Paralelni genetski algoritam sa lokalnom pretragom

Ivana Nestorović 130/2019
Andrijana Ivković 115/2019

September 2023

Sadržaj

1 Uvod

U realnom svetu se javljaju razni problemi optimizacije.Efikasno rešavanje ovih problema igra ključnu ulogu u poboljšanju performansi mnogih sistema. U cilju rada na ovim izazovima, razvijene su različite tehnike optimizacije, a tehnika koja je prikazana u ovom radu predstavlja kombinaciju genetskog alogoritma i lokalne pretrage, koji je hibridizovan paralelizacijom.

1.1 O radu

Na konkretnom problemu smo isprobali pojedinačne algoritme, zatim ih kombinovali, prikazivali njihove hibridizacije i uporedjivali dobijene rezultate. Za potrebe istraživanja u ovom radu smo koristili NP-težak problem TSP(Problem trgovačkog putnika).
TSP je klasičan kombinatorni problem optimizacije koji zahteva pronalaženje najkraće moguće ture koja prolazi kroz sve gradove i vraća se u polazni grad. Svaki grad treba posetiti tačno jednom, a ukupna udaljenost koju putnik predje treba da bude minimizovana. Ovaj problem ima važne primene u logistici, rutiranju, planiranju putovanja i drugim oblastima gde je potrebno naći najefikasniji put obilaska nekoliko lokacija.
Hibridizacija genetskog algoritma (GA) predstavlja efikasan pristup za rešavanje problema trgovačkog putnika (TSP). Iako je nemoguće garantovati potpuno tačno rešenje zbog NP-težine ovog problema, kombinacija GA i lokalne pretrage omogućava pronalaženje visoko kvalitetnih rešenja. Rešenja se mogu dodatno poboljšati podašavanjem parametara genetskog algoritma i lokalne pretrage.
Projekat je implementiran u programskom jeziku Python, a izvršavan je pomoću Jupyter Notebook-a.

1.2 Skup podataka

Za skup ulaznih podataka koristili smo json fajl, koji sadrži koordinate 29 gradova iz Zapadne Sahare. Zbog kompaktnosti,informacije o gradovima čuvali smo u grafu, pri čemu grane grafa predstavljaju njihove udaljenosti, podrazumevajući da izmedju svaka dva grada postoji put. Za računanje rastojanja izmedju dva grada, korišćeno je Euklidsko rastojanja.

2 Genetski algoritam

Genetski algoritam (GA) je optimizacioni algoritam inspirisan procesima evolucije u prirodi. Zasnovan je na Darvinovoj teoriji evolucije. GA je moćan alat koji se koristi za rešavanje raznih problema optimizacije, kao i za pretraživanje prostora rešenja u velikim i kompleksnim prostorima. Medjutim, treba napomenuti da nije uvek garantovano da će pronaći globalni optimum, jer zavisi od parametara i prirode problema.
Prvi korak genetskog algoritma je inicijalizacija početne populacije jedinki. Svaka jedinka predstavlja moguću rutu obilaska gradova. Početna populacija je generisana nasumično. Svaka jedinka u GA ima svoj genetski kod koji predstavlja potencijalno rešenje problema. U kontekstu TSP-a, svako moguće rešenje se posmatra kao jedinka koja ima svojstvo (genetski kod) koje predstavlja redosled poseta gradova. U okviru jedinke implementirana je i fitnes funkcija koja kvantifikuje koliko je jedinka "dobro prilagodjena" za rešavanje problema. Cilj je da se minimizuje ili maksimizuje vrednost fitness funkcije u zavisnosti od prirode problema. U TSP-u, fitnes jedinke je ukupna dužina puta koji se predje obilazeći gradove u redosledu odredjenim genetskim kodom, cilj nam je minimizacija.

Naredni koraci GA su:
Selekcija: U procesu selekcije, bolje ocenjene jedinke (sa manjim fitnesom) imaju veću verovatnoću da budu izabrane kao roditelji za stvaranje nove generacije. U ovom implementiranom algoritmu koristi se selekcija turnirom, gde se nasumično bira nekoliko jedinki iz populacije, a najbolje ocenjena jedinka od odabranih jedniki postaje roditelj.
Ukrštanje: proces kombinovanja genetskog materijala dve roditeljske jedinke kako bi se stvorila nova jedinka (potomak). Ovde smo koristili uniformno ukrštanje, gde se nasumično odabere deo genetskog koda od jednog roditelja, a preostali deo genetskog koda se dopisuje iz drugog roditelja, bez ponavljanja već posećenih gradova.
Mutacija: proces nasumične promene genetskog koda jedinke. U ovom slučaju, jedan gen (grad) može biti zamenjen sa nekim drugim gradom, uz odredjenu verovatnoću kako bi se osigurala raznolikost u populaciji.

Da bismo sačuvali najbolje članove koristili smo **elitizam**. To je tehnika koja se često koristi u genetskim algoritmima kako bi se najbolje jedinke iz jedne populacije prenele u narednu, bez modifikovanja. Ova tehnika pomaže u očuvanju visoko ocenjenih rešenja, čime se osigurava da se vrhunska svojstva prenose kroz generacije.

Genetski algoritam se izvršava u nekoliko iteracija (generacija) i može se zaustaviti nakon odredjenog broja generacija ili kada se postigne zadovoljavajuće rešenje, odnosno kad nema promene najboljeg fitnesa posle odredjenog broja generacija.

3 Lokalna pretraga

Lokalna pretraga je optimizacioni algoritam koji se koristi za pronalaženje najboljeg rešenja u okolini trenutnog rešenja. Ovaj algoritam fokusira se na poboljšanje rešenja iterativnim postupcima, pri čemu započinje s nekim početnim rešenjem i pokušava ga poboljšati tako da traži rešenja koja su bliža lokalnom optimumu u okolini trenutnog rešenja.
Za naš slučaj, ovaj algoritam nastoji da poboljša jedinku tako što slučajno menja njene gene i proverava da li se fitness funkcija poboljšava. Ako se fitness povećava, prihvatamo promenu; inače, kod jedinke ostaje nepromenjen. To se ponavlja tokom nekoliko iteracija kako bi pronašli bolja rešenja.

3.1 Kombinovanje sa genetskim algoritmom

Dodavanjem lokalne pretrage u genetski algoritam unapredjujemo GA na način da omogućavamo dodatno poboljšanje rešenja koje dobijamo pomoću genetskog algoritma.
Dodatna lokalna pretraga povećava verovatnoću pronalaženja boljeg rešenja jer GA može pronaći bolje inicijalno rešenje, a zatim ga dodatno poboljšati lokalnom pretragom.
Lokalna pretragu primenjujemo na jedinke u svakoj generaciji GA-a. Nakon što generišemo nove jedinke, na odredjen broj jedinki iz populacije (unapred zadat) primenjujemo lokalnu pretragu kako bi se potencijalno poboljšao njihov kvalitet. Izbor broja jedinki koji podvrgavamo lokalnoj pretrazi je bitan, jer primenom na sve jedinke iz populacije moguće je da dobijemo neko lokalno rešenje.

4 Paralelizacija

Paralelizacija omogućava istovremeno izvršavanje više ekosistema jedinki. Pošto se kod višeprocesorskih računara život jedinki odvija paralelno, za slično vreme kao kod jednog sistema, postižemo pronalaženje najboljih jedinki dobijenih različitim inicijalizovanjem početnog skupa jedinki.
Različite GA instance mogu konvergirati prema različitim lokalnim optimumima, čime se povećava diverzifikacija populacije i smanjuje rizik od zaglavljenosti u lokalnim optimumima.
U prvoj verziji našeg paralelizovanog algoritma tražimo jednu jedinku koja ima najbolji fitnes, u odnosu na jedinke iz svih ekositema i nju uzimamo kao globalni optimum. Medjutim dodatno poboljšanje rezultata možemo postići korišćem paralelnog algoritma za inicijalizovanje početnog skupa jedniki jedinstvenog ekositema.

4.1 Inicijalizovanje populacije

U svakom od procesa, izvršava se GA sa različitim početnim populacijama. Nakon završetka tih GA instanci, rezultati se prikupljaju u red.

Sve najbolje jedinke iz tih instanci se kombiniraju u jednu novu populaciju koja se zatim koristi kao deo početne populacije za još jednu iteraciju GA-a. Stare jedinke će predstavljati elitni deo populacije.
Nakon kombinacije populacija, pokreće se još jedna iteracija GA-a s tom novom populacijom. Ovo omogućava dalje poboljšavanje rešenja.

5 Postignuti rezultati

Za testitanje rezultata koristili smo sledeće parametre:
-veličinu populacije od 1000 jedinki,
-broj iteracija 100,
-veličinu turnira 5,
-verovatnoću mutacije 0.05,
-upotrebu elitizma, veličine skupa elitnih jedinki 10,
-zaustavljanje nakon 10 iteracija ukoliko nema promene fitnesa.

Postavljenjem drugačijih parametara postigli bismo drugačije rezultate, o čemu će biti reči kasnije.
Primeri su testirani na računaru sa 4 jezgra, brzine 2.30GHz.

5.1 Običan genetski algoritam

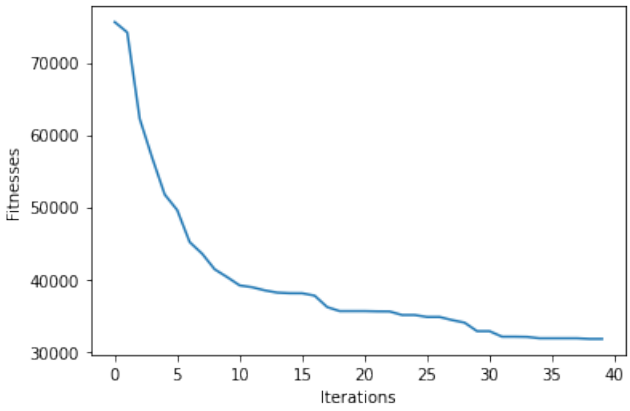

Figure 1: Genetski algotitam

Algoritam se zaustavlja pre 40 iteracije, i dostiže najbolji fitnes 31728. Vreme potrebno za izvršavanje algoritma je 3.8126986026763916s.

Na grafiku iznad vidimo najbolje fitnese koji se dobijaju pozivanjem GA algoritma 10 puta.

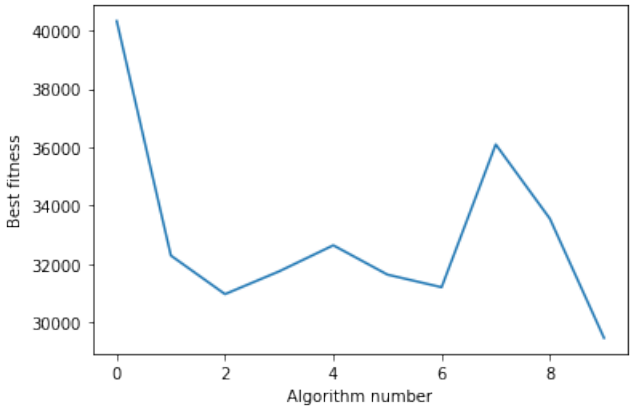

Figure 2: Genetski algoritam

5.2 GA sa lokalnom pretragom

Za parametar broj jedinki za lokalnu pretragu koristimo 10.

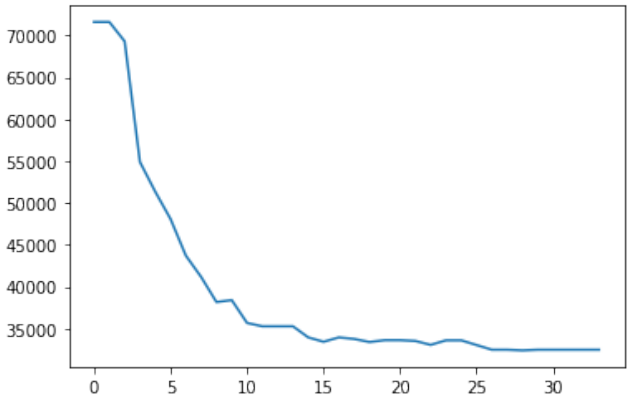

Figure 3: GA sa lokalnom pretragom

Algoritam se zaustavlja posle 30 iteracije, i dostiže najbolji fitnes 32467. Vreme potrebno za izvršavanje algoritma je 9s. Vidimo da je vreme potrebno za izvršavanje veće u odnosu na običan genetski algoritam, ali posle samo 10 iteracija se približava minimumu.
Ovaj algoritam ima problem sa upadanjem u lokalni minimum.

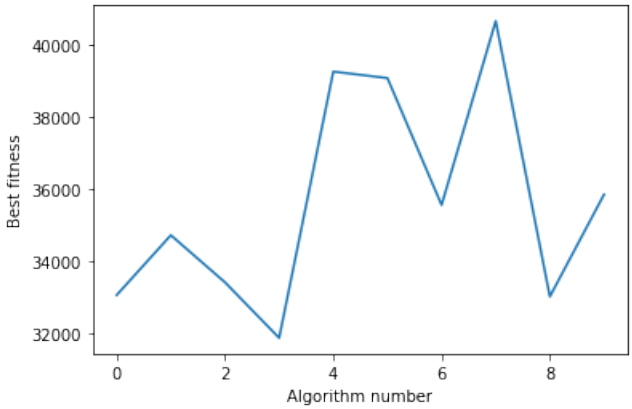

Figure 4: GA with local seatch

5.3 Paralelizacija

Za broj procesa uzimamo 4, zbog postojanja jezgra procesora.

5.3.1 Najbolja jedinka iz više procesa

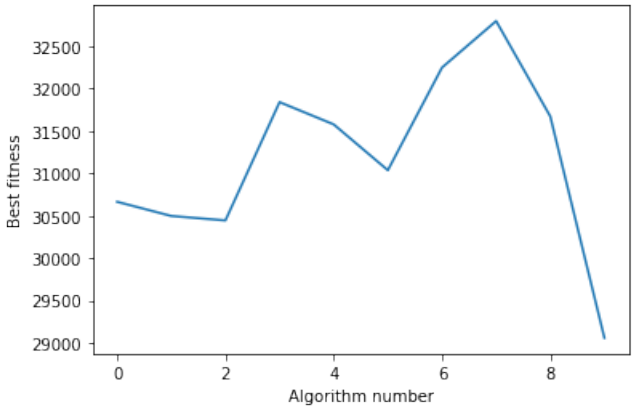

Figure 5: Najbolja jedinka

Na ovom grafiku možemo uočiti dostizanje najmanje fitnes vrednosti u odnosu na ostale algoritme, ali takodje primećujemo da se fitnesi manje razlikuju ukoliko više puta pozivamo ovaj isti algoritam.
To je zbog toga što svaka primena algoritma podrazumeva izbor jedne od 4 najbolje. U desetoj iteraciji dobijamo minimum od 29000.

5.3.2 Novi genetski algoritam sa najboljim jedinkama iz svakog procesa

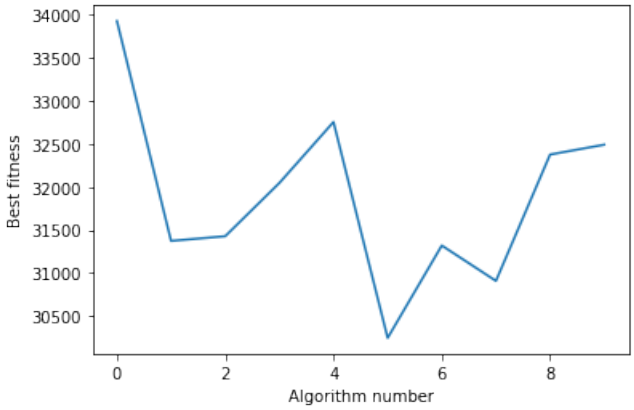

Figure 6: Paralelni GA

6 Zaključak

6.1 Prednosti i mane

Ukratko ćemo opisati uočene prednosti i mane istraženih algoritama.

Genetski algoritam (GA):
Prednosti: Jednostavan za implementaciju i razumevanje. Daje dobre rezultate ako imama dobru početnu populaciju. Brz je.
Mane: Može konvergirati prema lokalnom optimumu, za loš izbor početne populacije. Biće mu potrebno više vremena da iskonvergira ka optimumu.

GA sa lokalnom pretragom:
Prednosti: Pomaže u bržem konvergiranju prema boljim rešenjima. Istražuje prostor lokalno.
Mane: Povećava vremensku složenost algoritma zbog dodatne lokalne pretrage. Može zahtevati pravilno podešavanje parametara. Upadanje u lokalni minimum.

Paralelni GA sa lokalnom pretragom za jednu jedinku: **Prednosti**: Značajno ubrzava proces pretrage korištenjem paralelizacije. Povećava šanse za pronalazak boljih rešenja u istom vremenskom okviru. Omogućava pronalaženje rešenja bližem lokalnom optimumu.
Mane: Zahteva više računarskih resursa.

Paralelni GA sa lokalnom pretragom za izbor više jedinki i ponovno GA:
Prednosti: Kombinuje prednosti paralelizacije s lokalnom pretragom i dodatnim ponovnim GA-om. Povećava šanse za pronalazak boljih rešenja i bolje konvergira prema globalnom optimumu. Poboljšava raznolikost populacije kroz kombinaciju najboljih jedinki.
Mane: Zahteva više računarskih resursa od običnog GA-a. Može biti složen za implementaciju i podešavanje parametara.

Eksperimentiranje s različitim pristupima i podešavanjem parametara često je ključno za postizanje najboljih rezultata.

6.2 Podešavanje parametara

Podešavanje parametara u genetskom algoritmu i njegovim varijacijama ključno je za postizanje uspješnih rezultata. Svaki parametar utiče na ponašanje algoritma i njegovu sposobnost pronalaženja optimalnih rešenja.

Veličina populacije: Povećanje veličine populacije može poboljšati raznolikost populacije i povećati šanse za pronalaženje boljih rešenja. Smanjenje veličine populacije može smanjiti resurse potrebne za izvodjenje GA-a i smanjiti vreme izvršavanja.
Broj iteracija:
Povećanjem broja iteracija povećavamo verovatnoću pronalaženja boljih rešenja. Medjutim, prevelik broj iteracija može produžiti vreme izvodjenja algoritma.
Verovatnoća mutacije:
Povećanje verovatnoće mutacije može pomoći algoritmu da istraži više različitih rešenja, ali previše mutacija može dovesti do gubitka korisnih informacija.
Veličina turnira: Manja veličina turnira može povećati pritisak selekcije, što može dovesti do brže konvergencije prema boljim rešenjima, ali i smanjiti raznolikost populacije.
Upotreba elitizma i veličina elitizma:
Elitizam može osigurati da najbolja rešenja ne budu izgubljena tokom evolucije. Povećanje elitizma može povećati pritisak selekcije prema najboljim rešenjima, ali može smanjiti raznolikost populacije.
Zaustavni kriterijum:
Podešavanje zaustavnog kriterija može kontrolisati trajanje izvodjenja algoritma. Zaustavljanje posle manjeg broja iteracija može rezultirati bržim zaustavljanjem, ali postoji rizik od nedovoljnog istraživanja prostora rešenja.
Veličina lokalne pretrage u GA s lokalnom pretragom:
Povećanjem veličine lokalne pretrage možemo tražiti bolja lokalna poboljšanja, ali to može povećati vremensku složenost i šansu za konvergencijom ka lokalnom optimumu.

7 Literatura

Materijali sa kursa Računarska inteligencija
"Computational Intelligence - An Introduction", Andries Engelbrecht, John Willey and Sons, 2007