

TRAVELING SALESMAN PROBLEM

Genetic Algorithm with Parallelized Island Model

ŠTA JE TSP?

Traveling Salesman Problem (TSP) je optimizacioni izazov gde osoba mora da pronađe najkraću moguću rutu da poseti dati skup gradova, obilazeći svaki grad tačno jednom, i zatim da se vrati u početni grad. To je fundamentalni problem u kombinatornoj optimizaciji, poznat po tome što je NP težine, što znači da ne postoji efikasan algoritam za rešavanje svih instanci problema za veliki broj gradova. TSP ima široku primenu u oblastima kao što su logistika, planiranje ruta i sekvenciranje.





IMPLEMENTACIJA

Genetski algoritam imitira proces prirodne selekcije i evolucije kako bi pronašao rešenje za optimizacione probleme. Ključni elementi su:

- **Populacija:** Skup mogućih rešenja, gde svako rešenje predstavlja "jedinku" ili "turu".
- **Selekcija:** Odabir najboljih jedinki iz populacije koje će poslužiti kao "roditelji" za sledeću generaciju. U našoj implementaciji, biramo dve ture sa najkraćim rastojanjem.
- **Ukrštanje (Crossover):** Proces kombinovanja "gena" (delova rute) dva roditelja kako bi se stvorilo novo rešenje, odnosno "dete".
- **Mutacija:** Nasumična promena u genima deteta, koja unosi raznovrsnost u populaciju i sprečava zaglavljivanje u lokalnim optimumima.
- **Fitnes funkcija:** Merilo kvaliteta rešenja. U ovom slučaju, fitnes funkcija je ukupna distanca rute, a cilj je da je minimizujemo.

IMPLEMENTACIJA

- **main.rs**: Glavni fajl koji upravlja celim procesom. On učitava gradove i rute iz datoteke, pokreće seriju i paralelnu verziju algoritma i meri vreme izvršavanja.
- **tour.rs**: Definiše strukturu *Tour* (tura/putovanje) koja predstavlja jedno rešenje. Svaki *Tour* sadrži listu gradova i ukupnu pređenu distancu. Poseduje funkciju *init_tour* koja inicijalizuje rutu sa nasumično raspoređenim gradovima i funkciju *calculate_tour_distance* koja izračunava ukupnu pređenu udaljenost.
- **population.rs**: Definiše strukturu *Population* (populacija), koja je kolekcija *Tour* objekata.
- **genetic_algorithm.rs**: Sadrži osnovnu logiku genetskog algoritma:
 - *selection*: Funkcija koja bira dva najbolja tura iz populacije.
 - *crossover*: Funkcija koja stvara novo dete-tur kombinovanjem delova dva roditelja.
 - *mutation*: Funkcija koja nasumično menja raspored gradova unutar tura kako bi se sprečilo ponavljanje rešenja.
 - *evolution*: Glavna funkcija koja orkestrira celokupni proces evolucije populacije.



SERIJSKA I PARALELNA

U implementaciji postoje dve verzije algoritma:

- **Serijska verzija:** Algoritam se izvršava na jednoj niti. Cela populacija evoluira u jednoj petlji, a najbolje rešenje se prati kroz svaku generaciju.
- **Paralelna verzija:** Populacija je podeljena na više "ostrva", gde svako ostrvo evoluira na zasebnoj niti. Povremeno, "migranti" (najbolja rešenja sa jednog ostrva) se šalju na druga ostrva da unesu novu genetiku i poboljšaju rešenja. Ovo omogućava brže pronalaženje boljih rešenja, naročito kod velikih instanci problema. Mogu se videti senderi i receiveri koji šalju i primaju migrante.

ARGUMENTI

Program omogućava korisniku da podesi ključne parametre genetskog algoritma i njegove paralelne implementacije direktno preko komandne linije. Ovi argumenti su esencijalni za prilagođavanje algoritma specifičnim problemima i performansama. Svaki argument ima kratku i dugu formu, zajedno sa opisom i podrazumevanom vrednošću.

Opis argumenata

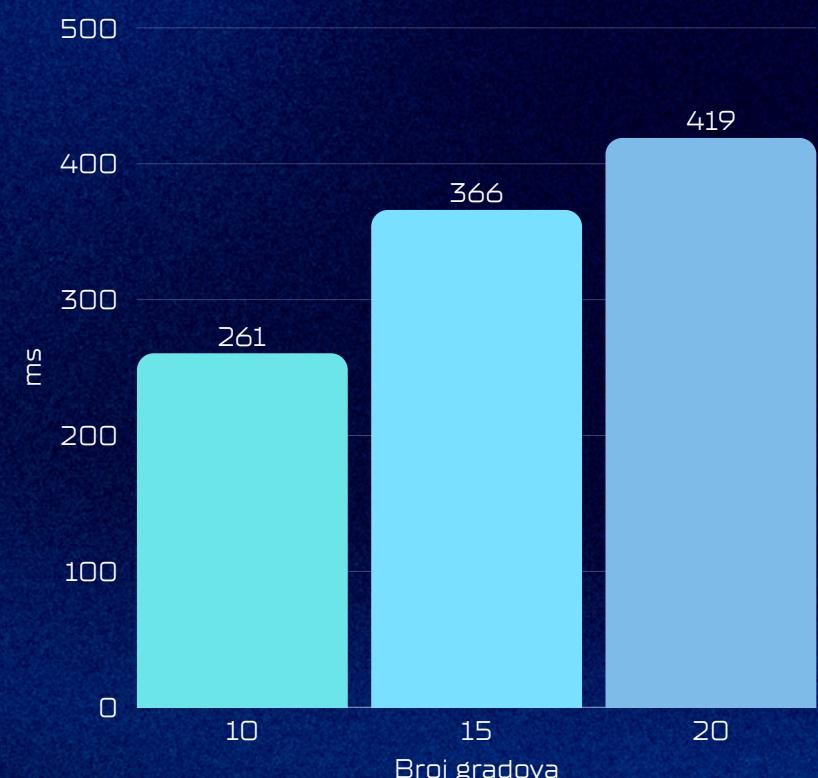
- **-t, --threads:** Broj ostrva [threads]. Podrazumevana vrednost je 14. Svako ostrvo predstavlja zasebnu populaciju koja evoluira nezavisno, a paralelno izvršavanje na više niti omogućava brže rešavanje problema.
- **-c, --cities:** Broj gradova u problemu. Podrazumevana vrednost je 10. Ova vrednost određuje veličinu instance TSP problema koji se rešava.
- **-g, --generations:** Maksimalan broj generacija. Podrazumevana vrednost je 100. Algoritam se izvršava do ovog broja generacija, nakon čega se zaustavlja, bez obzira na kvalitet rešenja.
- **-m, --migration:** Interval migracije. Podrazumevana vrednost je 10. Migracija se dešava na svakih 10 generacija, kada se najbolja rešenja razmenjuju između ostrva.
- **-n, --no_migrants:** Broj migranata. Podrazumevana vrednost je 2. Ovaj argument određuje koliko se najboljih rešenja (jedinki) prenosi sa jednog ostrva na drugo tokom migracije.
- **-e, --expected-mutation-possibility:** Verovatnoća mutacije. Podrazumevana vrednost je 0.2. Mutacija je važan mehanizam koji unosi raznolikost u populaciju i pomaže u izbegavanju lokalnih optimuma. Verovatnoća mutacije se primenjuje na svakoj jedinki u svakoj generaciji.
- **-p, --population:** Veličina populacije po ostrvu. Podrazumevana vrednost je 20. Veličina populacije direktno utiče na raznolikost rešenja i brzinu konvergencije algoritma. Veća populacija obično zahteva više vremena za obradu, ali može dati bolja rešenja.

Population; Generations	10 cities	15 cities	20 cities
10; 100	17ms; 6770 / 33ms; 6590	16.5ms; 11150 / 40.5ms; 10153	27.5ms; 13073 / 58ms; 11535
20; 100	68.5ms; 4002 / 33ms; 3842	38s; 10218 / 94.5ms; 9268	59ms; 10350 / 139.5ms; 11650
30; 100	36ms; 6508 / 80.5ms; 6078	56ms; 7905 / 148.5ms; 7385	169.5ms; 12663 / 226.5ms; 11680
10; 200	61.44ms; 6320 / 120.88ms; 6100	108ms; 9650 / 161ms; 9170	139.5ms; 12453 / 214ms; 12250
20; 200	142.89ms; 4538 / 194.90ms; 4298	74.63ms; 8623 / 189.63ms; 7580	88.94ms; 12333 / 224.37ms; 11175
30; 200	120.54ms; 6300 / 287.54ms; 5740	121.95ms; 9420 / 277.34ms; 9103	161.17ms; 12658 / 398.29ms; 11218
10; 500	66.32ms; 4498 / 160.55ms; 4498	131.47ms; 9353 / 369.62ms; 8808	162.06ms; 12390 / 440.97ms; 11840
20; 500	124.70ms; 5908 / 409.78ms; 5800	208.75ms; 9403 / 752.29ms; 8418	271.97ms; 11803 / 979.17ms; 11200
30; 500	260.52ms; 6005 / 624.14ms; 5455	366.70ms; 10185 / 1.22s; 8990	419.51ms; 12168 / 1.67s; 11508

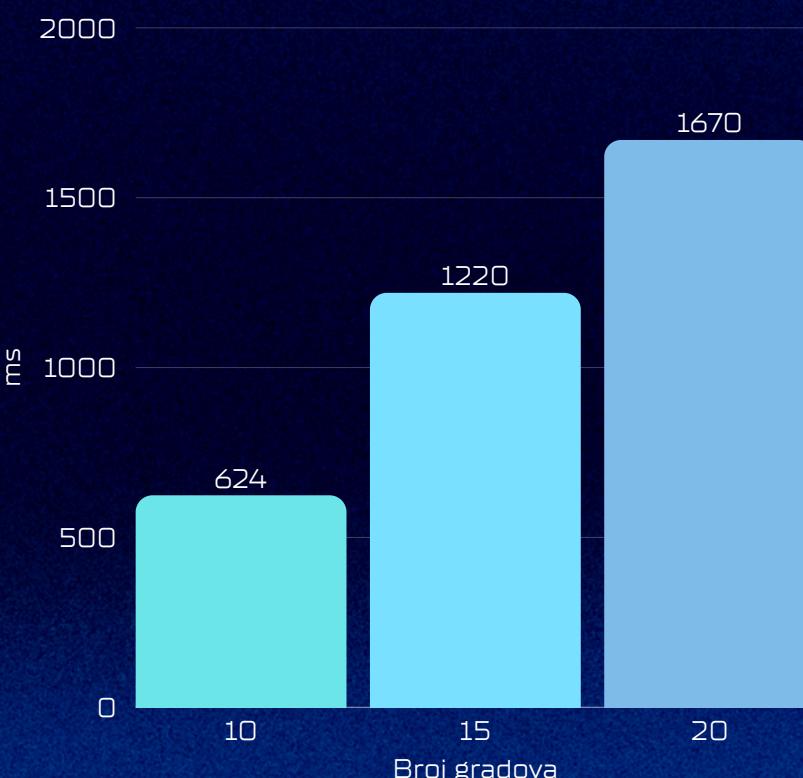
SERIJSKI VS PARALELNO

Vreme izvršavanja

Serijski

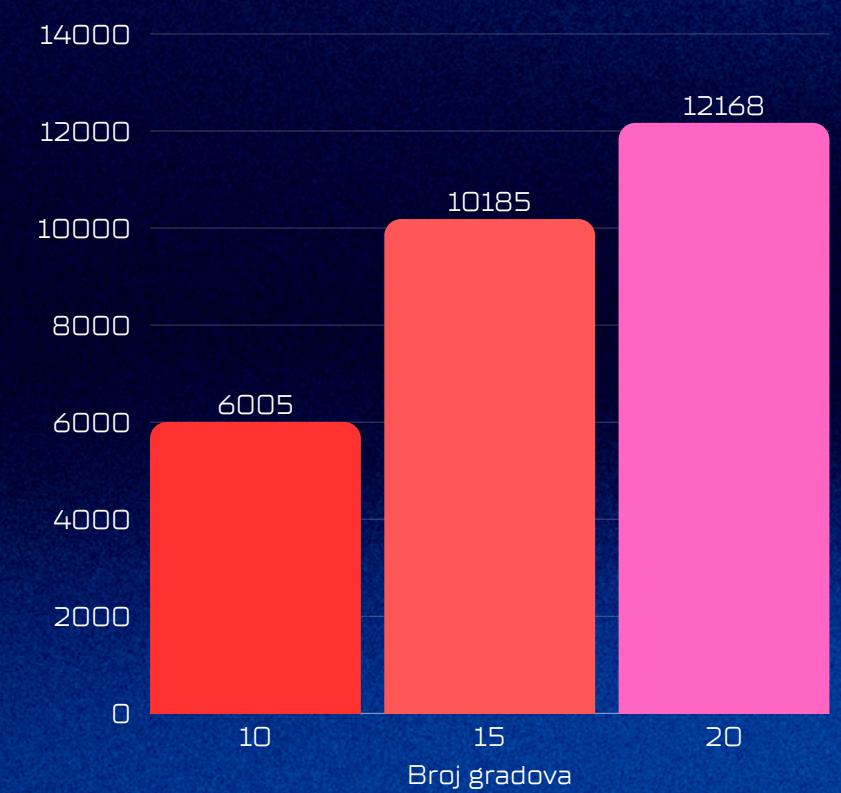


Paralelno

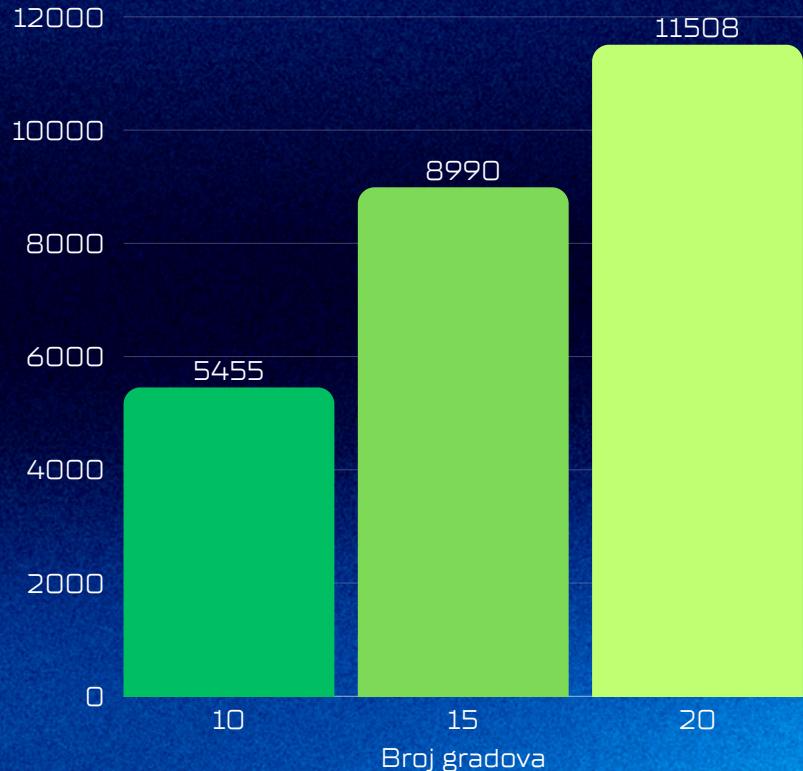


Dužina rute

Serijski



Paralelno



Serijski

Serijska implementacija osvaruje i duplo kraće vreme izvršavanja za iste podatke

Paralelno

Paralelna se duplo duže izvršava, ali nalazi bolje rezultate

ZAKLJUČAK

Serijska implementacija se kraće izvršava i daje dovoljno dobre rezultate od paralelne, koja zbog obima podataka ne daje očekivano bolje rezultate. Potencijalna unapređenja serijske implementacije se mogu ogledati u bolje izboru "roditelja" za sljedeću generaciju, dok kod paralelne bi se model paralelizacije mogao optimizovati, čak i pokušati sa drugim modelima, te uporediti podatke. Ideja ovog projekta jeste upoznavanje sa TSP-om, te radom u Rust programskom jeziku, na jednom ovakovom problemu.

P.S. Imam ideju da napravim front koji bi dozvolio da se na mapi biraju gradovi i da se pronađe udaljenost u odnosu na te tačke na mapi.



HVALA!