Rešavanje problema minimalnog raspoređivanja transfera datoteka

Popov David, Golubović Stefan
mi16102@alas.matf.bg.ac.rs, mi16135@alas.matf.bg.ac.rs

May 25. 2020

Sažetak

Ovaj rad se bavi metodama minimalnog raspoređivanja transfera datoteka. Cilj je minimizovati potrebno vreme za transfer fajlova u distribuiranoj mreži. Dati problem je NP-kompletan, i zbog toga je kao reper uzet algoritam Demand Protocol, a kao poboljšanje je korišćen genetski algoritam uz modifikaciju parametara eksperimentalnim putem.

Sadržaj

1 Uvod **2**

2 Opis sistema **2**

3 Algoritmi **2**
3.1 Demand Protocol 1 2
3.2 Genetski algoritam 3
3.2.1 Genetski algoritam prva verzija 3
3.2.2 Genetski algoritam druga verzija 4

4 Rezultati **5**

5 Zaključak **7**

1 Uvod

U ovom radu se bavimo nalaženjem minimalnog rasporeda tranfera datoteka sa ciljem da minimizujemo vreme potrebno da se raspored napravi kako bi se transfer datoteka u mreži obavio što je brže moguće. Problem je NP-kompletam tako da ne garantujemo nalaženje rasporeda transfera u polinomijalnom vremenu.

Naš model je predstavljen neusmerenim grafom G=(V,E), gde V predstavlja čvor u mreži koji sadrži spisak datoteka koje treba da prosledi ostalim članovima mreže, dok E je grana čija je težina srazmerna vremenu potrebnom da se datoteka transferuje do drugog čvora.

Kako savremene mreže postaju sve kompleksnije sa ovim problemom se često susrećemo, rad na njegovoj optimizaciji nalazi veliku primenu u praksi. Genetski algoritmi i ostale heurističke metode nalaze primenu u srodnim problemima raspoređivanja.

2 Opis sistema

U ovom delu rada biće prikazan opis sistema, komponenti koje ga čine i način njihove implementacije. Svaki server je pojedinačan objekat koji implementira funkciju za komunikaciju sa drugim serverima u mreži implementiranu na osnovu rada Coffman et al. i pokreće se u zasebnoj niti. Pretpostavke vezane za svaki server su da svaki server salje tačno jedan fajl kroz mrežu, da je protok mreže konstantan, takođe važi pretpostavka da svaki server ima tačno jedan port koji služi za slanje i za primanje sadržaja. Nit završava sa radom onda kada je završila slanje sadržaja. Objekat i dalje postoji tako da nije ometena komunikacija sa drugim serverima.

3 Algoritmi

Za nalaženje što boljeg rasporeda fajlova koji se šalju korisćen je algoritam dat u radu Coffman et al., dok se naša verizija razlikuje u tome što se koriste dve verzije genetskog algoritma sa ciljem bržeg nalaženja takvog rasporeda. U nastavku će biti opisani ovi algoritmi.

3.1 Demand Protocol 1

Mrežu zamišljamo kao povezani graf gde su čvorovi serveri u mreži, a grane predstavljaju fajlove koji se šalju. U nastavku je prikazan algoritam za nalaženje rasporeda slanja fajlova. Q_v predstavlja red koji u sebi sadrži niz zadataka. Svaki zadatak sadrži fajl koji treba poslati i server kome treba poslati taj sadržaj. Server ima listu zadataka koje treba izvršiti. U svakoj iteraciji uzima jedan zadatak i ako ima slobodan port pokušava da pošalje sadržaj drugom serveru. Zatim ukoliko je port slobodan čeka da primi fajl od drugog servera.

Algorithm 1 Demand Protocol 1

```
1: Q_v ← list_of_tasks
2: while Q_v.length > 0 do
3:    v.task = Q_v.pop()
4:    if v.has_an_idle_port() then
5:       u = v.call()
6:       if u! = null then
7:          v.transfer(u)
8:          delete(v.task)
9:       else
10:         Q_v.push(v.task)
11:      end if
12:   end if
13:   if v.has_an_idle_port() then
14:      v.wait()
15:      u = v.received_call()
16:      if u! = null then
17:         u.transfer_task(v)
18:      end if
19:   end if
20: end while
```

U nastavku ćemo primenom genetskog algoritma pokušati da optimizujemo nalaženje što boljeg rasporeda zadataka.

3.2 Genetski algoritam

Svaki genetski algoritam ima nekoliko komponenti koje moraju biti specifikovane, kao što je reprezentacija jedinki, proces selekcije, mutacije, ukrštanje jedinki i pravljenje nove generacije od jedinki koje su učestvovale u ukrštanju. U nastavku će biti opisan način implementaije navedenih koraka u svrhu rešavanja datog problema.

3.2.1 Genetski algoritam prva verzija

Inicijalna ideja bila je da svaki server u mreži sadrži listu fajlova koje treba da prosledi ostalim učesnicima u mreži, kao i promenljivi broj portova preko kojih bi ostali korisnici mogli da uspostave komunikaciju sa datim serverom. U ovoj verziji pokreće se po jedan genetski algoritam za svaki server u mreži. Sledi opis specifičnih komponenti ove verize genetskog algoritma.

Reprezentacija jedinki

Jedan hromozom je lista objekata koji predstavljaju fajlove, a sadrže sledeće informacije:

- ime fajla
- ID servera kome fajl treba da se pošalje

- broj portova servera koji treba da primi fajl
- potrebno vreme da se pošalje fajl

.

Inicijalizacija populacije

Inicijalna populacija je napravljena tako sto se lista koja predstavlja informacije o fajl objektima podeli random indeksom na dva dela i zatim objedini.
Funkcija prilagodjenosti

Funkcija prolagodjenosti je izracunata koriscenjem sledece formule:

$$\sum_{i=0}^{brFajlova} = \frac{brPortovaServeraKomeSeSaljeFajl}{potrebnoVremeZaSlanjeFajla * brFajlova - i}$$

Selekcija, ukrštanje i mutacija

Selekcija je bila turnirskog tipa. Eksperimentalno je određen broj jedinki koje učestvuju u turniru i bira se jedinka sa najvećom vrednosti funkcije prilagođenosti. Eksperimentalno je određen i parametar koji govori o broju jedinki koje učestvuju u reprodukciji. Za ukrštanje je korisceno n poziciono ukrštanje, od dva roditelja nastaje jedno dete. Mutacija se dešava sa verovatnoćom od 0.1 i izvodi se tako što se zamene pozicije dva fajla ukoliko su ispunjeni uslovi za mutaciju.U smeni generacije učestvuje 40% najprilagođenijih jedinki.

3.2.2 Genetski algoritam druga verzija

Glavni nedostatak prve verzije genetskog algoritma bilo je to što se za svaki server u mreži pokretao zaseban genetski algoritam. Takvim razmišljanjem optimizacije na jednoj strani utiču negativno na drugu. Potrebno je bilo težiti globalnoj optimizaciji sistema. Dakle u ovoj verziji se pokreće samo jedan genetski algoritam koji teži da napravi što bolji raspored za slanje fajlova i uključuje u sebe sve servere u mrezi. Sledi opis komponenti ove verzije genetskog algoritma.

Reprezentacija jedinki

Jedan hromozom je skup skih servera u mreži. Serveri sa manjim indeksom će imati prioritet u slanju fajlova. Svaka jednika u hromozomu sadrži:

- indeks
- procenjeno vreme do završetka (ukljucujuci čekanje zbog zauzetih portova)
- listu suseda u mreži
- fajl koji treba da pošalje
- fajl koji treba da primi

Inicijalizacija populacije

Inicijalna populacija je napravljena tako sto se objekti koji predstavljaju servere promešaju u hromozomu na pseudo slučajan način.

Funkcija prilagodjenosti

Vreme koje potrebno celom sistemu da završi slanje fajlove predstavljaće funkciju prilagođenost. Kako se svaki sistem jak onoliko koliko je jaka njegova najslabija karika, to vreme predstavlja vreme završetka rada poslednjeg servera u mreži. Svaki server u jednom rasporedu zadataka procenjuje svoje vreme izvršavanja. Funkcija prilagodjenost jednog hromozoma predstavlja maksimalno vreme izvršavanja pojedinačnog servera. Postupkom obrnutog inžinjeringa Demand Protocol algoritma pokušavamo da procenimo vreme izvršavanja.

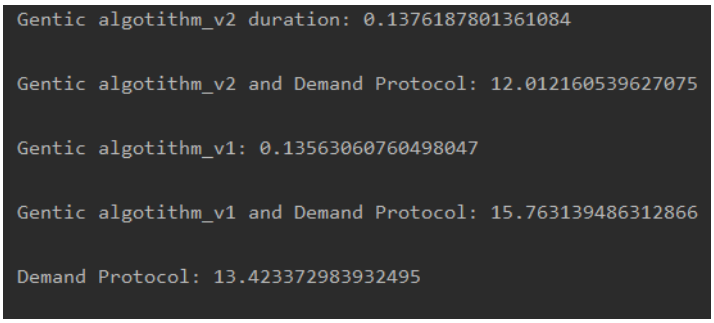
Selekcija, ukrstanje i mutacija

Selekcija je bila turnirskog tipa. Eksperimentalno je odredjen broj jedinki koji učestvuje u turniru i bira se jedinka sa najmanjom vrednosti funkcije prilagođenosti. Eksperimentalno je određen i parametar koji govori o broju jedinki koje učestvuju u reprodukciji. Za ukrštanje je korišćeno ukrštanje zasnovano na permutacijama, kako se ne bi izgubili ili duplirali neki serveri. Od dva roditelja nastaje jedno dete. Mutacijom se menjaju redosledi u hromozomu dva servera.

4 Rezultati

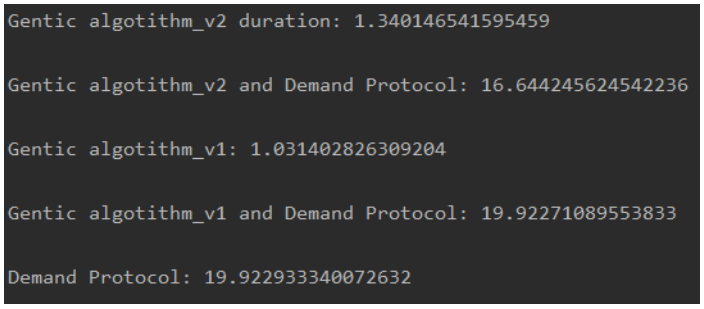
U ovom delu će biti prikazani rezultati izvršavanja sva tri algoritma. Rezultate izvršavanja algoritama ćemo prikazati na različitom broju servera u mreži to jest kada je broj servera 10, 50 i 100. U nastavku će biti prikazani rezultati u sledećoj formi:

- Genetic algorithm_v2: vreme izvršavanja genetskog algoritma, druge verzije
- Genetic algorithm_v2 and Demand Protocol: vreme potrebno sistemu da izvrsi slanje fajlova nakon primene genetskog algoritma
- Genetic algorithm_v1: vreme izvršavanja genetskog algoritma, prve verzije
- Genetic algorithm_v1 and Demand Protocol: vreme potrebno sistemu da izvrsi slanje fajlova nakon primene genetskog algoritma
- Demand Protocol: vreme potrebno sistemu da izvrsi slanje fajlova



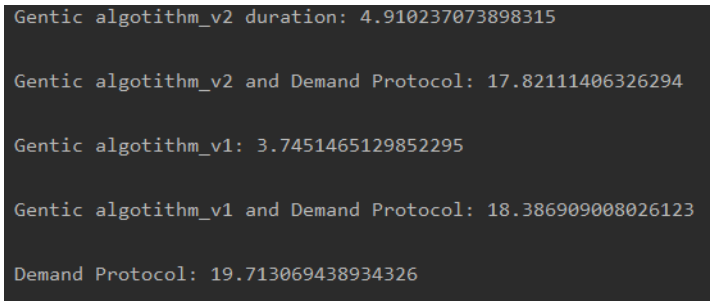
Slika 1: Rezultati izvršavanja sva tri algoritma na 10 servera

Iz priloženih rezultata nad 10 servera možemo zaključiti da se prva verzija genetskog algoritma ponaša najlošije. Zatim vidimo da imamo umereno poboljšanje u odnosu na Demand Protocol algoritam kada primenimo drugu verziju genetskog algoritma.



Slika 2: Rezultati izvršavanja sva tri algoritma na 50 servera

Iz priloženih rezultata nad 50 servera možemo zaključiti da se prva verzija genetskog algoritma ponaša skoro identično kao Demand Protocol algoritam. Zatim vidimo da se druga verzija genetskog algoritma ponaša bolje od ostala dva.



Slika 3: Rezultati izvršavanja sva tri algoritma na 100 servera

Iz priloženih rezultata nad 100 servera možemo zaključiti da se prva verzija genetskog algoritma ponaša malo bolje od Demand Protocol algoritam. Zatim vidimo da se druga verzija genetskog algoritma ponaša bolje od ostala dva.

Treba naglasiti da ovakvo ponašanje nije pravilo za genetski algoritam druge verzije, jer je vreme izvršavanja direktno zavisno od funkcije prilagođenosti.

5 Zaključak

Prilikom rada na ovom problemu, uvidjamo da se algoritam Demand Protocol dat u radu Coffman et al., ponaša sasvim zadovoljavajuće. Uzevši u obzir i lakoću njegove implementacije i pretpostavku da ne zahteva detaljno poznavanje topologije mreže, dolazimo do zaključka da je primena ovog algoritma u realnom svetu sasvim prihvatljiva. Naša prva verzija genetskog algoritma, za koju smo mislili da će dati poboljšanje, zapravo se ponašala slično kao Demand Protocol algoritam, ali mana je to što je potrebno uložiti dodatno vreme za implementaciju ovog pristupa za skromna ili nikakva poboljšanja. Dalje druga verzija genetskog algoritma daje bolje vreme izvršavanja, ali je potrebno uključiti pretpostavke da znamo celu tolopogiju mreže. Takođe finkciju prilagođenosti je teško napisati, a od nje direktno zavisi kvalitet ovog algoritma. Za nastavak istraživanja na ovu temu preporučujemo korišćenje neuronskih mreža za optimizaciju funkcije prilagođenosti.

Literatura

E. G. Coffman, M. R. Garey, and D. S. Johnson. Scheduling file transfers in a distributer network. URL https://github.com/popdav/RI27-MINIMUM-FILE-TRANSFER-SCHEDULING/blob/master/RI-pomocni-rad.pdf.