Genetski algoritam za višeciljnu optimizaciju

Aleksandra Bošković, Miloš Miković
aleksandra94@hotmail.rs, milos.mikovicpos@gmail.com

14. april 2020.

Sadržaj

1 Uvod

Problem trgovačkog putnika (Traveling Salesman Problem, TSP) je jedan od najpoznatijih problema iz grupe NP - teških problema diskretne i kombinatorne optimizacije. U osnovi ovaj problem je definisan na sledeći način: trgovački putnik tačno zna koje gradove treba da poseti i međusobnu udaljenost između svaka dva grada sa spiska, jedini problem je što je u obavezi da poseti svaki grad samo jednom i vrati se u grad iz kog je krenuo a da pritom pređeni put bude minimalan. Ovako definisan problem, ima samo jednu funkciju cilja i nju treba da minimizuje. Ovom osnovnom problemu dodaćemo jos jedan podatak koji trgovački putnik ima a to je vreme potrebno da se pređe put između svaka dva grada. Cilj je pronaći tačnu putanju kojom trgovački putnik treba da ide tako da i ukupno provedeno vreme na putu i ukupna pređena dužina puta budu minimalni. Ovako definisan problem ima dve funkcije cilja i u našem slučaju, obe pokušavamo da minimizujemo.

Ideja zasnovana na idejama iz literature [3] , [5], [1] i [4]

2 Algoritam

Ulaz: podaci o udaljenosti između svakog od gradova, kao i potrebno vreme za prelazak puta između svakog od gradova
Izlaz: najbolje pronađeno rešenje genetskim algoritmom

Da bismo mogli da poredimo rešenja definišimo prvo funkciju cilja za ovaj problem[3]:

$$min \begin{cases} f1(x) = \sum_{i,j \in x} d_{ij}, \\ f2(x) = \sum_{i,j \in x} t_{ij} \end{cases}$$

pri čemu su d_{ij} i t_{ij} pređeni put i vreme između para gradova i i j, a x predstavlja jednu permutaciju gradova. $f1(x)$ i $f2(x)$ predstavljaju totalni pređeni put i vreme za zadatu rutu x.

Ovako definisan problem, javlja se u literaturi kao početna funkcija cilja, u realni uslovima, ona i nije mnogo primenjiva, stoga ćemo je malo modifikovati. Sa obzirom na to da je često nemoguće u isto vreme minimizovati obe funkcije, a u cilju podjednakog vrednovanja obe veličine definišimo funkciju cilja za genetski algoritam na sledeći način

$$f(x) = alfa * \frac{D_x}{\sum_{i=1}^{m} D_i} + beta * \frac{T_x}{\sum_{i=1}^{m} T_i}$$

pri čemu je m broj jedinki u generaciji, D_x ukupna dužina pređenog puta za permutaciju gradova x, a T_x ukpno vreme potrebno da se pređe odabrani put x. Parametri $alfa$ i $beta$ su regulatorni i default-ne vrednosti su postavljene na 0.5 za oba parametra. Ovi parametri, omogućavaju nam da prioritiziramo pređeni put u odnosu na vreme i obrnuto. Na sličan način formiramo i optimalno rešenje koje dobijamo brute-force algoritmom, stim što su nam u tom slučaju poznate sve moguće rute, i zato egzaktno možemo pronaći najbolje rešenje sa ovako definisanom funkcijom cilja. Formulacija ove funkcije za brute-force algoritam, data je nešto kasnije.

Naš cilj je da za odabrano rešenje x važi:

$$f(x) < f(x'), \forall x' \in \Omega$$

pri čemu je Ω skup svih permutacija skupa gradova kroz koje putnik treba da prođe a x izabrana najbolja permutacija.

2.1 Opis implementacije elemenata genetskog algoritma

Kodiranje jedinke:
Jedinka je predstavljena kao permutacija skupa svih gradova i predstavlja redosled kojim trgovački putnik obilazi gradove.
Inicijalna populacija:
Generisana je random iz skupa svih mogućih permutacija obilaska gradova.
Selekcija:
Turnirska selekcija odabira jedinki za ukrštanje. (fitness-srazmerna)
Mutacija:
Zamena random odabrana dva indeksa niza koji predstavlja jedinku [2]. Pored ove, korišćene su i druge vrste mutacija za kombinatorne probleme, ali one nisu značajno uticale na krajnje rešenje.
Ukrštanje:
Prvi način: Ukrštanje prvog reda za kombinatorne probleme[2]
Drugi način: Random odabran segment jednog roditelja staviti na početak deteta a ostatak niza dopuniti preostalim elementima onim redosledom kojim se javljaju u drugom roditelju a ne sadrži ih odabrani segment prvog roditelja.

Drugi način se pokazao kao bolji jer prvi prebrzo konvergira.
Nova populacija:
Za formiranje nove populacije, korišćena je tehnika elitizma, manje od 10% jedinki se direktno prebacuje u narednu generaciju, a ostatak se dobija ukrštanjem i mutacijama.
Kriterijum zaustavljanja:
Dostignut zadati broj iteracija.

3 Poređenje sa brute-force algoritmom

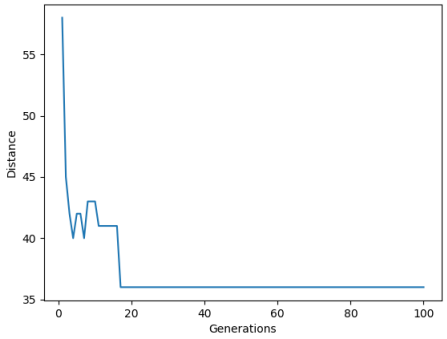
U cilju ocenjivanja dobijenog rešenja ovakvim algoritmom uporedimo sa rešenjem dobijenim brute-force algoritmom, koji ispituje sve moguće permutacije. U algoritmu brute-force pri odabiru najboljeg rešenja tražimo minimum funkcje:

$$f(x) = alfa * \frac{D_x}{\sum_{i=1}^{(n-1)!} D_i} + beta * \frac{T_x}{\sum_{i=1}^{(n-1)!} T_i}$$

pri čemu su korišćene oznake iste kao i kod rešenja za genetski algoritmom, n predstavlja broj gradova koje treba obići. Uočimo razliku da ovde u imeniocu sumiramo vrednosti svih mogućih permutacija jer su nam one poznate, dok nam kod rešenja genetskim algoritmom ova suma zavisi od populacije u svakoj generaciji.

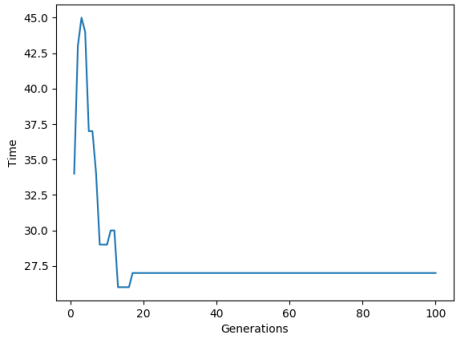
U nastavku na slikama su dati rezultati posmatranog problema za 11 gradova, 100 generacija, verovatnoćom mutacije 0.05, veličinom generacije 20, veličinom turnira 5. Pri čemu najbolje rešenje koje pronalazi algoritam brute force za ovih 11 gradova iznosi $D = 29$ za ukupnu dužinu optimalnog puta i $T = 32$ za vreme potrebno da se pređe optimalan put.

Na slici 1 možemo pratiti na koji način se pređeni put najbolje jedinke u svakoj generaciji menja i približava vrednosti koji je pronašao algoritam brute force za dužinu optimalnog puta.



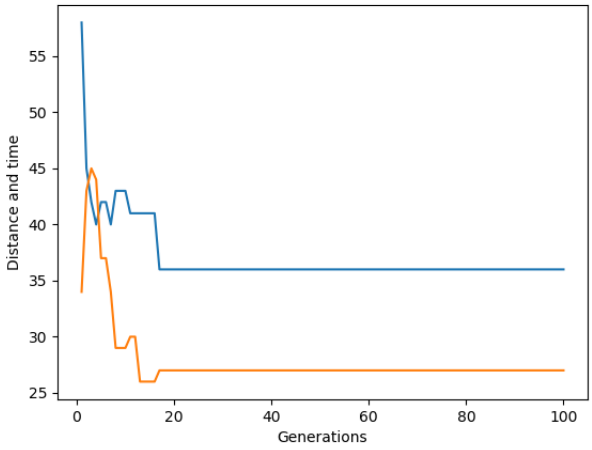
Slika 1: Dužina puta izabranog optimalnog rešenja kroz generacije

Na slici 2 možemo pratiti na koji način se vreme potrebno da se predje put najbolje jedinke u svakoj generaciji menja i približava vrednosti koju je pronašao algoritam brute force.



Slika 2: Vreme potrebno da se pređe put kroz generacije za najbolju jedinku u svakoj generaciji

Slika 3 objedinjuje prethodne slike i prikazuje kako najbolja jedinka napreduje kroz generacije. Na slici se može primetiti da se teži minimizaciji obe funkcije cilja, i da porastom vrednosti jedne, druga opada i obrnuto. Ova pojava, često je i najveći problem višeciljne optimizacije.



Slika 3: Poređenje pređenog puta i vremena za najbolju jedinku kroz 100 generacija

Na sledećim tabelama prikazaćemo na primeru 11(tabela 1) i 10(tabela 2) gradova koliko procentualno predloženi genetski algoritam pronalazi 5 najboljih rešenja dobijenih brute-force algoritmom na 100 iteracija pokretanja algoritma.

Tabela 1: Vrednosti u tabeli važe za posmatran problem 11 gradova, veličinom populacije 20, verovatnoćom mutacije 0.05,veličinom turnira 5 i brojem generacija 3000. Distanca i vreme, dati su za pet najboljih rešenja dobijenih brute-force algoritmom. Kolona "Odabran za optimalno" prikazuje procenat pogotka tog rešenja genetskim algoritmom.

Distanca	Vreme	Odabran za optimalno
29	32	27%
33	29	0%
32	30	46%
36	27	0%
33	30	0%

Tabela 2: Vrednosti u tabeli važe za posmatran problem 10 gradova, veličinom populacije 20, verovatnoćom mutacije 0.05,veličinom turnira 5 i brojem generacija 2500

Distanca	Vreme	Odabran za optimalno
30	29	17%
23	35	18%
26	33	32%
32	28	0%
27	33	0%

Tabela 3: Prosečno vreme izvršavanja na 100 iteracija programa

Broj gradova	Brute Force	Genetski algoritam
10	3.14 sec	1.5070 sec
11	31.3 sec	1.81 sec

Svi testovi(3)rađeni su na računaru sledećih karakteristika:
OS: GNU/Linux
CPU: Intel i7-6498DU 2.50ghz
RAM: 8GB
GPU: Integrisana Intelova kartica

4 Zaključak

Prikazani genetski algoritam na problemu trgovačkog putnika sa dodatim zahtevima za vreme može pronaći globalno optimalno rešenje sa velikom verovatnoćom za 11 i manje gradova. Problem testiranja na više od 11 gradova, predstavljao nam je brute-force algoritam, koji se na 12 gradova izvršavao više od sat vremena, pa je ceo proces analize prekinut. Iako smo prilično sigurni da bi se predloženi genetski algoritam dobro pokazao i na skupu od 12 i više gradova, ovo nismo mogli da potvrdimo iz navedenih razloga. Funkcija cilja ko-

ja je korišćena modeluje linearnu zavisnost između pređenog puta i vremena. Postavlja se pitanje na koji način želimo da modelujemo odnos između vremena i pređenog puta. U zavisnosti od toga, izabraćemo i odgovarajuću funkciju cilja, možda neku koja bolje modeluje optimalno rešenje u realnim uslovima. Stoga, mislimo da na tom planu mogu da se vrše dalja istraživanja. U cilju poboljšanja ovog algoritma moguće je inicijalnu populaciju generisati pažljivim odabirom permutacija, tako da pri konstrukciji permutacije gradove navodimo tako da njihova međusobna udaljenost ili vreme budu što je moguće manji [5]. Ova poboljšanja u vidu inicijalne populacije mogu povećati vreme izvršavanja algoritma, ali će konačna rešenja češće biti bliža optimalnim. Takođe, može se probati sa nekim novim vidom ukrštanja, npr. PMX ukrštanje ili neko drugo koje se dobro pokazalo za kombinatorne probleme. Parametre algoritma kao što su "mutation rate", "population size", "elitism size" i druge, potrebno je detaljno testirati za određene instance problema i sprovesti detaljnu analizu za odabir najboljih. Iako smo pokušavali sa raznim vrednostima ovih parametara tokom testiranja, na ovom planu ima prostora za dalja unapređivanja. Jedna od zanimljivih ideja koje smo videli, jeste podela populacije po kategorijama i određivanje fitness funkcije jedinki na osnovu kategorija u kojima se nalaze. Ovo rešenje zahtevalo bi od nas da pri selekciji, dodatno računamo kvalitet jedinki koje su upale u najbolju kategoriju, kako bi bili sigurni koja je zaista najbolja ili da vratimo bilo koju jedinku od svih koje se nalaze u najboljoj kategoriji, mada bi drugi način doveo do loše konvergencije genetskog algoritma. [5]Ovo rešenje nismo testirali u praksi, iako smo napravili dobar deo koda koji ga demonstrira, iz razloga što smo mislili da bi na ovaj način bespotrebno povećali složenost izvršavanja.

Literatura

[1] RajatKumar Palc Chiranjit Changdara, G.S.Mahapatrab. An efficient genetic algorithm for multi-objective solid travelling salesman problem under fuzziness, 2014. on-line at: https://www.sciencedirect.com/science/article/abs/pii/S2210650213000679?fbclid=IwAR3bJlMb-0O51Up-kRFKaJQ3brZ9Z4cMRAaqPsdNxqmj8XkXrBuVipnF5L0.

[2] Andries P. Engelbrecht. *Computational intelligence, second edition.* 2007.

[3] Ibrahim A. Hameed. Multi-objective Solution of Traveling Salesman Problem with Time, 2020. on-line at: https://www.researchgate.net/publication/331824516_Multi-objective_Solution_of_Traveling_Salesman_Problem_with_Time?fbclid=IwAR2vnsmFLYhjphtIPyrfIbbMHMgbp4Wuc4n3jmF00jPC5raP1VGfYchJtCA.

[4] Thibaut Lust and Jacques Teghem. The Multiobjective Traveling Salesman Problem: A Survey and a New Approach, 2011. on-line at: http://www-desir.lip6.fr/~lustt/Papers/BookTeghem_TL.pdf?fbclid=IwAR2jz4GPNG4UqQuSDrC_BLecSyb4KGOJcHae3dbg7Fl05j2rUIcyHdapDJ0.

[5] Mei Ma and Hecheng Li. A hybrid genetic algorithm for solving bi-objectivetraveling salesman problems, 2017. on-line at: https://iopscience.iop.org/article/10.1088/1742-6596/887/1/012065/pdf?fbclid=IwAR2o9DPMsVBKjzJBgRS_ez-rwYbGRKSGOJZ8eOfRWtsgQkzL4qilV75-CiM.