# Optimizacija kolonijom mrava i njena vizuelizacija u VR

Seminarski rad
Matematički fakultet,
Univerzitet u Beogradu

Peđa Trifunov, Vuk Novaković, Luka Sokolov

5. Februar 2019

# Sadržaj

1. Opis....3
2. Zašto koristimo ACO algoritam?....3
3. Ponašanje ACO algoritma....3
    3.1 Parametri....3
    3.2 Odabir puta....4
    3.3 Funkcionisanje....4
4. Vizuelizacija u VR....5
    4.1 Izgled i orijentacija....5
    4.2 UI....5
    4.3 Arhitektura....6
5. Eksperimentalni rezultati....7
    5.1 Genetski algoritam....8
6. Zaključak....8
7. Literatura....8

# 1. Opis

Algoritam optimizacije kolonijom mrava (ACO - engl. *Ant colony optimization*) je probabilistički metod rešavanja određenih optimizacionih problema, zasnovan na tehnici oponašanja grupe mrava prilikom potrage za resursima u njihovoj okolini.

U ovom radu, problem na koji ćemo primeniti ACO se naziva problemom trgovačkog putnika (TSP – eng. *Traveling Salesman Problem*) u kome je potrebno naći najkraći put u zadatom skupu gradova, tako da nijedan grad neće posetiti dva puta. U ovom radu razmatramo verziju problema u kojoj početni i završni grad nisu isti. Za primer TSP-a smo uzeli glavne gradove Evrope, sa karakteristikom veličine koja se odnosi na broj stanovnika u njima.

ACO algoritam ćemo primeniti na naš problem koji će dati lokalni optimum za mogući put između odabranih gradova.

# 2. Zašto koristimo ACO algoritam?

TSP je NP težak problem, što znači da se za njega ne može naći rešenje u polinomijalnom vremenu. Zato TSP možemo predstaviti kao optimizacioni problem, a ACO kao optimizaciona tehnika, se može lako primeniti na njega. Takođe, lako je shvatljiv, što znači da se prilikom njegove implementacije nećemo susresti sa mnogo poteškoća.

# 3. Ponašanje ACO algoritma

U prirodi, mravi u cilju nalaženja optimalnog puta koriste hemijsku izlučevinu, feromon, signal koji mravi ispuštaju za sobom u cilju da ga ostali mravi ‘pokupe’, i prate do ciljnog odredišta. Feromon isparava tokom vremena, i mrav sa većom verovatnoćom prati put na kome je feromon manje ispario, jer je taj put potencijalno najkraći. Tada će se sve više i više mrava kretati tim putem, ispuštati svoje feromone i na taj način pojačavati rutu kojom se kreću. Vremenom to će postati najpoželjniji put kojim će se oni kretati i svaki mrav će njega koristiti između dva odredišta.

## 3.1 Parametri

Glavnu ulogu u algoritmu imaju mravi. Oni se ponašaju kao putnici koji se kreću od grada do grada i na taj način prave svoju rutu po Evropi. Ponašanje kolonije se reguliše parametrima:

- *a* – reguliše ulogu feromona u biranju sledećeg grada

- *b* – reguliše ulogu distance u biranju sledećeg grada
- *c* – faktor isparenja koji reguliše isparenje feromona
- broj mrava u koloniji
- broj iteracija kolonije
- količina feromona koja se ostavlja na svakom putu.

## 3.2 Odabir puta

Mrav *k* koji se nalazi u gradu *x* sledeći grad bira na osnovu formule:

$$p_{xy}^{k}=\frac{(t_{xy}^{a})*(n_{xy}^{b})}{\sum_{z\in neposećeni_{x}}(t_{xz}^{a})*(n_{xz}^{b})},\forall\, y\in neposećeni_{x}$$

gde je:

- $p_{xy}^{k}$ verovatnoća da mrav pređe u grad y
- $t_{xy}^{a}$ faktor feromona između dva grada
- $n_{xy}^{b}$ faktor dužine puta, gde je: $n=\frac{1}{d_{xy}}$ .

Računa se verovatnoća svakog prelaska, vrši se ruletska selekcija za konačni odabir sledećeg grada i postupak se ponavlja sve dok ima neposećenih gradova.

Nakon što svaki mrav završi svoju rutu, količina feromona se ažurira. Prvi korak u ažuriranju je isparenje:

$$t_{xy}=t_{xy}*(1-c)$$

gde je c faktor isparenja.

Drugi korak je dodavanje feromona. Svaki mrav na put kojim je prošao postavi određenu (konstatnu) količinu feromona. Zatim se ceo postupak ponavlja dok se ne ispuni zadati broj iteracija. Kada se to završi, biće odabrani svi putevi tako da ukupna dužina između puteva ne odstupa u velikoj meri od najkraće.

## 3.3 Funkcionisanje

Algoritam je implemetiran u jeziku C#, radi lakše kompatibilnosti sa vizuelizacijom kroz Unity platformu. Graf je predstvaljen matricom udaljenosti. Obzriom da je graf potpun, data matrica je

simetrična. S tim u vidu, korišćena je donje trougaona komponenta te matrice radi uštede vremena i prostora.

Za svaku iteraciju mravi se puštaju po mapi. Kada završe obilazak, svaki mrav ima rutu koja je okarakterisana kao njegov najkraći put. U skladu sa tim, azurira se matrica feromona na gore opisan način.

# 4. Vizuelizacija u VR

Za vizuelizaciju algoritma je korišćena platforma Unity, a jezik C#. Postoji više opcija za korišćenje algoritma, u cilju prilagođenosti. Sledeće poglavlje daće uvid u opcije aplikacije.

## 4.1 Izgled i orijentacija

Korisnik se može kroz aplikaciju kretati pogledima glave sa stanovišta više kamera, koje su postavljene na stubovima i u određenim gradovima, radi lakšeg pregleda. Postoji 5 stubova, po jedan sa svake strane sveta i jedan izdignut koji daje pogled na mapu iz visine. Takođe se nalazi i u šest gradova. Odabir kamere se vrši na tzv. *Gaze-look* način, tako što se korisnik zagleda željeni stub i nakon nekoliko sekundi prebaciće se u željenu kameru.


*Slika 1. Pogled odozgo*

## 4.2 UI

Korisnik može interfejsu pristupiti odabirom opcije *Toggle UI* i tu odabrati određene opcije. U početku, sve vrednosti su postavljene na optimalne:

1- *Town options:* mogućnost odabira gradova. Mogu se odabrati svi gradovi (*All towns*), samo veliki gradovi (*Large towns*), gradovi srednje veličine (*Medium towns*), i mali gradovi (*Small towns*). Takođe postoji opcija da aplikacija sama uzme nasumičan broj i nasumične gradove (*Random towns*).

2- *Path options:* mogućnost da korisnik bira boju kojom će potencijalni putevi kojima se mravi kreću biti obojeni (crvena, plava i zelena boja). Finalni, odabrani putevi su inicijalno bele boje i ne mogu se menjati.

3- *Ant options:* mogućnost odabira količine mrava koji će se koristiti u algoritmu kao i brzina kojom se mravi kreću.

4- *Pheromone options:* mogućnost odabira koeficijenta evaporacije i količine ispuštanja feromona.

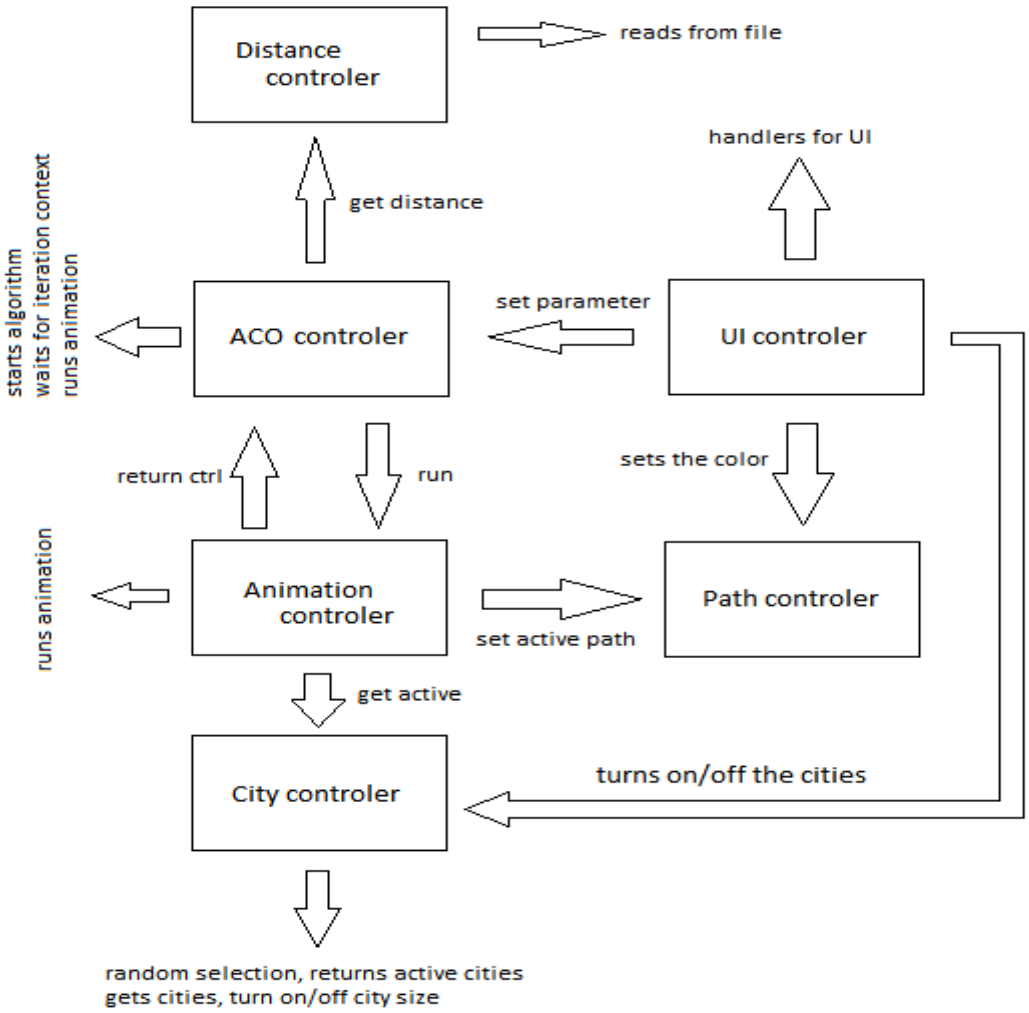
5- *Miscellaneous options:* mogućnost odabira broja iteracija, kao i parametara $\alpha$ i $\beta$.

6- *Start* dugme koje pokreće aplikaciju i *Stop* koje zaustavlja.


*Slika 2. User interface*

## 4.3 Arhitektura

Arhitektura je predstavljena sledećom slikom, gde pravougaonici predstavljaju klase a strelice između njih, interakcije. Izlazne strelice su dodatne operacije klase:




*Slika 3. Arhitektura aplikacije*

# 5. Eksperimentalni rezultati

Kao ideja za implementaciju korišćen je kod [1]. Početni rezultati sa parametrima kao iz idejnog koda, nisu bili dovoljno dobri, jer je u određenim slučajevima najkraći put imao u sebi granu koja se prostirala preko čitave Evrope (slika 4.).


*Slika 4. put koji se dobija sa* β = 0.8


*Slika 5. put koji se dobija sa* β = 8.8

Prilagođavanjem parametra β, dobio se kraći put, kao što je prikazano (slika 5).

Za poređenje korišćen je kod [2], koji je, iako sam po sebi ne nalazi optimalan rezultat, za veći broj gradova, nalazio bolje puteve od ACO algoritma. Za celu Evropu, navedeni kod je našao najkraći mogući put dužine 32023.55 km, dok ACO obično nalazi put dužine 34000km.

## 5.1 Genetski algoritam

U cilju nalaženja što boljih parametara, razvili smo genetski algoritam koji nam je to i omogućio. Iskoristili smo rezultate koje nam je dao, i njih podesili kao podrazumevane početne vrednosti u algoritmu. Za fitnes funkciju genetskog algoritma je korišćena dužina puta, ukoliko je vreme izvršavanja bilo u zadatom vremenskom intervalu (5 s). Ukoliko nije, za fitnes funkciju se uzima zbir dužine puta i vreme prekoračenja u ms. Ova fintes funkcija je izabrana da ne bi došlo do predugačkog izvršavanja (da treniranje ne bi predugo trajalo). Pošto algoritam daje različite rezultate za iste parametre (nije deterministički), radi stabilnijih trening rezultata, ACO algoritam je puštan tri puta pre računanja fitnes funkcije. Zatim je fitnes funkcija računata sa prosečnim vrednostima za ta tri puštanja.

Nakon dodatnih podešavanja, pored parametara iz genetskog algoritma povećali smo broj iteracija i broj mrava u koloniji, koji su dodatno poboljšali rezultate.

# 6. Zaključak

ACO algoritam nije deterministički i neće za iste parametre uvek dati isto rešenje, ali se dužine pronađenih puteva neće mnogo razlikovati. Ovo je takođe uticalo na rešenja genetskog algoritma, tj. nije mogao da se iskoristi njegov pun potencijal.

Takođe, algoritam će se bolje ponašati za veći broj iteracija, kao i za veći broj mrava, za $\alpha < \beta$, za srednje količine feromona, i srednju količinu koeficijenta isparenja.

Za velike grafove, ACO će teško naći optimalan, ili makar blizu optimalan put.

# 7. Literatura

[1] https://github.com/pjmattingly/ant-colony-optimization

[2] https://github.com/dmishin/tsp-solver

[3] ACO Algorithms for the Traveling Salesman Problem,

Thomas Stutzle, Marco Dorigo

[4] An ant colony optimization method for generalized TSP problem,

Jinhui Yang, Xiaohu Shi, Maurizio Marchese, Yanchun Liang. 2008