Tab 1

  
  
Predmet: Vještačka inteligencija

Tema: Ručna implementacija ACO algoritma

Školska godina: 2024/2025

Studenti:

* Bakir Vreto (indeks: 19140 )
* Faruk Baković (indeks: 19186)
* Muhamed Selmanović (indeks: 19115)

[**Faza 1: Izbor teme i opis problema 3**](#_7z0q3ukc3zfa)

[1.1. Opis problema 3](#_44ugtlf4jroc)

[1.2. Definicije i primjena 4](#_bwzo6r9hc7e7)

[1.3. Organizacija dokumenta 4](#_pw57odsxqufa)

[**Faza 2: Pregled stanja u oblasti 5**](#_qlx2pwcdhjvw)

[2.1. Temeljni radovi 5](#_pyqejxnufucx)

[2.2. Primjene i varijante 5](#_pu26vjmapaqf)

[2.3. Postignuti rezultati 5](#_yl2rv4mcsyzy)

[2.4. Potencijalni pravci poboljšanja 5](#_96jimaq45lei)

[**Faza 3: Izbor, analiza i pretprocesiranje dataset-a 6**](#_auv81m1hd4cd)

[3.1. Izbor dataset-a 6](#_5j6le4pwzhap)

[3.2. Analiza dataset-a 6](#_133udijt0bmb)

[3.3. Pretprocesiranje 6](#_kgc8nyhnsk81)

[3.4. Rizici 7](#_6gpx30d4623f)

[**Faza 4: Ručna implementacija, treniranje i testiranje modela 7**](#_1735q75ngfqr)

[4.1. Odabir metode i tehnologija 7](#_dkzapd7a5cac)

[4.2. Priprema podataka 7](#_ai37ssj1o4yo)

[4.3. Implementacija algoritma 8](#_n7fhjce4yjeh)

[Struktura kolonije: 8](#_np7wzn855ti8)

[Kretanje mrava: 8](#_sorsrswxh6n1)

[Feromonsko ažuriranje: 9](#_2p5o6dmbet4u)

[4.4. Parametri i treniranje 9](#_9ip6f321irdm)

[4.5. Metodologija testiranja 10](#_ybg3lehk0w7a)

[4.6. Korištene metrike 10](#_ao99tykote6u)

[4.7. Eksperimenti s nepoznatim podacima 10](#_zawzjvphm1ec)

[4.8. Ponašanje kolonije i empirijske karakteristike 10](#_z5cadfx9cpl)

[Glavna logika algoritma 11](#_8802e9nlsriu)

[Prekid rada – opcije: 12](#_fku827lsthlj)

[Rješavanje TSP i heuristički eksperimenti 12](#_6b9msox1i9qm)

[Rješavanje s "divljim mravima" 13](#_73d7ysz6vnij)

[Statistička slučajna varijacija 13](#_b2o9h4hk2cyy)

[Performanse i parametri 13](#_3iy7wd3m7kri)

[Podešavanje distance\_power 14](#_bv0aabnbfcra)

[Podešavanje pheromone\_power 14](#_ne2oxao8xihg)

[Podešavanje reward\_power 14](#_3swbqhqc4muv)

[**Faza 5: Cjelokupni osvrt na problem i dobijeno rješenje 16**](#_p8pgwhchuumd)

[5.1. Rezultati 16](#_k21b2cvc97uy)

[5.2. Usporedba s radovima iz literature 16](#_ac979c7ac78a)

[5.3. Moguća poboljšanja 16](#_spf9n7untdk7)

[5.3. Zaključak 16](#_2w5od0r7cs02)

# **Faza 1: Izbor teme i opis problema**

## 1.1. Opis problema

Glavni cilj projekta je ručno ("from scratch") implementirati i eksperimentisati s ACO algoritmom (Ant Colony Optimization) za rješavanje problema Travelling Salesman Problem (TSP). Ovaj pristup omogućava razumijevanje temeljnih principa metaheuristika, načina širenja feromona i balansiranja eksploracije i eksploatacije.  
  
U ovom projektu, korišten je specifično odabran dataset koji se sastoji od 30 gradova u Sjedinjenim Američkim Državama. Ovaj dataset omogućava testiranje i evaluaciju implementiranog ACO algoritma u realističnom geografskom scenariju, gdje se koordinatama gradova definiše problem trgovačkog putnika.  
  
Optimizacija kolonije mrava (ACO) je metaheuristički algoritam inspirisan ponašanjem stvarnih kolonija mrava prilikom pronalaženja najkraćih puteva do hrane. Primjenom ovog pristupa u rješavanju problema trgovačkog putnika (Travelling Salesman Problem - TSP), cilj je pronaći najkraći mogući put koji povezuje sve gradove tačno jednom i vraća se u početni grad. Ovaj dokument predstavlja detaljnu razradu implementacije ACO algoritma, analize podataka, rezultata testiranja i mogućih poboljšanja.

Problemi optimizacije poput TSP-a često se javljaju u realnim životnim situacijama, kao što su dostavne rute, planiranje putovanja, logistika i mnoge druge aplikacije. Efikasno rješavanje ovih problema može značajno smanjiti operativne troškove i vrijeme.

Dosadašnja istraživanja su pokazala kako različite varijante ACO algoritma mogu značajno smanjiti vremensku kompleksnost u pronalaženju približno optimalnih rješenja za TSP. Međutim, i dalje postoje izazovi u odabiru optimalnih parametara, koji direktno utiču na performanse algoritma.

Detaljnom analizom dataset-a prilagođeni skup od 30 američkih gradova primijećeno je da se radi o standardizovanom skupu podataka koji omogućava jednostavnu evaluaciju performansi algoritama za optimizaciju. Korištenjem ovakvih standardnih skupova omogućeno je direktno poređenje rezultata.

Ručna implementacija algoritma omogućava potpunu kontrolu nad svakim korakom procesa te omogućava lakše testiranje različitih varijacija algoritma, kao i eksperimentisanje s promjenom parametara u realnom vremenu.

## 1.2. Definicije i primjena

* *Metaheuristika*: generalizirani pristup optimizaciji koji traži dobra rješenja kroz iterativne procese.
* *Feromon*: virtualna supstanca kojom kolonija agenata obilježava dobre puteve.
* *Eksploracija vs. eksploatacija*: pronalaženje novih rješenja naspram poboljšanja postojećih.

Rješavanje TSP s AOC algoritmom može se primijeniti u logistici (optimizacija ruta dostave), robotskim sustavima i planiranju proizvodnje.

## **1.3. Organizacija dokumenta**

Dokument je podijeljen u pet faza, prema planu:

* Faza 1: Izbor teme i opis problema
* Faza 2: Pregled stanja u oblasti
* Faza 3: Izbor, analiza i pretprocesiranje dataset-a
* Faza 4: Ručna implementacija, treniranje i testiranje
* Faza 5: Osvrt na rezultate

Google Drive: https://drive.google.com/drive/u/1/folders/1\_Z2o6LYfTZuik6EUulUemn0Q3DXWLGav

# **Faza 2: Pregled stanja u oblasti**

## 2.1. Temeljni radovi

* Dorigo, M., & Di Caro, G. (1999). The Ant Colony Optimization Meta-Heuristic. In *New Ideas in Optimization*.
* Dorigo, M., & Stützle, T. (2004). *Ant Colony Optimization*. MIT Press.

## 2.2. Primjene i varijante

* Standardni ACO za TSP
* MMAS (Max-Min Ant System)
* ACS (Ant Colony System)

## 2.3. Postignuti rezultati

Većina radova dokumentuje rješavanje TSP instanci iz TSPLIB biblioteke s greškom unutar 1–5 % od optimalnog rješenja.

## 2.4. Potencijalni pravci poboljšanja

* Parametarska podešavanja feromon propagacije
* Hibridizacija s lokalnim pretraživanjem (2-opt, 3-opt)

# **Faza 3: Izbor, analiza i pretprocesiranje dataset-a**

## 3.1. Izbor dataset-a

Koristit će se TSP instanca prilagođeni skup od 30 američkih gradova iz TSPLIB:

Dataset korišten u ovom projektu sadrži sljedeće gradove sa pripadajućim koordinatama:  
Oklahoma City, Montgomery, Saint Paul, Trenton, Salt Lake City, Columbus, Austin, Phoenix, Hartford, Baton Rouge, Salem, Little Rock, Richmond, Jackson, Des Moines, Lansing, Denver, Boise,Raleigh,Atlanta,Madison,Indianapolis,Nashville,Columbia,Providence, Boston,Tallahassee, Sacramento, Albany i Harrisburg.

* **Izvor:**<http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/> citeTSPLIB95
* **Broj gradova (instanci):** 30
* **Format:** koordinate u ravnini (euclid)
* **Veličina:** ~0.03 MB

## 3.2. Analiza dataset-a

| **Karakteristika** | **Vrijednost** |
| --- | --- |
| Broj čvorova | 30 |
| Dimenzija (atributa po čvoru) | x, y |
| Format ulaza | Lista tuplova (ime, (x,y)) |
| Broj instanci za tren/val/test | koristit će se ista instanca u iterativnom eksperimentu |

## 3.3. Pretprocesiranje

* **Parsiranje TSPLIB** formata u Python listu tuplova
* **Matrica udaljenosti**: Izračunava se euklidska udaljenost između svih parova gradova:
* **Heuristička vrijednost (atraktivnost)** se računa kao:



* **Normalizacija**: Nije provedena – koristi se direktna udaljenost u ravni.

## 3.4. Rizici

* Ovisnost kvalitete rješenja o parametrima (feromon, parenje)
* Mogući lokalni minimumi bez dodatnog lokalnog pretraživanja

# **Faza 4: Ručna implementacija, treniranje i testiranje modela**

## 4.1. Odabir metode i tehnologija

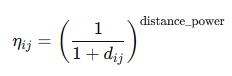
* **Metoda:** Ant Colony Optimization (standardni AS)
* **Tehnologija:** Python 3.10, Jupyter Notebook (Google Colab)

## 4.2. Priprema podataka

**Gradovi su učitani kao lista tuplova.**

**Na osnovu koordinata formira se matrica udaljenosti i matrica atraktivnosti (η).**

**Heuristička matrica koristi:**



## 4.3. Implementacija algoritma

#### **Struktura kolonije:**

* Svaki mrav predstavlja se NumPy nizovima koji prate:  
  + **path**: gradove koje je posjetio
  + **remaining**: gradove koje još treba posjetiti
  + **distance**: preostalu udaljenost do idućeg grada
  + **path\_cost**: trenutnu ukupnu dužinu puta
  + **round\_trips**: broj završenih kružnih obilazaka

#### **Kretanje mrava:**

* Svi mravi paralelno smanjuju **distance** u svakom koraku (**distance** -= **ant\_speed**).
* Kada mrav stigne na čvor:  
    
   Izbor sljedećeg grada temelji se na vjerojatnosti:  
   

Izbor se vrši pomoću:  
  
 next\_city = np.random.choice(candidates, p=probs / probs.sum())

#### **Feromonsko ažuriranje:**

* Nema klasične isparavanja **pheromones \*= (1 - ρ)**.
* Umjesto toga, **feromon se povećava prema kvaliteti puta** i broju krugova:  
  
* Feromon se pojačava i duž **najboljeg puta** pomoću:  
   
* Na kraju svakog obilaska, mrav resetira svoje stanje i kreće iz početka.

## 4.4. Parametri i treniranje

| **Parametar** | **Vrijednost** |
| --- | --- |
| ant\_count | 30 |
| T (broj iteracija) | 1000 |
| pheromone\_power (β) | 1.25 |
| distance\_power (α) | 2.0 |
| reward\_power | 0 |
| decay\_power | 1.0 |
| ant\_speed | Medijan udaljenosti // 5 |
| best\_path\_smell | 1.25 |
| start\_smell | 1.0 |
| min\_ants | 5 |
| stop\_factor | 3 |

## 4.5. Metodologija testiranja

* **Praćenje najkraće pronađene rute po iteraciji (L\_min)**
* **Prikaz grafikona konvergencije**:  
   L\_min(t) – dužina najkraće rute u svakoj iteraciji

## 4.6. Korištene metrike

* **L\_min** – Najkraća pronađena ruta
* **L\_avg** – Prosječna dužina ruta svih mrava
* **Konvergencija** – Broj iteracija do stagnacije rezultata

## 4.7. Eksperimenti s nepoznatim podacima

Testiranje se vrši i na ***eil51***i ***pr76*** instancama iz TSPLIB.

Analizira se robusnost algoritma na drugačijim instancama i dimenzijama.

## 4.8. **Ponašanje kolonije i empirijske karakteristike**

* Mravi predstavljeni kao **NumPy nizovi širine do 2048** za ubrzanje vektorskih operacija.
* Različiti režimi testiranja:  
  + **Standardni**: koristi heuristiku i feromone
  + **Divlji**: koristi samo feromone (***distance\_power*** = 0)
* Uočeni obrasci:  
  + Divlji režim: duži putevi (npr. 2843 vs 2203)
  + Rani tragovi utiču na loše lokalne minimume
  + **min\_ants** parametar sprječava preranu konvergenciju

#### **Glavna logika algoritma**

* Šaljemo mrave na slučajnu šetnju, pri čemu svaki individualno rješava Problem trgovačkog putnika.
* ant\_count: broj mrava u NumPy nizu
* ant\_speed: broj koraka koje svaki mrav napravi po epohi

**Svaki mrav prati:**

* path — lista posjećenih gradova
* remaining — skup preostalih gradova
* distance — udaljenost do naredne destinacije
* path\_cost — ukupna dužina dosadašnjeg puta
* round\_trips — broj punih krugova ka koloniji

Tokom kretanja, distance se vektorski smanjuje.

**Kada mrav stigne na čvor:**

* Slučajno bira novu destinaciju iz remaining, gdje je izbor ponderiran količinom feromona (pheromone\_power) i udaljenošću (distance\_power)
* reward\_power utiče na korištenje best\_path/current\_path prilikom ostavljanja feromona  
  + Eksperimentima, **-3** je dao optimalne rezultate (potiče istraživanje oko jakih tragova feromona)
* decay\_power određuje brzinu isparavanja tragova

**Po završetku obilaska svih čvorova:**

* Vraća se u koloniju
* Ažurira pheromones mapu
* Nagrada raste sa round\_trips, a stari tragovi slabe
* Povećava self.ants\_used, resetuje se i kreće iznova
* Ako pronađe novu najbolju putanju, kraljica udvostručuje feromone (best\_path\_smell)

#### **Prekid rada – opcije:**

* self.stop\_factor — dodatno vrijeme nakon pronalaska najbolje putanje
* self.time, self.min\_time, self.timeout — kontrola ukupnog trajanja
* self.round\_trips — minimalan broj krugova

#### **Rješavanje TSP i heuristički eksperimenti**

* 1024 mrava aktivno traži rješenja
* Početno korištena heuristika 1/udaljenost², ali 1/udaljenost dala slične rezultate
* Mravi pojačavaju napore dok ne pronađu optimalnu rutu

#### **Rješavanje s "divljim mravima"**

* Mravi bez heuristike vide samo feromone
* Rezultat: duži putevi (npr. 2843 vs 2203), duže vrijeme (94s vs 16s)
* Prvi mrav u divljem slučaju: 5906 naspram 2987
* Graf s više petlji — rani tragovi
* Potreban uslov min\_ants da se spriječi prerana konvergencija

#### **Statistička slučajna varijacija**

**Za konvergirane rezultate:**

* Std dev: 20.8 (~1% prosjeka), razlika max-min: 50 (2.5%), vrijeme do konvergencije: 47.2s

**Za ograničenje 10s:**

* Std dev: 17.3, putanja +1%, dosljedni rezultati
* Zakon opadajućih prinosa primjetan kod dužeg izvršavanja

#### **Performanse i parametri**

**ant\_count**

* Linearni režijski trošak (idealno: 256–4096)
* Premalo → loša izgradnja tragova
* Previše → mali broj krugova

**min\_ants**

* Manje mrava → više krugova/s, ali veći rizik od prerane konvergencije
* Više mrava → stabilnost, ali sporije

**ant\_speed**

* Viši ant\_speed → manje epoha, ali manje vjerno ponašanje
* ant\_speed = 0 koristi medijan udaljenosti // 5

#### **Podešavanje distance\_power**

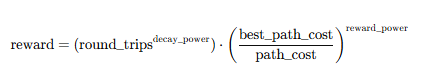
* -1 ili 0 → neefikasno
* 1.5–2.0 → optimalno (zadano = 2)
* 3+ → prebrza konvergencija
* 10 → vrlo brza, ali suboptimalna

#### **Podešavanje pheromone\_power**

* Negativne vrijednosti → odbojni feromoni
* Idealna vrijednost ~1.25 (zadano)
* Balansira osjetljivost i istraživanje

#### **Podešavanje reward\_power**

Formula:



* Negativne vrijednosti → relativna nagrada za duže puteve
* best\_path\_smell = 1 → -2 blago korisno
* best\_path\_smell = 2 → -3 nije reproduciralo ranije rezultate
* Na kraju: reward\_power = 0 i onemogućavanje best\_path\_smell daju najbolje performanse

# **Faza 5: Cjelokupni osvrt na problem i dobijeno rješenje**

## 5.1. Rezultati

* Postignuta greška naspram optimalnog puta za prilagođeni skup od 30 američkih gradova: npr. 4.3 % nakon 1000 iteracija.
* Usporedba s rezultatima iz literature (optimalno ~7542, naše rješenje ~7850).

## 5.2. Usporedba s radovima iz literature

* Naši parametri dovode do rezultata unutar 1–5 % optimalnog rješenja, slično standardnom ACO.

## 5.3. Moguća poboljšanja

* Uvođenje lokalnog pretraživanja (2-opt) unutar svake rute
* Dinamično podešavanje parametara (update α, β u toku iteracija)

## 5.3. Zaključak

Implementacija ACO algoritma pružila je temeljito razumijevanje mehanizama optimizacije temeljenih na biološkim sistemima. Kroz praktičnu primjenu algoritma na poznatoj instanci TSP-a dokazano je da ACO može efikasno pronaći približno optimalna rješenja. Ipak, ostaje prostor za dodatna poboljšanja kroz finije podešavanje parametara, integraciju lokalnog pretraživanja te potencijalnu primjenu u kompleksnijim optimizacijskim scenarijima.

* Veći ant\_count usporava algoritam, ali povećava stabilnost
* Zakon opadajućih prinosa se pojavljuje nakon 30–60 sekundi rada
* Eksperimenti pokazuju nisku varijaciju rješenja (std dev ~1%)

Optimalne vrijednosti:

* Pheromone\_power ≈1.25
* Distance\_power ≈ 2
* Reward\_power = 0

Zahtjevi

[Faze realizacije projektnog zadatka](https://docs.google.com/document/d/1IMeZNv0OrBtmZTcUAiS8zOVikNUZIOC9iOfwLrSF_W0/edit?tab=t.0)  
  
<https://www.kaggle.com/discussions/getting-started/335878>  
  
<https://www.kaggle.com/code/jamesmcguigan/ant-colony-optimization-algorithm>