Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 51
DOI: https://doi.org/10.24867/32JV01Josic

OPTIMIZACIJA KOLONIJOM MRAVA

ANT COLONY OPTIMIZATION

Nikolina Jošić, *Fakultet tehničkih nauka, Novi Sad*

Oblast - MATEMATIKA U TEHNICI

Kratak sadržaj – *U ovom radu istražen je algoritam optimizacije kolonijom mrava (Ant Colony Optimization - ACO), koji se koristi za rješavanje kompleksnih optimizacijskih problema. Poseban fokus stavljen je na problem trgovačkog putnika (Traveling Salesman Problem - TSP), gde je potrebno pronaći najkraću moguću putanju koja prolazi kroz zadati skup gradova. Praktični dio rada je urađen u programskom jeziku Python. U radu je implementiran ACO algoritam, eksperimentisano je sa različitim parametrima, i analizirane su performanse algoritma. Rezultati su pokazali efikasnost i adaptivnost ACO algoritma, kao i uticaj različitih parametara na njegovu performansu.*

Ključne reči: *Optimizacija kolonijom mrava (ACO), Problem trgovačkog putnika (TSP), Optimizacijski algoritmi, Python, Performanse algoritma, Analiza parametara*

Abstract – *In this paper, the Ant Colony Optimization (ACO) algorithm, used for solving complex optimization problems, is investigated. Special focus is placed on the Traveling Salesman Problem (TSP), where the objective is to find the shortest possible route that passes through a given set of cities. The practical part of the study was conducted in the Python programming language. The ACO algorithm was implemented, various parameters were experimented with, and the algorithm's performance was analyzed. The results demonstrated the efficiency and adaptability of the ACO algorithm, as well as the impact of different parameters on its performance.*

Keywords: *Ant Colony Optimization (ACO), Traveling Salesman Problem (TSP), Optimization Algorithms Python, Algorithm Performance Parameter Analysis*

1. UVOD

Problemi optimizacije su veoma važni u oblasti nauke i industrije. Neki primjeri problema optimizacije iz stvarnog života su pravljenje rasporeda, pravljenje reda vožnje, planiranje kapaciteta, problem trgovačkog putnika, problem rasporeda grupne radnje, itd. Mnogi algoritmi optimizacije su razvijeni iz ovog razloga.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Nebojša Ralević, red. prof.

Optimizacija kolonijom mrava je jedan od njih. Optimizacija kolonijom mrava je probabilistička tehnika za pronalaženje optimalnih putanja.Algoritam optimizacije kolonijom mrava se koristi za rješavanje različitih računarskih problema.

Ovaj algoritam je uveden na osnovu ponašanja mrava u potrazi za hranom za traženje puta između svoje kolonije i izvorne hrane. U početku se koristio za rješavanje dobro poznatog problema trgovačkog putnika. Kasnije se koristi za rješavanje različitih teških problema optimizacije. U ovom radu detaljno je razmotrena teorijska osnova ACO algoritma, kao i njegova primjena u različitim oblastima sa posebnim fokusom na problem trgovačkog putnika TSP.

U ovom problemu dati su skup lokacija i udaljenosti između njih. Problem se sastoji u nalaženju zatvorene šetnje minimalne dužine koja prolazi kroz svaku lokaciju tačno jednom (osim one prve u koju se i vraća).

2. OPTIMIZACIJA KOLONIJOM MRAVA

Optimizacija kolonijom mrava (Ant Colony Optimization -ACO) je metaheuristički algoritam inspirisan ponašanjem stvarnih mrava u prirodi, posebno načinom na koji oni pronalaze najkraći put između kolonije i izvora hrane. Ovaj algoritam je prvi put predložio Marko Dorigo 1990-ih godina, i od tada je postao jedan od najpoznatijih pristupa u rješavanju složenih problema optimizacije.

Mravi koriste hemijski trag poznat kao feromon kako bi komunicirali međusobno. Kada mrav pronađe izvor hrane, na putu nazad do kolonije ostavlja trag feromona. Ostali mravi imaju veću vjerovatnoću da prate puteve sa većom koncentracijom feromona, što dovodi do formiranja sve efikasnijih puteva kako se više mrava kreće tom rutom. Ovaj princip je osnova ACO algoritma: mravi virtuelno ostavljaju tragove feromona na putevima između različitih tačaka problema koje treba optimizovati, i postepeno pronalaze optimalno rešenje kroz iterativnu optimizaciju.

ACO algoritam za problem trgovačkog putnika

Problem trgovačkog putnika (TSP) je jedan od najčešćih problema optimizacije koji ACO algoritam rešava. Cilj je pronaći najkraći put koji polazi iz jednog grada, prolazi kroz sve druge tačno jednom, i vraća se u početni grad. ACO koristi veštačke mrave koji pretražuju prostor mogućih rješenja, imitirajući ponašanje stvarnih mrava.

Vjerovatnoća da mrav pređe sa jednog čvora *i* u drugi čvor *j* je definisana formulom:

$$P_{ij} = \frac{[\tau_{ij}(t)]^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\sum_{k \epsilon J}[\tau_{ik}(t)]^{\alpha} \cdot [\eta_{ik}]^{\beta}}$$

gdje su:

- $\tau_{ij}(t)$ - količina feromona na putu između čvorova *i* i *j* u trenutku t,
- η_{ij} – heuristička vrijednost , obično definisana kao recipročna vrijednost udaljenosti između *i* i *j* ($\eta_{ij} = 1/d_{ij}$),
- α i β su parametri koji kontrolišu relativan značaj feromona i heurističke informacije,
- J je skup neposjećenih čvorova.

Ažuriranje feromona

Nakon svake iteracije, tragovi feromona se ažuriraju kako bi odražavali kvalitet pronađenih rješenja. Proces isparavanja feromona je ključan za sprječavanje preuranjene konvergencije na suboptimalna rješenja. Ažuriranje feromona za svaki put (i,j) odvija se po formuli:

$$\tau_{ij}(t+1) = (1-\rho) \cdot \tau_{ij}(t) + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}$$

gdje je:

- ρ – faktor isparavanja feromona (0 <ρ<1),
- $\Delta\tau_{ij}^{k}$ – količina feromona koju ostavlja k-ti mrav.

Količina feromona koju jedan mrav ostavlja proporcionalna je kvalitetu njegovog rješenja (obrnuto proporcionalna dužini puta koji je prešao):

$$\Delta\tau_{ij}^{k} = \frac{Q}{L_k}$$

gdje je:

- Q – konstanta.
- L_k – dužina puta kojeg je prešao *k*- ti mrav.

3. Praktična primjena

Proces pretrage

Na početku pretrage, svi mravi su postavljeni nasumično na različite čvorove grafa koji predstavlja problem. Svaki mrav pretražuje prostor mogućih rješenja koristeći pravilo proporcionalno količini feromona i heurističke vrijednosti, birajući sledeći čvor na osnovu vjerovatnoće P_{ij}. Kako iteracije napreduju, putevi sa većom koncentracijom feromona postaju popularniji, što vodi ka globalnom optimizovanom rješenju.

ACO algoritam je fleksibilan i lako se može prilagoditi različitim problemima optimizacije, uključujući probleme rasporeda, rutiranja i druge kombinatorne probleme.

Optimizacija kolonijom mrava (ACO) implementirana je u Python programskom jeziku zbog njegove jednostavnosti, bogatog seta biblioteka i čitljivog koda. Python omogućava brzu prototipizaciju i vizualizaciju složenih algoritama, što je idealno za testiranje i analizu performansi ACO algoritma.

Implementacija ACO algoritma uključuje nekoliko ključnih koraka:

1. Inicijalizacija kolonije mrava: Svaki mrav predstavlja potencijalno rešenje problema, npr. za TSP problem, mrav predstavlja rutu između gradova.

2. Izbor sledećeg čvora: Koristi se pravilo proporcionalno feromonima i heurističkim vrednostima kako bi mravi birali sledeći čvor.

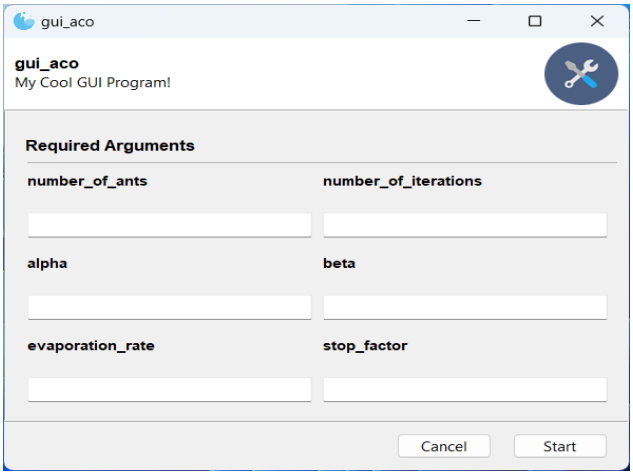
3. Ažuriranje feromona: Nakon što svi mravi završe iteraciju, tragovi feromona se ažuriraju, uzimajući u obzir dužinu pređenih ruta.

4. Kriterijum zaustavljanja.

3.1. Grafički korisnički interfejs (GUI)

Jedan od ključnih elemenata praktične primjene ACO algoritma je razvoj grafičkog korisničkog interfejsa (GUI). GUI omogućava korisnicima da vizualno prate proces optimizacije i podešavaju parametre kao što su broj iteracija, broj mrava, stopa isparavanja feromona i početne vrijednosti feromona. Za razvoj GUI-ja u Pythonu korištena je biblioteka Tkinter, koja omogućava jednostavno kreiranje prozora, dugmadi, polja za unos i grafikona.

Na sledećoj slici je prikaz jednostavnog GUI-ja koji omogućava unos parametara ACO algoritma i pokretanje vizualizacije:

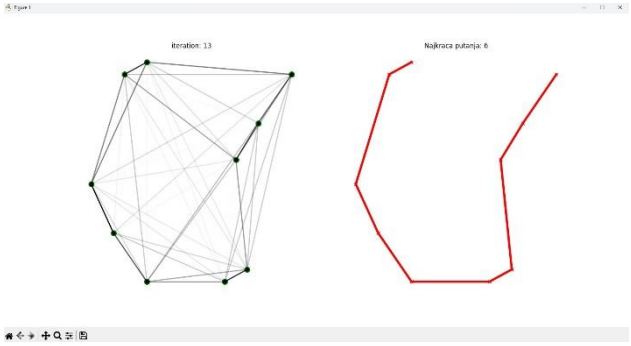


Slika 1. *Grafički korisnički interfejs GUI*

Korisnici mogu lako prilagoditi parametre i pratiti kako algoritam prolazi kroz iteracije, tražeći optimalno rješenje problema. Vizualizacija pokazuje kako se rute mjenjanju i konvergiraju ka optimalnim tokom vremena, što je posebno korisno za analizu performansi algoritma.

3.2. Vizualizacija rezultata

Biblioteka Matplotlib koristi se za vizualizaciju pređenih ruta i feromonskih tragova u svakoj iteraciji. Na slici 2. je prikazan primer rute koju su mravi pronašli nakon nekoliko iteracija. Crne tačke predstavljaju gradove, a crte između njih predstavljaju rute koje mravi koriste. Debljina linije je proporcionalna količini feromona deponovanih na toj ruti, što omogućava vizuelno praćenje koji putevi su češće korišćeni.



Slika 2. *Prikaz iteracije*

3.3. Mjerenja

U ovoj podsekciji testiramo kako hiperparametri utiču na efikasnost agoritma. Parametri na raspolaganju su: n (broj mrava), m (broj gradova), α (važnost traga), β (važnost vidljivosti) i ρ (faktor isparavanja), kao i stop faktor.

Tabela 1. *Mjerenja sa inicijalnim paramtrima programa sa random tačkama*

n	m	α	β	ρ	Najkraća putanja	Broj iteracija	Stop faktor
10	10	1	1	0.6	12	99	0
10	10	1	1	0.6	17	99	0
10	10	1	1	0.6	16	99	0
10	10	1	1	0.6	10	21	2
10	10	1	1	0.6	6	13	2
10	10	1	1	0.6	9	91	10

Tabela 2. *Mjerenja sa inicijalnim paramtrima programa sa fiksnim tačkama*

n	m	α	β	ρ	Najkraća putanja	Broj iteracija	Stop faktor
10	10	1	1	0.6	14	99	0
10	10	1	1	0.6	7	99	0
10	10	1	1	0.6	7	99	0
10	10	1	1	0.6	5	11	2
10	10	1	1	0.6	7	15	2
10	10	1	1	0.6	8	81	10

U prve dijve tabele testiramo funkciju stop faktora. Ostali parametri su fiksirani, s tim da su u prvoj tabeli gradovi random generisani, a u drugoj fiksirani.

Ako izostavimo stop faktor (odnosno postavimo ga na 0), onda algoritam vrshi 100 iteracija (unaprijed zadato). Ako je stop faktor 2, onda broj trenutne iteracije treba da bude duplo veći od broja najbolje iteracije kako bi se zaustavila pretraga.

Posmatrajući ove dvije tabele vidimo da je najkraća putanja pronadjena već do 20. iteracije, tako da nam je broj koraka 100 kao parametar zaustavljanja malo prevelik, to jest, bespotrebno se troše resursi, dakle bolje je staviti npr. da je stop faktor 2.

U narednim tabelama ostavićemo istih 10 fiksiranih tačaka, stavićemo da je stop faktor 2 i menjati ostale parametre.

U sledeće dve tabele mjenjamo α i β. Naime, u prethodnim tabelama su parametri α i β bili isti. U prvoj narednoj tabeli je β veća od α a drugoj α veća od β.

Tabela 3. *Mjerenja uz veću važnost vidljivosti*

n	m	α	β	ρ	Najkraća putanja	Broj iteracija	Stop faktor
10	10	2	3	0.6	6	13	2
10	10	2	3	0.6	3	7	2
10	10	2	3	0.6	7	15	2
10	10	3	5	0.6	4	9	2
10	10	3	5	0.6	4	9	2
10	10	3	5	0.6	3	7	2
10	10	1	2	0.6	7	15	2
10	10	1	2	0.6	3	7	2
10	10	1	2	0.6	4	9	2
10	10	1	5	0.6	6	13	2
10	10	1	5	0.6	2	5	2
10	10	1	5	0.6	2	5	2

Primjećujemo drastično poboljšanje uspjeha algoritma ukoliko mu povećamo zavisnost o vidljivosti (udaljenosti gradova). Naime, vidimo da je najkraća putanja najbrže nadjena kada je β mnogo veća od α.

Tabela 4. *Mjerenja uz veću važnost traga*

n	m	α	β	ρ	Najkraća putanja	Broj iteracija	Stop faktor
10	10	3	1	0.6	4	9	2
10	10	3	1	0.6	6	13	2
10	10	5	1	0.6	8	17	2
10	10	5	1	0.6	6	13	2
10	10	1	0	0.6	7	15	2
10	10	1	0	0.6	8	17	2
10	10	1	0	0.6	5	11	2
10	10	2	0	0.6	7	15	2
10	10	2	0	0.6	7	15	2
10	10	2	0	0.6	10	21	2

Naglaskom na trag mravi gube važnost najbližeg grada. U eksperimentima bez osjetljivosti na udaljenost (β=0) algoritam dolazi do najkraće putanje nakon mnogo koraka.

Pokušaj da se malo poveća uloga traga u algoritmu bez mijenjanja koeficijenta α i β može se postići povećanjem faktora isparavanja. Smanjenjem ρ bi trebalo da poraste uticaj vidljivosti u odnosu na trag. Iz rezultata eksperimenata iz tabele 1. vidljivo je da faktor isparavanja ne utiče na rezultat tako jako kao sami α i β.

U tabeli 6 provjerićemo da li broj mrava utiče na efikasnost algoritma za naših 10 fiksiranih tačaka.

Tabela 5. *Mjerenja važnosti faktora isparavanja*

n	m	α	β	ρ	Najkraća putanja	Broj iteracija	Stop faktor
10	10	1	1	0.4	7	15	2
10	10	1	1	0.4	8	17	2
10	10	1	1	0.4	7	15	2
10	10	1	1	0.5	8	17	2

10	10	1	1	0.5	6	13	2
10	10	1	1	0.5	4	9	2
10	10	1	1	0.7	9	19	2
10	10	1	1	0.7	7	15	2
10	10	1	1	0.7	4	9	2
10	10	1	1	0.8	7	15	2
10	10	1	1	0.8	4	9	2
10	10	1	1	0.8	7	15	2

Međutim, eksperimentalni rezultati takođe ukazuju na neke izazove i ograničenja ACO algoritma. Na primjer, performanse algoritma mogu značajno varirati u zavisnosti od inicijalnih parametara, što zahtjeva pažljivo podešavanje kako bi se postigli najbolji rezultati. Takođe,

Tabela 6. *Mjenjanje broja mrava*

n	m	α	β	ρ	Najkraća putanja	Broj iteracija	Stop faktor
2	10	1	1	0.6	9	19	2
2	10	1	1	0.6	5	10	2
6	10	1	1	0.6	10	21	2
6	10	1	1	0.6	5	11	2
10	10	1	1	0.6	7	15	2
10	10	1	1	0.6	1	3	2
10	10	1	1	0.6	5	11	2
14	10	1	1	0.6	12	25	2
14	10	1	1	0.6	8	17	2
18	10	1	1	0.6	13	27	2
18	10	1	1	0.6	10	21	2
20	10	1	1	0.6	14	29	2
20	10	1	1	0.6	9	19	2

Vidimo da se povećanjem broja mrava efikasnost algoritma ne povećava, kao i da se u slučaju jednakog broja mrava i gradova algoritam zaustavio nakon samo 3 iteracije. Zaključujemo da nema smisla uzimati veći broj mrava, nego je bolje uzeti isti ili manji broj mrava od gradova.

6. ZAKLJUČAK

Evaluacija rezultata implementacije pokazala je da ACO algoritam može efikasno da rješava problem trgovačkog putnika, često nalazeći optimalne ili blizu optimalne rute. Jedna od prednosti ovog algoritma je njegova sposobnost da se prilagodi različitim konfiguracijama problema, uz relativno brz proces konvergencije ka optimalnom rješenju.

7. LITERATURA

[1] Dorigo, M., Optimization, Learning and Natural Algorithms (in Italian). PhD thesis, Dipartimento di Elettronica, Politecnico di Milano, Milan, Italy, 1992.

[2] Dorigo, M., and Gambardella, L.M., Ant Colony System: A Cooperative Learning Approach to the Travelling Salesman Problem, IEEE Transactions on Evolutionary Computation, 1, 53-66, (1997).

[3] Dorigo, M., and Gambardella, L.M., Ant Colony System: A Cooperative Learning Approach to the Travelling Salesman Problem, IEEE Transactions on Evolutionary Computation, 1, 53-66, (1997).

[4] Lučić, P., and Teodorović, D., Transportation Modeling: An Artificial Life Approach, Proceedings of the 14th IEEE "International Conference on Tools with Artificial Intelligence", pp. 216-223, November 4-6, 2002 Washington D.C.

[5] Teodorović, D., Šelmić, M., Računarska inteligencija u saobraćaju, Beograd, 2019.

vrijeme izvršavanja algoritma može postati problematično za vrlo velike ili kompleksne instance TSP-a, gde broj mogućih rješenja eksponencijalno raste sa brojem gradova. Medjutim, u ovom konkretnom radu uveli smo stop faktor, koji omogućava da se algoritam zaustavi u relativno kratkom vremenu.

Ovaj rad je demonstrirao da ACO algoritam, iako već dobro poznat u teoriji optimizacije, ima ogroman potencijal za praktične primjene, posebno kada se kombinuje sa savremenim tehnologijama kao što su Python i grafički korisnički interfejsi. Iako postoje izazovi u vezi sa performansama i skalabilnošću, ovi problemi su rješivi kroz dalja istraživanja i unapređenja algoritma.

Kratka biografija:

Nikolina Jošić rodjena je 14. juna 1985. godine u Trebinju. Završila je gimnaziju "Jovan Dučić" 2004. godine. Diplomirala je 2013. godine na Elektrotehničkom fakultetu u Banjaluci, smjer Elektronika i telekomunikacije. U oktobru 2017. godine upisuje master studije primjenjene matematike na Fakultetu tehničkih nauka u Novom Sadu.