Paralelni Ostrvski Genetski Algoritam za Dizajn Radio Mreže

Projektni zadatak u okviru kursa Računarska inteligencija
Matematički fakultet, Univerziteta Beograd

Tijana Todorov, David Nedeljković
tijana.todorov710@gmail.com, dnedeljkovic710@gmail.com

16. april 2020

Sažetak

Ovaj rad je baziran na realnom problemu kombinatorne optimizacije, kao primer koji pokazuje kako genetski algoritam (GA) može biti paralelizovan na efikasan način. Problem koji ćemo razmatrati ima za cilj određivanje najboljeg skupa lokacija radio predajnika kako bi se pokrio što veći prostor po optimalnoj ceni. Dati problem ćemo prvo testirati na Standardnom Genetskom Algoritmu (SGA), a potom na Ostrvskom modelu Genetskog Algoritma (IGA) koji ćemo paralelizovati na višejezgarnom procesoru. Na kraju ćemo diskutovati o kvalitetu rezultata dobijenih primenom oba algoritma, kao i ubrzanja postignuta procesom paralelizacije.

Sadržaj

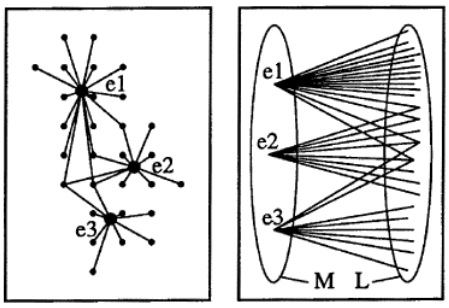
1 Uvod

Jedan od ključnih pitanja sa kojima se susreću telekomunikacione kompanije prilikom dizajniranja radio mreža je izbor dobrog skupa lokacija radio predajnika. Problem se svodi na pokrivanje maksimalnog područja sa minimalnim brojem predajnika. Za područje kažemo da je pokriveno skupom predajnika ako svaka lokacija tog područja može da primi emitovani signal sa bilo kog predajnika. Skup lokacija na kojima se mogu postaviti predajnici uzima se kao ulaz, a naš cilj je da pronađemo minimalan podskup lokacija koji pruža kvalitetnu uslugu.

Ostatak rada organizovan je na sledeći način. Sekcija 2 pokazuje kako se problem može modelovati. Prvi dobijeni rezultati sa SGA koji je implementiran i testiran na veštačkom primeru prikazani su u Sekciji 3. Sekcija 4 predstavlja koncept *ostrva* i prikazuje kako je naš algoritam paralelizovan. Performanse paralelizovanog IGA diskutovaćemo u Sekciji 5.

2 Definisanje problema

Problem radio pokrivenosti koji je uveden u Sekciji 2 odnosi se na pokrivanje područja skupom predajnika [4]. Deo područja koje pokriva predajnik naziva se ćelija. Ćelije obično nisu istih dimenzija ali radi demonstracije u našem radu će biti fiksne. Posmatrajmo dva skupa, L i M. Skup L predstavlja sve potencijalno pokrivene lokacije a skup M sve potencijalne lokacije predajnika. Vezu između predajnika i potencijalnih lokacija koje pokrivaju biće predstavljena grafom G (slika 1). Graf G za čvorove ima skup potencijalnih lokacija predajnika(M) kao i skup potencijalnih lokacija koje pokrivaju(L) dok je E skup ivica takvih da svaka lokacija predajnika je povezana sa lokacijama koje on pokriva. Traženje minimalnog podskupa predajnika koji pokriva maksimalnu površinu područja definisaćemo na sledeći način: $M' \subseteq$ M tako da je |M'| minimalna i $|Susedi(M', E)|$ tako da je maksimalna, gde je $|Susedi(M', E)| = \{u \in L \mid \forall\ v \in M', (u,v) \in E\}$.



Slika 1: Bipartitan graf

Problem koji posmatramo je sličan problemu pokrivanja jedinstvenog skupa(USCP) za koji se zna da je NP-težak[1]. Naš problem se razlikuje od USCP-a po tome što je cilj odabrati podskup predajnika koji će obezbediti dobro pokrivanje područja za razliku od USCP-a koji osigurava potpunu pokrivenost. Iz našeg problema proizilazi činjenica da imamo dvostruki cilj. Ako bi minimizirali |M'| dobili bi trivijalno rešenje(M' =

$\emptyset$). Ako bi maksimizirali broj pokrivenih lokacija problem bi sveli na već pomenuti problem USCP [1]. Zbog toga, definišemo fitnes funkciju sa dva cilja na sledeći način:

$$f(x) = \frac{StopaPokrivenosti^{\alpha}}{BrojOdabranihPredajnika}$$

gde je, StopaPokrivenosti = 100 x |Susedi(M')| / |Susedi(M)|. Parametar α se može podesiti tako da favorizuje stopu pokrivenosti u odnosu na broj predajnika. Sada je cilj maksimizovati funkciju. U praksi, do sada su telekomunikacione firme smatrale da se za parametar $\alpha = 2$ dobijaju kvalitetni rezultati.

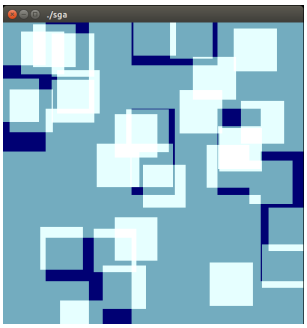
3 Testiranje problema sa Standardni Genetski Algoritam (SGA)

Ovaj algoritam [3] radi sa skupovima koje zovemo populacija. Populacija se sastoji od jedinki koje mogu biti potencijalna rešenja. Svaka jedinka je predstavljena nizom Bool vrednosti (True,False) koji označava da li je lokacija predajnika izabrana ili ne. Izvršavanje SGA završava se za unapred definisan broj generacija. Algoritam ćemo testirati na veštačkom primeru i kvalitet rezultata ćemo uporediti sa algoritmom IGA koji ćemo kasnije obraditi.

Za naše testiranje problema definisaćemo polje dimenzija 287x287 koje će biti podeljeno na 49 ćelija dimenzija 41x41. U svaku ćeliju ćemo postaviti po 3 predajnika od kojih je jedan u centru ćelije a druga dva na slučajno odabranim lokacijama. Za razliku od realnog problema gde je jačina emitovanja signala predajnika različita, u našem primeru će biti fiksna. Da bi postojala mogućnost da se dobije totalna pokrivenost polja postavili smo jačinu predajnika na 20.5 kako bi svaka ćelija bila pokrivena centralnim predajnikom. Algoritam ćemo testirati sa 250 iteracija u okviru populacije od 160 jedinki. Za proces elitizma uzećemo 15% najboljih jedinki i preneti ih u narednu populaciju. Koristićemo jednopoziciono ukrštanje a za vrednost parametra operatora mutacije uzećemo 0.007. Za operator selekcije koristili smo turnirsku selekciju. Izvršavanje algoritma je trajalo oko 30 minuta (slika 2). Osnovna dva nedostatka ovog algoritma su sledeća: kao prvo, veoma je spor i kao takav nije upotrebljiv u realne svrhe i kao drugo ne daje uvek rešenja blizu optimalnom.

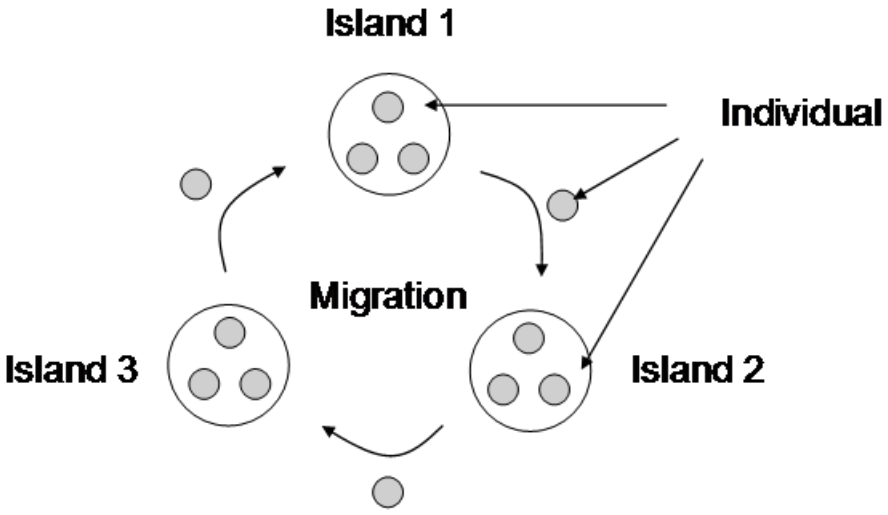
4 Testiranje problema sa Paralelni Ostrvski Genetski Algoritam (PIGA)

Jedan od nedostataka SGA je sporost izvršavanja programa za koji postoje dva razloga koja to objašnjavaju. Prvi, generisanje bipartitnog grafa koji predstavlja odnose izmedju predajnika i pokrivenih lokacija, svaki put kada se računa fitnes funkcija jedinke. Drugi razlog se odnosi na populaciju koja sadrži veliki broj jedinki koji zahteva dosta računanja kako bi se dobili kvalitetniji rezultati iz čega proizilazi znatno usporavanje algoritma. Ovaj nedostatak ćemo rešiti ostrvskim modelom GA [3].



Slika 2: Rezultat testiranog SGA

Ostrvski model SGA predstavlja njegovo proširenje. Glavna osobina ovog modela je ta, da se populacija podeli na nekoliko podpopulacija. Svaka podpopulacija treba da sadrži više od jedne jedinke. Podpopulaciju još nazivamo ostrvo po čemu je i ovaj model dobio ime. Na svako ostrvo ćemo primeniti SGA koji se mogu nezavisno izvršavati i tako povećati šansu za pronalaženje dobrog rešenja, što sugeriše na to da se IGA može paralelizovati. Ostrva razmenjuju informacije kako bi održali raznolikost populacije. Razmena jedinki između ostrva je poznata kao operator migracije (slika 3). Ovaj operator, ostrvski model razlikuje od drugih paralelnih modela genetskog algoritma. Pseudokod ostrvskog modela GA je dat u nastavku:



Slika 3: Prstenasta topologija za operator migracije

Algorithm 1: Pseudo-kod ostrvskog modela GA

```
Generisanje početne populacije od p jedinki ;
Ocenjivanje fitnesa za sve jedinke ;
Podela populacije na n ostrva ;
repeat
    foreach ostrvo do
        repeat
            Selekcija jedinki za primenu genetskih operatora ;
            Ukrštanje za izabrane parove jedinki ;
            Mutacija izabranih jedinki ;
            Ažuriranje fitnesa modifikovanih jedinki ;
            Generisanje populacije za sledeću generaciju ;
        until broj iteracija po ostrvu;
        Migracija m najboljih jedinki na susedno ostrvo ;
    end
until maksimalan broj iteracija;
```

Operatori selekcije, ukrštanja i mutacije se vrše istovremeno na svakom ostrvu. Ostrva su raspoređena u prstenastu topologiju u kojoj se migracije vrše duž tog prstena. Ova topologija se koristi da bi se minimizovala količina migracije kao i komunikacija između ostrva. Nakon izračunavanja nove populacije, šalje se nekoliko najboljih jedinki na sledeće ostrvo. Naredno ostrvo u prstenu dobija nove jedinke koje zamenjuje sa najlošijim na trenutnom ostrvu. Za paralelizaciju IGA koji koristi jedan procesor koristili smo mehanizam multi-threading u okviru OpenMP biblioteke. Ovaj algoritam je testiran na 10, 20 i 40 ostrva što je i prikazano na slici 4.

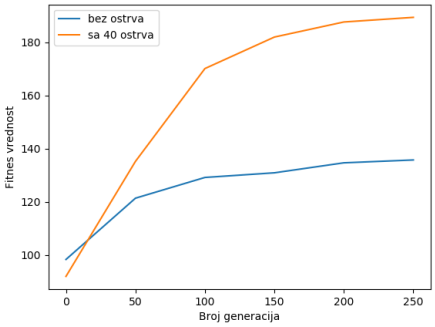
Slika 4: Rezultati paralelnog IGA na 10, 20 i 40 ostrva

5 Kvalitet rešenja i ubrzanje

Na slici 4 prikazana su rešenja PIGA [2], redom za 10, 20 i 40 ostrva. Za fitnes vrednost primenom SGA sa jednom populacijom od 160 jedinki(slika 2) dobili smo 139,26. Primenom PIGA za 10 ostrva i 16 jedinki po ostrvu za fitnes vrednost dobili smo 165,08 dok za 20 ostrva i 8 jedinki po ostrvu 184,88 i za 40 ostrva i 4 jedinke po ostrvu 198,66. Posmatrajući dobijene rezultate možemo uvideti da PIGA daje znatno kvalitetnije rezultate u odnosu na SGA. Iz ovoga možemo zaključiti da broj ostrva značajno utiče na krajnje rešenje. Ovakvo ponašanje možemo objasniti na sledeći način: ukoliko populacija ostrva prebrzo konvergira ka lokalnom optimumu, tu konvergenciju možemo usporiti odabirom najbolje jedinke koja je migrirala iz susednog ostrva.

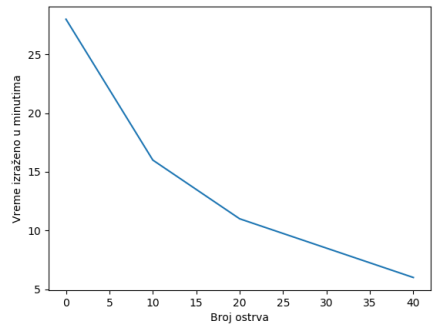
Na slici 5 smo prikazali kako broj generacija utiče na vrednost fitnes

funkcije. Može se primetiti da se fitnes vrednost brže povećava kod populacije koja je podeljena na 40 ostrva u odnosu na populaciju kod SGA.



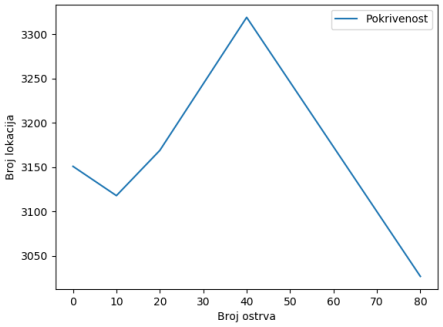
Slika 5: Odnos fitnes funkcije sa brojem generacija

Testiranjem PIGA smo zaključili da se povećanjem broja ostrva vreme izvršavanja algoritma linearno smanjuje. Međutim treba biti obazriv sa odabirom broja ostrva jer može doći do toga da vreme izvršavanja prestane da se smanjuje i da se čak poveća, što se desilo pri testiranju algoritma sa 80 ostrva. Na slici 6 je prikazano vreme izvršavanja algoritma za 10, 20 i 40 ostrva gde se jasno vidi da vreme linearno opada. Razlog tome je paralelizacija koja je vršena multi-threding mehanizmom kao što je pomenuto u Sekciji 4. Komandom *#pragma omp parallel for* smo ograničili blok koji se paralelizuje i u kome se istovremeno izvršava određen broj niti. Broj niti je jednak broju ostrva, zbog čega se za veći broj ostrva npr 40, istovremeno izvršavaju 40 niti sa manjim brojem jedinki u sebi pa je za 250 iteracija bilo potrebno oko 6 minuta, dok je za 10 niti koji imaju vise jedinki u sebi bilo potrebno oko 16 minuta.



Slika 6: Odnos vremena izvršavanja i broja ostrva

Na slici 7 je grafički prikazana pokrivenost lokacija sa različitim brojem ostrva na kojoj se jasno vidi da 80 ostrva daju lošije rezultate. Iako se algoritam duže izvršavao, pokrivenost je bila manja a preklapanja predajnika veća nego na drugim testovima.



Slika 7: Odnos pokrivenih lokacija i broja ostrva

Oba algoritma su testirana na računaru koji ima sledeće karakteristike: RAM Memoriju - 4GB i snagu 4-jezgarnog procesora 2,3GHz. Za merenje performansi naše paralelne implementacije koristićemo sledeću metriku - *ubrazanje*. Ubrzanje nam pokazuje koliko je puta PIGA brži u odnosu na SGA. Ubrzanje ćemo definisati na sledeći način: $Ubrzanje = T_{SGA}/T_{PIGA}$, gde je T_{SGA} - ukupno vreme izvršavanja SGA a T_{PIGA} - vreme izvršavanja PIGA za N ostrva. Konkretno, za PIGA sa 40 ostrva se postiže značajno ubrzanje od 4 ipo puta u odnosu na SGA što ga čini najboljim u našem testiranju.

6 Zaključak

U ovom radu smo opisali jedan od realnih problema kombinatorne optimizacije koji se sreće u stvarnom životu u oblasti telekomunikacija. Na početku smo definisali model našeg problema nad veštačkim skupom podataka kako bi procenili kvalitet rešenja. Osnovni cilj nam je da dobijemo što veću pokrivenost sa što manjim brojem predajnika. Testiranjem našeg problema za dobijanje rešenja na SGA uvideli smo da zahteva dosta vremena pri izvršavanju, što ga čini neprihvatljivim u praksi. Pored nedostatka koji se odnosi na vreme izvršavanja, SGA ne eliminiše na efikasan način predajnike iz čega proizilazi veliki broj preklapanja što smatramo dodatnim nedostatkom. Za poboljšanje rezultata primenili smo IGA koji smo paralelizovali kako bi smanjili vreme izvršavanja. Nakon testiranja PIGA nad različitim brojem ostrva videli smo da se vreme izvršavanja algoritma smanjilo i do 4 puta, broj predajnika sveo na minimum a pokrivenost popela na 96% i kao takav veoma je koristan u praksi. U ovom radu, algoritmi su testirani na visejezgarom procesoru, što ne predstavlja maksimalno ubrzanje koje se može postići. Ukoliko žudite za brzinom, mi Vam preporučujemo da pročitate kako funkcioniše paralelizacija na više-procesorskim računarima.

Literatura

[1] Richard Dapoigny Moonis Ali. *Advances in Applied Artificial Intelligence*. Springer, Berlin, Heidelberg, 2006.

[2] OSMAN BALCI, RAMESH SHARDA and STAVROS A. ZENIOS. *Computer Science and Operations Research.* Pergamon 1992.

[3] Andries P.Engelbrecht. *Computational intelligence.* Chichester, England; Hoboken, N.J.: John Wiley & Sons, 2007.

[4] Patrice Calegari, Frederic Guidec, Pierre Kuonen, and Daniel Kobler. *Parallel Island-Based Genetic Algorithm for Radio Network Design.* Swiss Federal Institute of Technology, CH-1015 Lausanne, Switzerland, 1997.

[5] Patrice Calegari, Frederic Guidec, Pierre Kuonen, EPFL - DI - GRIP/LITH. *Urban radio network planning for mobile phones.* Supercomputing review 1997.