Generiranje rasporeda sati korištenjem genetskog algoritma Uvod u umjetnu inteligenciju

Daniela Džal

Siječanj 2020.

Mentori: Marko Jevtić
Saša Mladenović

Cilj projekta

Cilj ovog projekta je napraviti automatski generator rasporeda na fakultetu korištenjem genetskog algoritma i utvrđivanje parametara za optimalan rad algoritma.

Motivacija

Generiranje rasporeda nastave na fakultetu je kombinatorno optimizacijski problem za koji je dokazano da je NP složenosti [1]. Stvarni problemi ovog tipa su obično jako veliki i kompleksni i ručno traženje rješenja je dugotrajno i mukotrpno. Također, zbog veličine prostora rješenja iscrpni algoritmi pretrage su nepraktični ili čak neizvedivi unatoč sve većoj procesorskoj moći računala [2]. Predloženi su mnogi pristupi i algoritmi iz područja umjetne inteligencije kao npr. hill climbing algoritam, simulated anealing algoritam. Jedan od mogućih pristupa je i upotreba genetskih algoritama koji će ovdje biti istraženi.

Problem generiranja rasporeda

Problem generiranja rasporeda je problem raspoređivanja nastavnih aktivnosti uz zadovoljavanje ograničenja. U slučaju ovog projekta postavljena su sljedeća ograničenja:

- u svakoj se dvorani u nekom trenutku može odvijati samo jedan kolegij
- dodijeljena dvorana mora biti adekvatno opremljena za održavanje dodijeljenog kolegija (npr. računalna oprema)

- nastavnik istovremeno može održavati nastavu iz najviše jednog kolegija
- svaki student istovremeno može prisustvovati najviše jednom kolegiju

Genetski algoritmi

Genetski algoritmi simuliraju evoluciju u prirodi [3]. Kako se selekcijom, križanjem i mutacijom pojedine vrste u prirodi prilagođavaju okolini u promjenjivim uvjetima, tako u računalima genetski algoritmi pokušavaju od grube aproksimacije rješenja kroz više generacija doći do sve bolje aproksimacije rješenja. U jednostavnom genetskom algoritmu svaka jedinka predstavlja potencijalno rješenje problema kojeg obrađujemo. Svaka jedinka u populaciji predstavljena je jednakom podatkovnom strukturom i definirani su genetski operatori kojima dobivamo sljedeću generaciju - selekcija, križanje i mutacija. Također, jasno je definirana funkcija "dobrote" ili funkcija "troška". Te funkcije svakoj jedinki pridjeljuju broj kojim opisuju kvalitetu te jedinke. Selekcija, slično prirodnoj selekciji osigurava da bolje jedinke (bolje jedinke su one koje imaju veću vrijednost funkcije dobrote, odnosno manju vrijednost funkcije troška) imaju veću šansu za prenošenje svojstava. Kao što u prirodnoj selekciji bolje jedinke imaju veću šansu za reprodukciju tako selekcija u genetskom algoritmu osigurava da će "dobra" svojstva biti proslijeđena sljedećoj generaciji tj. sljedećem ciklusu algoritma. Križanjem se prenose svojstva roditelja na djecu dok se mutacijom slučajno mijenja neki gen jedinke. Selekcija i križanje osiguravaju da se jedinke u svakoj sljedećoj generaciji bolje prilagođavaju uvjetima tj. konvergiraju prema optimumu dok mutacija osigurava da algoritam ne "zapne" u nekom lokalnom optimumu.

Implementirani algoritam

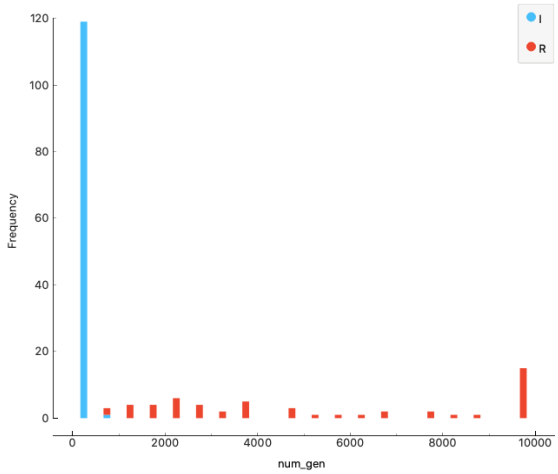
U radu `https://github.com/NDresevic/timetable-generator` je promatran problem generiranja rasporeda sati na fakultetu. Rješenje je pronađeno upotrebom (1+1) evolucijske strategije [4], tj. iz jedne jedinke "roditelja" mutacijom je dobiveno jedno "dijete". Uz korištenje već implementiranih struktura podataka i pomoćnih funkcija na taj projekt je dodan algoritam sa drukčijom evolucijskom strategijom. Spomenuti projekt je odabran za bazu ovog projekta jer je ciljani problem bio slično strukturiran te zbog jednostavnosti nadogradnje. Npr. bez većih promjena u kodu se može umjesto studijskih grupa unijeti popis upisanih studenata za pojedini kolegij (konkretan problem na PMFST-u zbog mnoštva izbornih predmeta), a onda kasnije i dobiti raspored sati za pojedinog studenta. U tom slučaju, bilo bi zanimljivo, osim adekvatne opreme, promatrati i kapacitet učionica.

Prikaz rješenja

Svaka jedinka u početnoj populaciji je jedno potencijalno rješenje problema. U ovom slučaju, uzeto je rješenje na način da su zadovoljena samo ograničenja vezana uz prostor. Dakle, ne može se dogoditi da je više predmeta zakazano u istoj učionici u isto vrijeme i zakazane učionice su po opremi odgovarajuće za predmet. U početnoj populaciji jedinke ne zadovoljavaju nužno ograničenja vezana uz profesore ni studente, odnosno, može se dogoditi da u nekom rasporedu u početnoj populaciji jedan profesor ili grupa studenata mora biti na više mjesta u isto vrijeme. Upravo brojem kršenja tih ograničenja je definirana funkcija troška svakog rasporeda. Svaka jedinka (raspored) u populaciji ima strukturu rječnika tako da je ključ ID predmeta, a vrijednost je par (vrijeme, učionica) koji je na slučajan način pridjeljen tom predmetu.

Evolucijska strategija

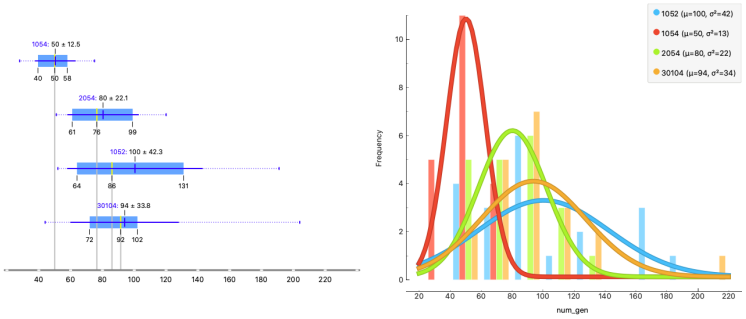
Veličina populacije je prvi parametar koji je bitno na dobar način izabrati kako bi algoritam bio što učinkovitiji.
Prvi korak algoritma nakon generiranja populacije je selekcija i ovdje je korištena eliminacijska selekcija. Svakoj jedinki populacije je izračunata vrijednost funkcije troška, a onda je od M jedinki eliminirano n jedinki i to na način da je vjerojatnost eliminacije proporcionalna vrijednosti funkcije troška.
Nakon što je dovršen postupak eliminacije, potrebno je reprodukcijom (križanjem) nadomjestiti izbrisane jedinke. Sada se n puta na slučajan načun biraju parovi jedinki iz preostale populacije i njihovim križanjem se dobivaju potomci koji se dodaju populaciji. Korišteno je križanje s jednom točkom prekida.
Treći korak genetskog algoritma je mutacija. Na slučajan način se bira jedinka i mjesto gena koji će mutirati (u našem kontekstu to znači da biramo predmet i mijenjamo mu par (vrijeme, učionica). Implementirane su dvije metode mutacije. U prvom slučaju mutira nasumični gen, tj. na slučajan način biramo predmet koji će opet na slučajan način promijeniti vrijeme i mjesto održavanja. U drugom slučaju tražimo predmet koji krši najviše ograničenja i pokušavamo ga smjestiti na mjesto gdje uopće ne krši ograničenja. Uočeno je da algoritam puno ranije dolazi do rješenja ako ne mutira slučajan gen nego onaj koji krši najviše ograničenja.



Slika 1: Broj generacija potrebnih za pronalazak rješenja uz mutaciju slučajnog gena i specijaliziranu mutaciju. Mutacija slučajnog gena je označena slovom R, odnosno crvenom bojom, a mutacija koja postavlja predmet na idealno mjesto označena je s I, tj. plavom bojom

Odabir parametara

Da bi performanse algoritma bile optimalne, nužno je naštimati parametre. To se kod genetskih algoritama može isključivo eksperimentalnim putem, a parametri su veličina populacije, broj jedinki za eliminaciju i broj mutacija. Na dva različita skupa podataka je pokrenut algoritam s različitim parametrima (svaka trojka parametara 10 puta). Pritom je korištena mutacija koja postavlja predmeta na idealno mjesto na kojem zadovoljava sva ograničenja. Uspoređujemo u koliko generacija algoritam dolazi do rješenja.



(a) Box plot (b) Distribucije

Slika 2: Broj generacija za postizanje rješenja u ovisnosti o odabranim parametrima

Na grafu vidimo da algoritam na promatranim podacima najprije (u najmanje generacija) dolazi do rješenja za parametre 1054 (veličina populacije je 10, eliminira se 5 jedinki i događaju se 4 mutacije u svakoj generaciji). Osim što za promatrane parametre imamo najmanji medijan broja generacija, i varijanca je najmanja.

Zaključak

Usporedimo li vrijeme izvođenja promatranih genetskih algoritama, jasno je da je algoritam s (1+1) evolucijskom strategijom brži. S druge strane, i algoritam s većom populacijom daje dobar rezultat u razumnom vremenu, a imamo manji rizik od zapinjanja u lokalnom optimumu s obzirom da počinjemo od veće početne populacije. Iako zauzima puno više memorije i zbog toga je znatno sporiji, algoritam s većom populacijom pokriva veći prostor rješenja.
U specifičnom slučaju kada je okupiranost pojedinog tipa dvorane velika, evolucijska strategija koja koristi samo mutaciju bi mogla imati problem s obzirom da nema gdje premjestiti predmete koji krše velik broj ograničenja. S druge strane, inicijalno stvaranje veće populacije bi trebalo u startu ponuditi više djelomičnih rasporeda za popunjavanje tih učionica.

Literatura

[1] S. Sutar and R. Bichkar, "Parallel genetic algorithm for high school timetabling," *International Journal of Computer Applications*, 2017.

[2] M. Čupić, *Raspoređivanje nastavnih aktivnosti evolucijskim računanjem.* PhD thesis, Fakultet elektrotehnike i računarstva, 2011.

[3] M. Golub, *Genetski algoritam, skripta 1. dio.* 1997.

[4] M. Čeri and I. Malović, "Evolucijske strategije." FER, 2007.