Elektronski fakultet

Univerziteta u Nišu

Katedra : Veštačka inteligencija

Implementacija genetskog algoritma za rešavanje rasporeda časova na fakultetu

Autor :

Pavle Marinković

Br. Indeksa: 16178

Opis problema

Organizacija rasporeda časova podrazumeva raspoređivanje resursa bez kršenja strogo definisanih pravila. Na primeru fakulteta, treba rasporediti profesore, studente, predmete, amfiteatre po terminima svakoga dana rasporeda, tako da nikada ne dodje do preklapanja. Cilj je stvoriti implementaciju algoritma koji će koristiti unete resurse i vratiti raspored tih resursa sa 100% ispraćenim pravilima i granicama koje jedan raspored na fakultetu nadleže.

Za rešavanje problema rasporeda je primenljivo mnoštvo algoritama, mnogi od njih su korišćeni u komercijalnim implementacijama za rešavanje ovog problema. Neki od njih su npr. “Algoritam bojenja grafa” koji pokušava da dodeli boje (u našem slučaju termine) čvorovima grafa (u našem slučaju časovima) na takav način da dva povezana čvora ne budu iste boje, zatim algoritmi za menadžment databaza koji koriste algoritme za održavanje databaza kako bi generisali vremenski raspored dostupnosti različitih čvorova (servera) unutar databaze.

U ovom radu će biti obrađivan “Genetski algoritam”. Algoritam baziran na Darvinovoj teoriji o evoluciji. Tehnika se bazire na izboru najbolje jedinke (u našem slučaju rasporeda) iz nasumično kreirane populacije hromozoma. Ovaj algoritam se odvija u nekoliko faza, više reči o njima u sledećem poglavlju.

Formulacija problema

Nadalje ću koristiti sledeće vrednosti :

Broj sati u jednom radnom danu fakulteta = 8
Broj amfiteatra = 2
Broj dana u rasporedu = 5
Broj hromozoma u populaciji = 100

Prvo moramo definisati resurse i privalia (granice) kojima algoritam raspolaže.

Resursi :

1- Predavanje (može da traje nekoliko časova)

2- Predmeti

2- Profesori

3- Amfiteatri (ima određeni broj mesta)

4- Studentske Grupe (svaka grupa ima svoju velicinu, broj studenata u njoj)

Pravila:

1- Dva predavanja ne mogu biti u istom amfiteatru u isto vreme

2 - Jedan profesor ne može držati dva predavanja u isto vreme

3- Jedna studenska grupa ne može prisustvovati na više predavanja u isto vreme

4- Amfiteatar mora da ima dovoljno mesta da prihvati sve studentske grupe koje slušaju predavanje u njemu

Za realizaciju algoritma sam koristio python i Visual studio razvojno okruženje. Implementaciju sam pisao od nule bez ikakvih dodatnih biblioteka. O implementaciji će biti više reči na kraju ali bitno je naglasiti da su svi gore navedeni resursi realizovani kao poseban objekat u kontekstu objektno orijentisanom programiranja. Od svih gore navedeno resursa, najbitnije je predavanje.

Konkretno objekat predavanje povezuje sve ostale resurse. Jedno predavanje je definisano predmetom koji se predaje, profesorom koji predaje taj predmet, amfiteatrom u kome se predmet održava i grupama studenata koje prisustvuju predavanju. Algoritam je definisan tako da očekuje listu predavanja a vraća raspored.

Definicija hromozoma

Kako bi primenili genetski algoritam na problem rasporeda časova na fakultetu, prvenstveno moramo definisati osnovnu jedinicu genetskog algoritma (hromozom) u kontekstu našeg problema. Kod nas će jedan hromozom biti celokupan raspored časova.
Na fakultetu je raspored sačinjen od termina. Svaki termin predstavlja sat vremena nekog časa u nekom amfiteatru. To znači da hromozom možemo predstaviti kao niz termina. Ako imamo npr. 8 sati u jednom radnom danu fakulteta, jedan dan rasporeda bi imao 8 termina za svaki amfiteatar. Onda naš niz termina kojim predstavljamo hromoz ima N stavki u sebi, gde je N broj koji se dobija sledećom formulom

N = broj sati u jednom radnom danu * broj amfiteatara * broj dana u rasporedu

Hromozom postaje niž od 80 praznih mesta, u svako mesto možemo smestiti reference ka predavanjima. Koristeći prvobitno definisane vrednosti, u ovom nizu od 80 termina prvih 8 bi bilo za prvi amfiteatar u prvom danu, narednih 8 termina su od drugog amfiteatra u prvom danu, narednih osam bi bilo za prvi amfiteatar u drugom danu itd. Moguće je smestiti i više objekata tipa predavanje u isti termin, posao algoritma je da primeti ovakvo kršenje pravila. Ako predavanje traje dva sata onda će zauzimati dva uzastopna mesta u nizu. Bez zalaženja u implementaciju recimo da uvek znamo gde se u svakom hromozomu nalazi početna pozicija nekog predavanja (u pythonu je ovo odradjeno preko mape koju ima svaki hromozom, u njoj se pamti gde je početni indeks svakog predavanja unutar hromozoma).

Faze genetskog algoritma

Genetski algoritam se odvija u 4 faze.

- Kreiranje inicialne populacije
- Ocenjivanje dobrote hromozoma unutar populacije
- Selekcija i Crossover (preklapanje) hromozoma
- Mutacija hromozoma

Kreiranje inicialne populacije

Za kreiranje inicilane populacije treba izgenerisati 100 hromozoma. Dovoljno je nasumično smestiti predavanja u neki od termina hromozoma. Kako bi smestili jedno predavanje u hromozom treba nam indeks (termin u nizu termina hromozoma u koji želimo da smestimo predavanje). Indeks možemo izračunati na sledeći način:

Indeks = broj sati u jednom radnom danu * ukupan broj amfiteatara * X

+

Broj sati u jednom radnom danu * Y

+

Z

Gde je X = redni broj nasumično izabranog dana, Y = redni broj nasumično izabranog amfiteatra,
Z = nasumično izabran sat u toku radnog dana

Nakon što smestimo predavanje u listu termina na mestu prethodno izračunatog indeksa, treba popuniti i nadalje termine u listi hromozoma, onoliko termina koliko dugo traje predavanje (jedno predavanje može da zauzima više termina u amfiteatru).

Ocenjivanje dobrote hromozoma unutar populacije

Ocenjivanje dobrote hromozoma možemo odrediti tako što ćemo bodovati hromozom. Za svako predavanje ćemo proveriti da li ispunjuje pravila. Ako predavanje ispunjuje neko pravilo dodaćemo jedan poen hromozomu. Na kraju je dobrota hromozoma:

Dobrota = broj poean / (ukupan broj predavanja * ukupan broj pravila)

Algoritam se završava kada nađemo hromozom sa dobrotom 1.

Problem kod ocenjivanja dobrote je odrediti sa kojim predavanjem treba uporediti ono koje trenutno razmatramo. Pošto za svako predavanje znamo gde je njegov početni indeks unutar niza termina, za otkrivanje da li amfiteatar ima dovoljno mesta za to predavanje treba pretvoriti indeks termina u indeks amfiteatra.

Koristeći prvobitno navedene vrednosti za broj sati u radnom danu i broj amfiteatra

Indeks amfiteatra = (indeks termina / 8) mod 2

Sa indeksom amfiteatra možemo indeksirati sve amfiteatre i pročitati broj mesta iz amfiteatra.

Za proveru da li ima više predavanja u istom amfiteatru možemo samo pročitati broj upisanih objekata tipa predavanje u terminu predmeta koji razmatramo. Želimo da postoji samo jedno predavanje u terminu.

Za proveru preklapanja profesora i grupe studenata treba uporediti predavanja u svim terminima koja se desavaju u isto vreme unutar svih amfiteatra u tom danu. (Pogledati source code funkcija "IzracunajDobrotu" za više detalja).

Crossover (preklapanje) hromozoma

Za izbor hromozoma između kojih će se desiti preklapanje koristim elitizam. To znači da algoritam aktivno prati nekoliko najboljih hromozoma (veličina liste najboljih hromozoma je podesiv parameter u implementaciji) i jedan od hromozoma koji će biti izabran za fazu preklapanja pripada ovoj listi najboljih hromozoma. Drugi hromozom se bira nasumično iz cele populacije hromozoma. Preklapanje dva hromozoma se radi kao N-tostruko ukrštanje predavanja. N je podesivi ulazni parametar u implementaciji. Zbog kompleksnosti problema nije moguće koristiti predefinisane granice za ukrštanje, ne bi dobili povoljne rezultate. Granice koje predstavljaju od kog predavanja u nizu predavanja menjamo od kog roditelja uzimamo, se nasumično generišu. Pošto N-tostruko ukrštanje predavanja donosi velike promene unutar hromozoma, a nekada će nam trebati samo male izmene (samo mutacija) dodao sam mogućnost da se crossover faza i ne odigra (postoji šansa za crossover fazu). Ova šansa je takođe podesiv parametar u implementaciji. Ovime smo omogućili da se nekada i samo mutacija odigra, što može dovesti do povoljnijih izmena u velikom hromozomu.

Mutacija hromozoma

Mutacija menja indeks početnog termina na kome se nalazi neko predavanje. To znači da treba iz niza termina sa pozicije starog indeksa i za sve nadalje termine u zavisnosti koliko dugo traje predavanje ukloniti referencu ka predavanju, a zatim dodati referncu na mestu novog indeksa i popuniti te termine u zavisnosti koliko traje predavanje. Izbor predavanja i novi indeks se određuju nasumično. Ova operacija se dešava N puta gde je N podesiv parametar unutar implementacije.

Algoritam ponavlja crossover i mutaciju M puta gde je M podesiv parametar unutar implementacije. Nakon svakog crossover-a i mutacije rezultat je novi hromozom koji ubacujemo u populaciju ali vodeći računa o tome da ne izbrišemo neki hromozom iz grupe najboljih hromozoma. Nakon toga ostaje samo još provera da li možda novonastali hromozom pripada grupi najboljih, čime bi se izbacio najslabiji iz grupe najboljih u slučaju da novi hromozom pripada toj grupi.

O implementaciji

Implementaciju sam radio sam, od nule bez dodatnih biblioteka. Kod je pisan u jeziku Python, korišćena je verzija 3.7.2 Python interpretera. Koristio sam razvojno okruženje Visual Studio 2019. Projekat možete naći na linku repozitorijuma :

https://github.com/StRaToX123/16178_Vestacka_Inteligencija_Projekat_02

Na linku repozitorijuma je postavljen Visual Studio projekat zajedno sa uputstvima kako pokrenuti python visual studio projekat. Ali postoji i druga opcija, zajedno sa ovim izveštajem je dostavljen jedan .py fajl u kome je projekat realizovan, i moguće je pokrenuti direktno njega kroz python okruženje.

U “main” funkciji programa možemo videti sve podesive parametre implementacije

```
if __name__ == '__main__':
    # Ovo su neke od globalnih promenljivih potrebne za rad genetskog algoritma
    brojDanaURasporedu = 5
    brojSatiUDanu = 12
    brojHromozomaUPopulaciji = 100
    brojCrossOverGranica = 2
    brojPermutacijaUJednojMutaciji = 2
    verovatnocaCrossover = 80 # od 100
    verovatnocaMutacija = 3 # od 100
    brojNovihHromozomaUSvakomKoraku = 8
    maxBrojNajboljihHromozoma = 5
```

Program pokreće primer u kome treba rasporediti 24 predavanja. Neka predavanja drže isti profesori, i neka su namenjena za više studentskih grupa. Postoje 4 studentske grupe svaka sa 19 studenata, i postoje dva amfiteatra jedan sa 24 slobodnih mesta a drugi sa 60 kako bi naterali algoritam da smesti neka predavanje sa više studentskih grupa u veći amfiteatar.

Hromozom je implementiran kao posebna klasa i u sebi sadrži funkcije:

1- IzracunajDobrotu

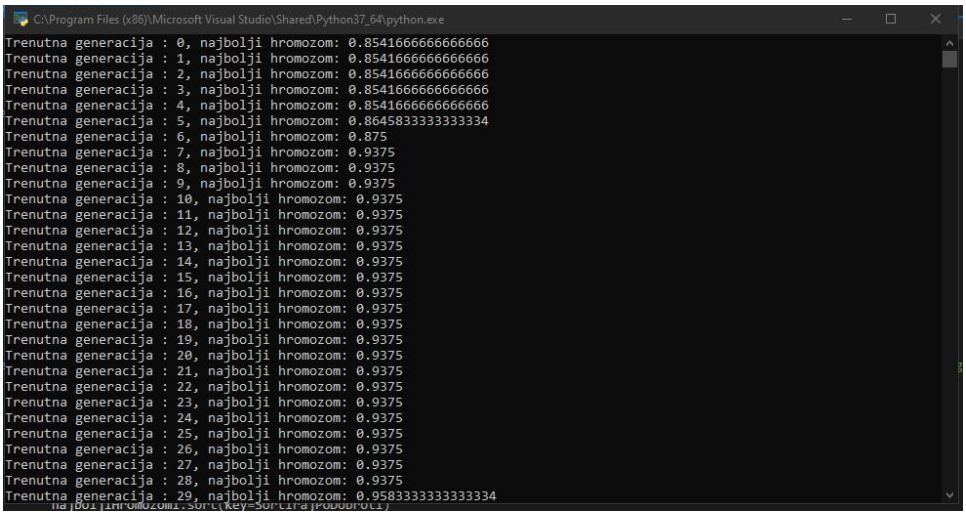
2- Crossover

3- Mutacija

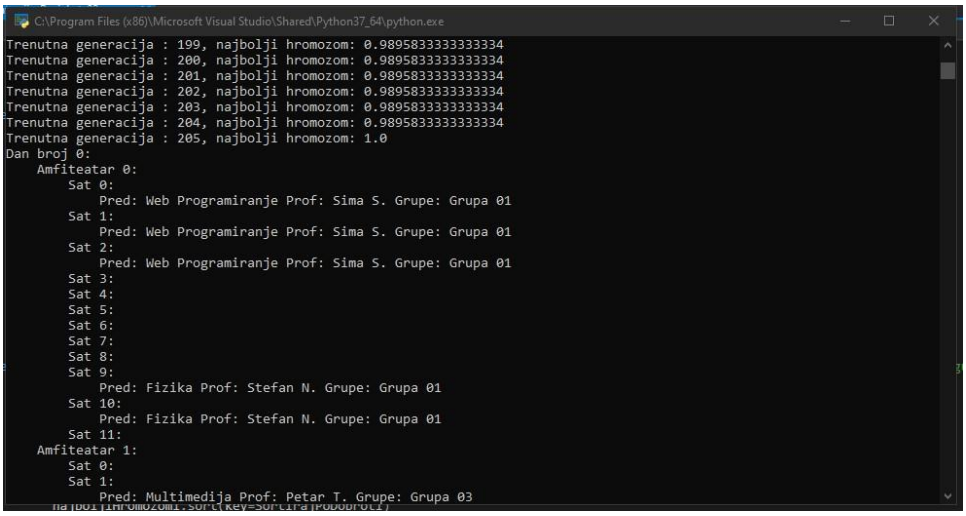
4- OdrediNajboljeHromozome

Sam algoritam, generisanje inicialne populacije i while petlja koja vrši selekciju hromozoma, radi crossover, mutaciju i konstrukciju nove populacije se nalazi u “main” delu programa. Sve dok ne nađe rešenje, program ispisuje na kojoj generaciji se trenutno nalazi (koja je ovo generacija po redu) i kolika je dobrota najboljeg hromozoma unutar trenutne generacije. Program se završava kada bude nađen hromozom sa dobrotom 1.0 i onda ispisuje konačan izgled rasporeda, svaki termin za svaki amfiteatar za svaki dan. Sa ovakvim podešavanjima parametara programa, dobijam dosta povoljne rezultate, potrebno je uglavnom između 200 do 500 generacije pre nego što dođemo do konačnog rešenja.

Za vreme izvršenja :



Na kraju :



Literatura

https://en.wikipedia.org/wiki/Genetic_algorithm

https://towardsdatascience.com/using-genetic-algorithms-to-schedule-timetables-27f132c9e280

Literatura sa kursa Veštačke inteligencije 2020/2021 Elektonskog fakulteta u Nišu