekvalitetnija 1. Takođe, važi pretpostavka da u populaciji ima tačno $n$ jedinki.

**Prosta, rulet selekcija** Svaka jedinka ima verovatnoću selekcije direktno proporcionalnu njenoj fitness-vrednosti, tj:

$$p_i = \frac{f_i}{\sum_{i=1}^{n} f_j}$$

**Selekcija zasnovana na rangu** Svaka jedinka ima verovatnoću selekcije direktno proporcionalnu njenom rangu, tj:

$$p_i = \frac{r_i}{\sum_{j=1}^{n} r_j}$$

## 2.3 Ukrštanje

Korišćena su 2 algoritma za ukrštanje:

**Ukrštanje rekombinacijom grana (edge recombination)** Ovaj algoritam od 2 roditelja pravi 1 dete, i to na sledeci nacin:

ALGORITAM: *1. Konstruiše matrice povezanosti:*

$$\begin{array}{cccc} g_1: & g_n & g_2 \\ g_2: & g_1 & g_3 \\ \vdots & \vdots & \vdots \\ g_n: & g_{n-1} & g_1 \end{array}, \quad \begin{array}{ccc} g'_1: & g'_n & g'_2 \\ g'_2: & g'_1 & g'_3 \\ \vdots & \vdots & \vdots \\ g'_n: & g'_{n-1} & g'_1 \end{array}$$

*2. Napravi njihovu uniju*

$$\begin{array}{cccccc} g_1 = g'_{i_1}: & g_n & g_2 & g'_{i_1-1} & g'_{i_1+1} \\ g_2 = g'_{i_2}: & g_1 & g_3 & g'_{i_2-1} & g'_{i_2+1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ g_n = g'_{i_n}: & g_{n-1} & g_1 & g'_{i_n-1} & g'_{i_n+1} \end{array}$$

*3. Napravi novu praznu listu $L$*

*4. Odabere početni čvor $L_1$*

*5. Odabere cvorove $L_2, \ldots L_{n-1}, L_n$ po sledecem principu: $L_{i+1}$ je sused čvora $L_i$, takav da $L_{i+1}$ ima najmanji mogući broj suseda, i ne nalazi se već u $L$*

*6. Lista $L$ je nova jedinka*

**Ukrštanje parcijalnim mapiranjem** Ovaj algoritam proizvodi 2 potomka od 2 roditelja.

ALGORITAM: *1. Odaberu se 2 cvora po slučajnom izboru, a i b*

*2. Iz roditelja 1 se kopira put od a do b u dete 1, a iz roditelja 2 u dete 2.*

*3. Dokle god to može (tj. dok se svaki čvor javalja samo jednom u detetu), algoritam popunjava pozicije u detetu 1 sa putevima iz roditelja 2 i u detetu 2 sa putevima iz roditelja 2*

*4. Ostatak čvorova se u decu dodaje na slučajan način*

## 2.4 Mutacija

Korišćena su 2 algoritma za mutaciju:

**Greedy mutacija** Biraju se 2 slučajna grada, i gleda se da li se njihovom zamenom dobija kraći put.

**2opt mutacija**

ALGORITAM: *1. Biraju se 2 para susednih gradova, $(a_1, b_1)$ i $(a_2, b_2)$*

*2. Ako je $d(a_1, b_2) + d(a_2, b_1) < d(a_1, b_1) + d(a_2, b_2)$, ivice $(a_1, b_1)$ i $(a_2, b_2)$ se brišu i dodaju se ivice $(a_1, b_2)$ i $(a_2, b_1)$*

# 3 Rezultati

## 3.1 Tehnički detalji

Program je napisan u programskom jeziku C++. Kao osnova za genetski algoritam, korišćena je biblioteka GALib, verzija 2.4.7, autora Mathew Wall-a sa MIT-a. Za grafički prikaz su korišćene biblioteke SDL i OpenGL.

U planu je izrada korisničkog interfejsa, gde korisnik može da bira između grafičkog i tekstualnog unosa podataka(koordinata gradova). U rar fajlu se nalazi sve sto je potrebno za pokretanje na 64-bitnom Windowsu. Makefile je pravljen takodje za Windows, i potrebno je imati sdl i opengl header-e negde gde kompajler moze da ih nadje.

## 3.2 Način upotrebe programa

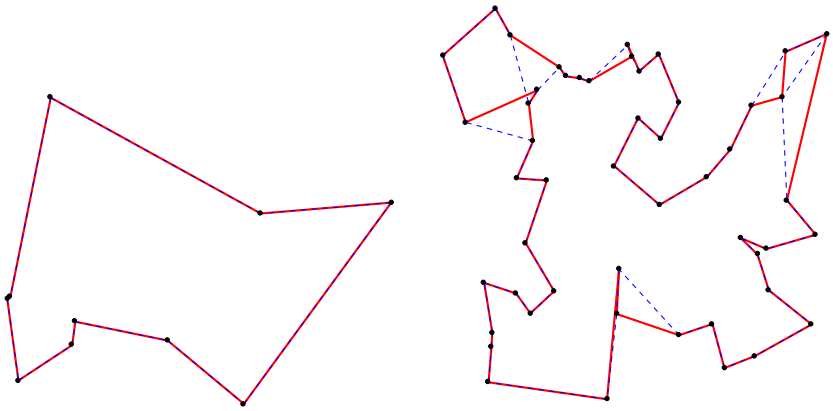
Glavni algoritam se nalazi u programu pod imenom tsp.exe. Program se koristi unošenjem koordinata gradova u fajl, i to u sledećem formatu:

$$\begin{array}{ccc} ID_1 & X_1 & Y_1 \\ ID_2 & X_2 & Y_2 \\ \vdots & \vdots & \vdots \\ ID_n & X_n & Y_n \end{array}$$

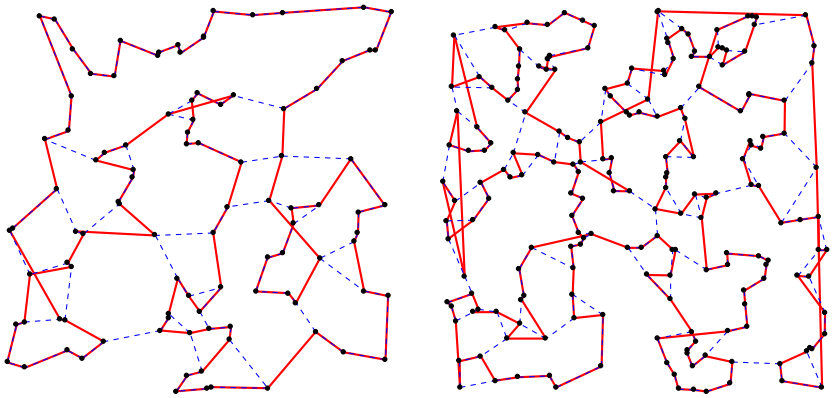
pri čemu su koordinate bilo koji realni brojevi, a $ID_i$ ceo broj. Tako snimljeni fajl se samo prevuče na *tsp.exe*. Tog trenutka će se započeti inicijalizacija, i za par trenutaka će se pojaviti grafički prikaz optimalnog rešenja. Program prestaje sa radom kada populacija potpuno konvergira(za 250 gradova za par sekundi, a za 1000, to je otprilike 2min). U konzoli se vidi dužina najboljeg i najgoreg puta u trenutnoj generaciji, kao i njihov odnos.

## 3.3 Primeri

Na sledeće 4 slike, prikazane su putanje generisane algoritmom datim u ovom radu,(crvena puna linija), kao i putanje generisane programom LKH (isprekdane plave linije) [Lin and Kernighan, 1973]. Slike su dobijene pomoću perl skripte *test.pl* koja je na osnovu tekstualnog izlaza navedenih programa generisala SVG slike, koje su dalje konvertovane u *.png* format programom Inkscape.



Slika 2: Primeri sa 10 i 50 gradova



Slika 3: Primeri sa 100 i 200 gradova

U sledećoj tabeli su prikazani uporedni rezultati (dužine puteva) opisanog algoritma i Lin-Keringhan heuristike iz [Lin and Kernighan, 1973]:

| broj gradova | rezultat GA | rezultat LKH | $\frac{GA}{LKH}$ |
|---|---|---|---|
| 10 | 93420.1 | 93420.087 | 100% |
| 20 | 126105 | 123182.653 | 102.37% |
| 50 | 183528 | 180723.382 | 100.01% |
| 100 | 259950 | 247460.568 | 105.05% |
| 200 | 426510 | 356127.217 | 119.76% |

### 3.4 Zaključak

Na osnovu testiranih primera i algoritama, vidi se da GA za manji broj gradova daje optimalno rešenje, i to u realnom vremenu. Za veći broj gradova (par stotina), algoritam radi ispod 3min i daje rešenja u proseku 20% gora nego Lin Keringhan heuristika. Razlog za to je da se algoritam prekida kada sve jedinke konvergiraju ka najboljoj, odakle je dalji razvoj moguć samo mutacijom.

Prema tome, jedna od mugućnosti za poboljšanje ovog algoritma je inkorporacija drugih heuristika, kao što je LKH. Međutim, opšti utisak autora(nastao tokom izrade i testiranja programa) je da GA možda i nisu najbolja metoda za rešavanje TSP. Za dalji rad verovatno najviše smisla ima okrenuti se celobrojnom programiranju i metodama opisanim u [Applegate et al., 2003].

## Literatura


D. Applegate, R. Bixby, V. ChvÃątal, and W. Cook. Implementing the dantzig-fulkerson-johnson algorithm for large traveling salesman problems. *Mathematical Programming*, 97:91–153, 2003. ISSN 0025-5610.

T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2nd edition, 2001. ISBN 0070131511.

G. Dantzig, R. Fulkerson, and S. Johnson. Solution of a large-scale traveling-salesman problem. *Journal of the operations research society of America*, pages 393–410, 1954.

J. Holland. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.

S. Lin and B. Kernighan. An effective heuristic algorithm for the traveling-salesman problem. *Operations research*, 21(2):498–516, 1973.