Genetski algoritam za rešavanje problema dodeljivanja mentora studentima

Seminarski rad u okviru kursa Računarska inteligencija

Matematički fakultet, Univerzitet u Beogradu

Vasić Teodora 1/2018

Gojić Tadej 79/2018

Sažetak

U ovom radu je predstavljen genetski algoritam namenjen rešavanju problema dodeljivanja mentora studentima. Za svakog mentora nam je poznat broj studenata koji mu može biti dodeljen, dok za svakog studenta znamo njegovu listu želja. Lista želja nam pokazuje kom mentoru bi student najviše (tom mentoru je dodeljena vrednost 0), odnosno najmanje želeo da bude dodeljen, pri čemu se svim mentorima dodeljuju različite vrednosti (u opsegu od 0 do broj mentora - 1). Cilj genetskog algoritma je da ispuni želje studentima što je više moguće. Kod koji prati rad je dostupan na: https://github.com/tadejgojic99/DodeljivanjeMentoraStudentima

Uvod

Uvođenjem programa studentskog mentorstva na Matematičkom fakultetu, brucošima će studije biti dosta olakšane (u vidu dobijanja saveta u vezi predmeta koje pohađaju od starijih studenata - mentora). Dodela odgovarajućih mentora predstavlja problem dodele resursa. Da bi studenti mentori što bolje mogli da usklade svoje obaveze na fakultetu, ali i da bi mogli što više da se posvete brucošima čiji su mentori, svaki od njih navodi koji je maksimalan broj brucoša kojima bi mogao da pomaže, u skladu sa svojim slobodnim vremenom. Takođe, da bi brucoši bili što zadovoljniji i dobili što sebi sličnije mentore (na osnovu zajedničkih interesovanja i predznanja), svaki od njih kreira listu želja, u kojoj se mentorima dodeljuju različite brojevne vrednosti, pri čemu manja vrednost označava da je mentor bolje pozicioniran na toj listi (0 je najmanja vrednost, dok je broj mentora – 1 najveća). U ovom slučaju mentori nemaju uticaj na to koji će mu brucoši biti dodeljeni, pošto organizatori još uvek nemaju dovoljno informacija o njima na osnovu kojih bi odabir mogao da se vrši.

Ovaj problem predstavlja adaptaciju SPA (Student project allocation) problema, tj. problema dodeljivanja projekta studentima.

SPA

SPA problem se odnosi na dodeljivanje projekata studentima, gde, na odluku o tome koji će projekat biti dodeljen studentu, utiču prosek studenta, kao i njegove želje za date teme. Takođe, na odluku utiče i maksimalan broj studenata koji zajedno može raditi projekat. Problem dodele projekata, tako da svim studentima, u što većoj meri, budu dodeljeni projekti, kao i da studenti sa većom prosečnom ocenom imaju prednost u odabiru, a da pritom ograničenja u vidu maksimalnog broja studenata koji zajedno rade bude ispunjeno, je NP težak problem.

Reprezentacija ulaznih podataka

Ulazni podaci predstavljaju kapacitete mentora i liste želja studenata.

Kapaciteti mentora su prikazani u vidu niza dužine jednake broju dostupnih mentora. I-ta pozicija u nizu predstavlja i-tog mentora i na datoj poziciji se čuva maksimalan broj studenata koje mentor može da prihvati. Mentori se numerišu počevši od 0.

Liste želja studenata se čuvaju u vidu matrice dimenzije broj studenata x broj mentora. U i-toj vrsti se čuva lista želja za i-tog studenta (numerisanje počinjemo od 0), koja je u obliku niza, u kom pozicija označava koja mu je to želja po redu (0 je prva želja), a brojevna vrednost koja se nalazi na toj poziciji predstavlja mentora.

Genetski algoritam

Pri rešavanju ovog problema, genetski kod jedinke predstavlja niz u kom se na prvih n_1 pozicija čuvaju parovi ($student_i$,$mentor_1$), gde je n_1 kapacitet prvog mentora, zatim na narednih n_2 pozicija parovi ($student_i$,$mentor_2$), gde je n_2 kapacitet drugog mentora, i slično, sve dok se svi studenti ne dodele nekome ($student_i$ predstavlja sve studente, bez ponavljanja) .

Takođe, svaka jedinka čuva i kapacitete mentora, liste želje studenata, broj mentora, broj studenata, kao i maksimalan broj studenata koji mogu biti primljeni u program na osnovu kapaciteta mentora.

Fitnes funkcija je suma vrednosti fitnes funkcije u svakom genu.

Svaki gen predstavlja jednog studenta i fitnes u tom genu predstavlja poziciju u listi želja na kojoj se nalazi mentor kom je taj student dodeljen.

U algoritmu se vrši turnirska selekcija. Bira se najbolja od n nasumicno izabranih jedinki, gde je n veličina turnira.

Ukrštanjem jedinki dobijamo dva deteta. Jedno dete se dobija prolaskom sa leve strane kroz oba roditelja, pri čemu se u svakom koraku do sada nedodeljena vrednost gena roditelja sa boljim fitnesom; dok je postupak za drugo dete analogan, s tim što se obilazak vrši sa desne

strane. Nakon toga, dodeljuju se nasumične vrednosti iz skupa nedodeljenih u genima gde vrednost roditeljskih gena nije mogla biti dodeljena. Na samom kraju, poredimo roditelje sa svakim detetom I preživljava najbolji od njih.

Mutacija jedinki se vrši, ukoliko je nasumična vrednost iz intervala [0,1] manja od praga mutacije, zamenom dve vrednosti u genetskom kodu jedinke.

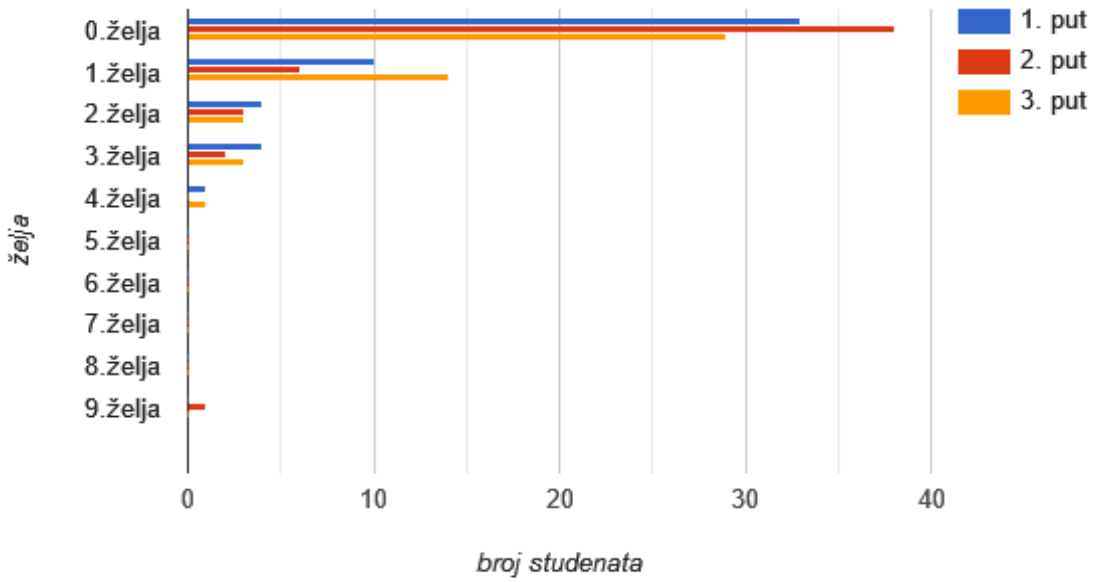
Prvi korak algoritma je generisanje populacije, tj. niza jedinki unapred zadate dimenzije.

Prilikom svake iteracije algoritma, populacija se sortira po fitness-u i u slučaju da je fitness najbolje jedinke 0, svakom studentu je dodeljena prva želja, pa se algoritam zaustavlja. U suprotnom se deo najboljih jedinki čuva, a od ostatka populacije se kreiraju nove jedinke. Turnirskom selekcijom se biraju dve jedinke, od kojih, ukrštanjem dobijamo dve nove jedinke. Zatim se poziva funkcija mutacije za nove jedinke. Mutaciju vršimo da bismo izbegli zaglavljivanje algoritma i nastanak istih jedinki. Nakon toga je potrebno izračunati fitness novodobijenih jedinki.

Dobijeni rezultati

1. primer

Prilikom tri izvršavanja algoritma na istom uzorku od 10 mentora i 50 studenata, gde je najbolji mogući fitnes 0, a najgori 450, dobijeni su sledeći rezultati:



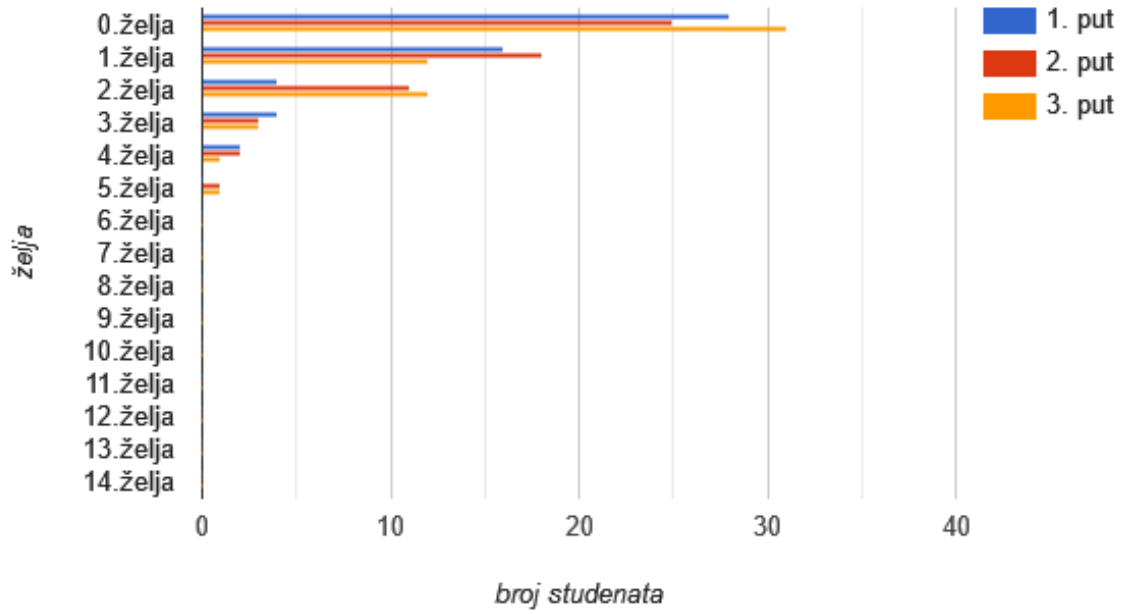
Prilikom prvog izvršavanja 66% studenata je ispunjena prva želja, dok se najgora ispunjena želja nalazi u prvih 40% želja na listi i dodeljena je samo jednom studentu. Fitnes najbolje jedinke je 30.

Prilikom drugog izvršavanja 76% studenata je ispunjena prva želja, dok se najgora ispunjena želja nalazi u prvih 90% želja na listi i dodeljena je samo jednom studentu. Fitnes najbolje jedinke je 27.

Prilikom trećeg izvršavanja 58% studenata je ispunjena prva želja, dok se najgora ispunjena želja nalazi u prvih 40% želja na listi i dodeljena je samo jednom studentu. Fitnes najbolje jedinke je 33.

2. primer

Prilikom tri izvršavanja algoritma, na uzorku sa 15 mentora i 60 studenata, gde je najbolji mogući fitnes 0, a najgori 840, dobijeni su sledeći rezultati:



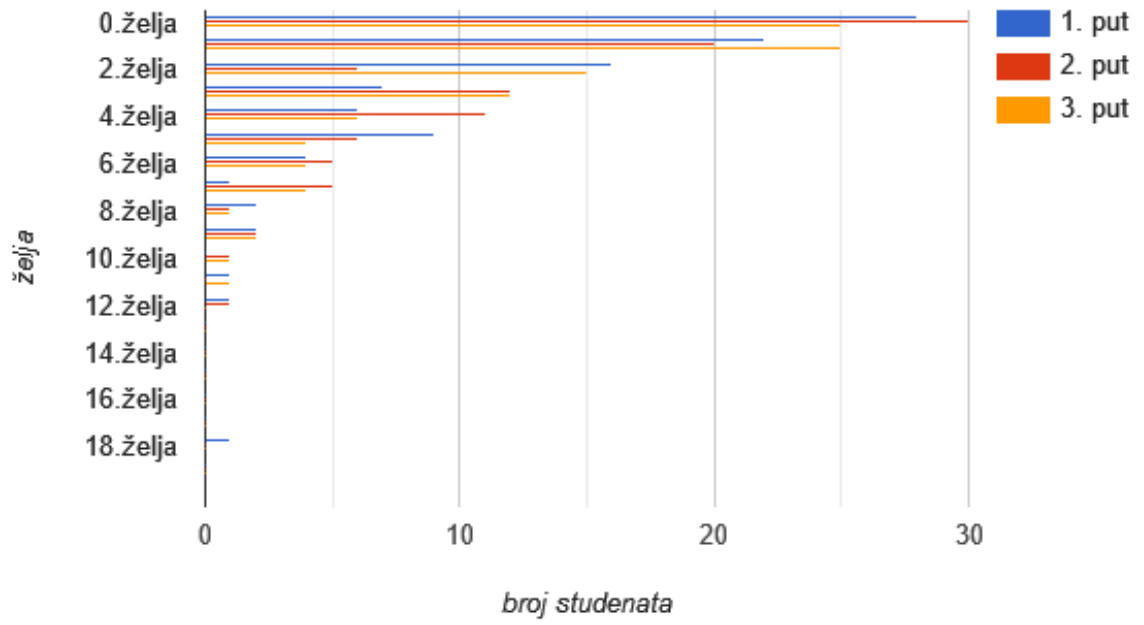
Prilikom prvog izvršavanja 46.67% studenata je ispunjena prva želja, dok se najgora ispunjena želja nalazi u prvih 33.33% želja na listi i dodeljena je samo dva studenta. Fitnes najbolje jedinke je 44.

Prilikom drugog izvršavanja 41.67% studenata je ispunjena prva želja, dok se najgora ispunjena želja nalazi u prvih 33.33% želja na listi i dodeljena je samo jednom studentu. Fitnes najbolje jedinke je 62.

Prilikom trećeg izvršavanja 51.67% studenata je ispunjena prva želja, dok se najgora ispunjena želja nalazi u prvih 33.33% želja na listi i dodeljena je samo jednom studentu. Fitnes najbolje jedinke je 54.

3. primer

Prilikom tri izvršavanja algoritma, na uzorku sa 20 mentora i 100 studenata, gde je najbolji mogući fitnes 0, a najgori 1900, dobijeni su sledeći rezultati:



Prilikom prvog izvršavanja 28% studenata je ispunjena prva želja, dok se najgora ispunjena želja nalazi u prvih 90% želja na listi i dodeljena je samo jednom studentu. Fitnes najbolje jedinke je 250.

Prilikom drugog izvršavanja 30% studenata je ispunjena prva želja, dok se najgora ispunjena želja nalazi u prvih 60% želja na listi i dodeljena je samo jednom studentu. Fitnes najbolje jedinke je 255.

Prilikom trećeg izvršavanja 25% studenata je ispunjena prva želja, dok se najgora ispunjena želja nalazi u prvih 55% želja na listi i dodeljena je samo jednom studentu. Fitnes najbolje jedinke je 234.

4. Primer

Na istom uzorku sa 2 mentora I 4 studenta smo pokrenuli genetski algoritam, brute force algoritam, takođe I pomoću Constraint programiranja dohvatili sva rešenja koja ispunjavaju ograničenja problema, pa ih zatim poredili. Ovo smo ponovili 3 puta I rezultati koji smo dobili su sledeći:

	NAJBOLJE REŠENJE	NAJBOLJA VREDNOST
GENETSKI ALGORITAM	`(1, 0), (0, 0), (2, 1), (3, 1)`	1
BRUTE FORCE	`(3, 0), (1, 0), (0, 1), (2, 1)`	1
POMOĆU CONSTRAINT	`(3, 0), (1, 0), (2, 1), (0, 1)`	1

	NAJBOLJE REŠENJE	NAJBOLJA VREDNOST
GENETSKI ALGORITAM	(3, 0), (0, 0), (1, 1), (2, 1)	1
BRUTE FORCE	(2, 0), (0, 0), (1, 1), (3, 1)	1
POMOĆU CONSTRAINT	(3, 0), (0, 0), (1, 1), (2, 1)	1

	NAJBOLJE REŠENJE	NAJBOLJA VREDNOST
GENETSKI ALGORITAM	(3, 0), (0, 0), (2, 1), (1, 1)	2
BRUTE FORCE	(3, 0), (0, 0), (1, 1), (2, 1)	2
POMOĆU CONSTRAINT	(3, 0), (2, 0), (1, 1), (0, 1)	2

5. Primer

Na uzorku sa 4 mentora I 8 studenata smo pokrenuli genetski algoritam, brute force algoritam, takođe I pomoću Constraint programiranja dohvatili sva rešenja koja ispunjavaju ograničenja problema, pa ih zatim poredili. Dobili smo sledeće rezultate:

	NAJBOLJE REŠENJE	NAJBOLJA VREDNOST
GENETSKI ALGORITAM	(3, 0), (7, 0), (6, 1), (2, 1), (0, 2), (1, 2), (4, 3), (5, 3)	1
BRUTE FORCE	(3, 0), (7, 0), (6, 1), (2, 1), (0, 2), (1, 2), (5, 3), (4, 3)	1
POMOĆU CONSTRAINT	(7, 0), (3, 0), (6, 1), (2, 1), (1, 2), (0, 2), (4, 3), (5, 3)	1

U ovom primeru vreme izvršavanja je:

GENETSKI ALGORITAM	1.078072309494018
BRUTE FORCE	2.418063640594482
POMOĆU CONSTRAINT	4.407788991928101

Prednosti

Korišćenjem prethodno opisanog genetskog algoritma se može veoma brzo (u navedenim primerima su u pitanju bile sekunde) doći do prilično dobrih rezultata. U slučaju ranog zaustavljanja izvršavanja programa, rezultati koji su dobijeni posle par desetina iteracija se mogu iskoristiti.

Mane

Korišćenjem genetskog algoritma nemamo garanciju optimalnosti.

Zaključak

Primena genetskog algoritma za rešenja problema dodeljivanja mentora studentima daje zadovoljavajuće rezltate. Takođe, algoritam se, uz manje adaptacije, može prilagoditi mnogim problemima sličnim ovom (na primer, u slučaju da je u pitanju dodeljivanje mentora studentima druge godine, njihov prosek može imati uticaja na želju koja će im biti

ispunjena. U tom slučaju bi se indeksiranje vršilo od 1, umesto 0 i želje množile sa razlikom maksimalne ocene(10) i proseka studenta.

Rešenje bi se možda moglo poboljšati korišćenjem hibridizacije genetskog algoritma sa simuliranim kaljenjem, koja smanjuje šansu upadanja algoritma u lokalne minimume.

Literatura

A Genetic Algorithm for Allocating Project Supervisors to Students, Hamza O. Salami, Federal University of Technology Minna, Department of Computer Science and Esther Y. Mamman, Federal Inland Revenue Service, Minna, Nigeria

A genetic algorithm for the project assignment problem, Paul R.Harper, Valterde SennaIsrael, T.VieiraArjan K.Shahani