

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

SEMINAR

Primjena genetskog algoritma na problem rasporeda sati

Mladen Jurković

Voditelj: *Domagoj Jakobović*

Zagreb, travanj, 2008.

Sadržaj

1. Uvod	2
2. Genetski algoritmi	3
2.1. Selekcija	3
2.2. Genetski operatori.....	4
2.2.1. Križanje	4
2.2.2. Mutacija.....	5
3. Problem rasporeda sati.....	6
4. Primjena genetskog algoritma na problem rasporeda sati	7
4.1. Prikaz rješenja.....	7
4.2. Selekcija	7
4.3. Križanje	8
4.4. Mutacija.....	8
4.5. Dobrota jedinke	8
5. Zaključak.....	9
6. Literatura.....	10
7. Sažetak	11

1. Uvod

Otkako je znanost i tehnologija napredovala, znanstvenike sve više zanima mogućnost stvaranja umjetne inteligencije. Preduvjet za umjetnu inteligenciju je sposobnost rješavanja problema. Pitanje je kako isprogramirati računalo da samostalno rješava probleme.

Jedini poznati proces koji je stvorio inteligenciju je evolucija. Čovjek kao izumitelj ne može se mjeriti s evolucijom, može samo učiti iz nje i pokušati ju oponašati. Evolucija je uspjela stvoriti jedinke izvrsno prilagođene uvjetima na Zemlji. Ako se evolucija prikaže kao algoritam kojem je cilj stvoriti dobre jedinke (dobra jedinka u ovom kontekstu znači jedinka prilagođena danim uvjetima), može se reći da je evolucija vrlo uspješan algoritam. Evolucija je stvorila različita bića koja su, svako biće na svoj način, "dobra". Evoluciju živih bića opisao je engleski prirodoslovac Charles Darwin u knjizi "Postanak vrsta" (1859.) Po njegovoj teoriji najsposobnije jedinke preživljavaju i prenose svoj genetski materijal na potomke. Svaka jedinka je u potpunosti određena svojim genima.

Postoje nekoliko algoritama koji su zasnovani na evoluciji živih bića. Jedna skupina su i genetski algoritmi. Počeci genetskih algoritama sežu u rane 70-te godine prošlog stoljeća kada ih je prvi opisao John H. Holland. [3]

Genetski algoritam (GA) je heuristička metoda optimiranja koja imitira evoluciju postupcima selekcije, križanja i mutacije. Koristi se za rješavanje NP problema, problema koji nisu polinomske složenosti i često imaju skupinu jednako dobrih rješenja. Područje pretrage kod NP problema je ogromno, tako da čak ni sadašnji kompjuteri ne mogu pronaći rješenje problema u realnom vremenu. GA traži optimalno rješenje ili skupinu rješenja. Svakim pokretanjem GA može naći različita rješenja. [1]

U ovom radu pokazuje se primjena GA na problem rasporeda sati. U drugom poglavlju opisan je genetski algoritam, princip njegovog rada i postupci kojima stvara moguća rješenja problema. U trećem poglavlju je opisan problem rasporeda sati. U četvrtom poglavlju opisana je već postojeći genetski algoritam koji rješava problem rasporeda sati. [2]

2. Genetski algoritmi

Svaki genetski algoritam može se podijeliti u nekoliko koraka. Algoritam počinje stvaranjem populacije jedinki, gdje je broj jedinki proizvoljan. Svaka jedinka označava jedno od mogućih rješenja problema. Jedinka ima svoju genetsku strukturu koja se može prikazati na različite načine. Najjednostavniji je binarni prikaz u kojemu se jedinka prikazuje nizom nula i jedinica generiranih slučajnim odabirom. Za svaku jedinku mora se moći reći koliko je "dobra". Zato se u GA definira faktor dobrote, tj. ocjena kojom algoritam procjenjuje jedinku. Faktor dobrote je funkcija definirana od strane čovjeka. Mogućnost ocjenjivanja jedinki daje algoritmu samostalnost u radu.

Početna populacija se mijenja postupcima selekcije, križanja i mutacije. Jedna populacija u nekom trenutku naziva se generacija. Izmjenom generacija jedinke postaju sve "bolje" (boljeg faktora dobrote) i postaju zadovoljavajuća rješenja problema. Broj generacija je parametar kojeg određuje korisnik. Umjesto toga korisnik može definirati uvjet završetka evolucijskog procesa. Ako se stvorilo generacija koliko je zadano ili je zadovoljen uvjet završetka algoritma, algoritam staje i ispisuje najbolje nađeno rješenje.

```
Genetski_algoritam
{
    t = 0
    generiraj pocetnu populaciju potencijalnih rjesenja P(0);
    sve dok nije zadovoljen uvjet završetka evolucijskog procesa
    {
        t = t + 1;
        selektiraj P'(t) iz P(t-1);
        križaj jedinke iz P'(t) i djecu spremi u P(t);
        mutiraj jedinke iz P(t);
    }
    ispiši rješenje;
}
```

Slika 2.1.: Struktura genetskog algoritma

Na slici 2.1. prikazana je struktura genetskog algoritma. Algoritam na početku stvori početnu populaciju, stvara nove jedinke sve dok se ne zadovolji uvjet završetka evolucijskog procesa. Postupci kojima GA stvara nove jedinke su objašnjeni u nastavku.

2.1. Selekcija

U evoluciji selekcija je proces kojim najsposobnije jedinke opstaju i prenose svoj genetski materijal dalje, kroz potomke. Isto tako kod genetskog algoritma samo najsposobnije jedinke, najbolja rješenja problema opstaju.

Postoji nekoliko vrsta selekcije. To su jednostavna, turnirska i eliminacijska selekcija. Kada se govori o selekciji značajan je i svojstvo elitizam koji je opisan kasnije kod primjene GA.

Jednostavna selekcija je selekcija kod koje svaka jedinka ima vjerojatnost pojavljivanja u sljedećoj generaciji, a ta je vjerojatnost proporcionalna s faktorom dobrote te jedinke. Nedostatak ove selekcije je mogućnost pojavljivanja iste jedinke dva ili više puta u sljedećoj generaciji što usporava algoritam.

Turnirska selekcija je selekcija kod koje slučajnim odabirom izabiremo k jedinki, svaku s jednakom vjerojatnošću, i izdvajamo najbolju od tih jedinki. Na tu jedinku djelujemo genetskim operatorima. Eliminacijska turnirska selekcija je vrsta turnirske selekcije kod koje se isto tako odabire k jedinki, najlošija se eliminira i zamjenjuje „djetetom“ dviju (opet slučajno odabranih) jedinki. Turnirska selekcija će biti još detaljnije opisana kasnije kod primjene GA.

Eliminacijska je selekcija slična jednostavnoj selekciji. Razlika je što se ne odabiru bolje jedinke nego one lošije te se za svaku jedinku definira funkcija kazne koja je obrnuto proporcionalna funkciji dobrote. Loše jedinke se zamjenjuju novima nastalim nakon križanja preostalih jedinki. Ova selekcija izbjegava stvaranje duplikata što je bitno za brzinu izvršavanja GA.

2.2. Genetski operatori

Genetski operatori mogu biti unarni, koji djeluju na jednu jedinku i binarni, koji djeluju na dvije jedinke odjednom. Jedinka koja nastaje djelovanjem binarnog operatora naziva se dijete. Operator koji bi koristio više od dvije jedinke odjednom može se prikazati kao binarni primjenjen na dvije jedinke od kojih se dobiva dijete, pa opet kao binarni operator na djetetu i još jednoj jedinki, i tako dok se ne bi iskoristile sve jedinke. Genetski operatori koji se koriste u GA su križanje i mutacija. Svaki genetski operator ima vjerojatnost primjene.

2.2.1. Križanje

Križanje u evoluciji je proces kod kojeg se genetski materijal roditelja miješa i nastaju djeca s obilježjima i jednog i drugog roditelja. Genetski materijal nalazi se u kromosomima i stoga se pojedina jedinka u GA naziva i kromosom.

Križanje je binarni operator koji djeluje na dvije jedinke i stvara novu jedinku, dijete. Dijete može imati i manji faktor dobrote od roditelja, ali dugotrajno gledajući križanje stvara sve bolje i bolje jedinke. Križanje ovisi o prikazu jedinki. Križanja se mogu podijeliti na uniformno, s jednom ili dvije točke prekida, segmentno i miješajuće.

Kod binarnog prikaza jedinki križanje stvara novu jedinku tako da slučajnim odabirom uzima bit po bit iz jednog ili drugog roditelja i stvara dijete. Opisano križanje naziva se uniformno križanje jer svaki bit roditelja ima jednaku vjerojatnost da bude ugrađen u dijete. Koristi se za genetske algoritme koji nemaju veliku populaciju, pa je kromosom jedinki sastavljen od malo bitova.

Kod križanja s jednom ili dvije točke prekida, definiraju se točke prekida kromosoma. Uzima se dio kromosoma jednog od roditelja do točke prekida i nastavlja s drugim roditeljom od točke prekida pa do druge točke i tako do kraja kromosoma. Segmentno je križanje s više točaka prekida, kod kojeg se prekidi i dio kromosoma kojeg roditelja se određuju slučajno.

Miješajuće je križanje je križanje koje se sastoji od tri dijela. Prvo se izmješaju bitovi roditelja, onda se koristi križanje s jednom ili dvije točke prekida i nakon toga bitovi roditelja vrate se na početak kako bi roditelji ostali neizmjenjeni.

2.2.2. Mutacija

Mutacija u evoluciji je pojava kod koje se mijenja genetski materijal jedinki. Mutacija je unarni operator, djeluje nad jednom jedinkom i ta promjenjena jedinka nastavlja živjeti u novoj generaciji, ali s izmjenjenim genetskim materijalom. Razlikujemo nekoliko vrsta mutacija: jednostavnu mutaciju, miješajuću mutaciju i invertirajuću miješajuću mutaciju. Kod jednostavne mutacije svakom bitu se pridjeljuje određena vjerojatnost mutacije, koja za binarni prikaz nulu mijenja u jedinicu i obratno.

Miješajuća mutacija je ona kod koje se odabire slučajni kromosom, početak i kraj mutiranja i unutar tih granica kromosom se mutira tako da se izmješaju geni ili generiraju ponovno slučajno. Invertirajuća miješajuća mutacija jednako kao miješajuća odabire granice, ali unutar tih granica invertira sve bitove. Ako bi sve jedinke populacije imale na nekom bitu jedinicu (za primjer), ta se jedinica ne bi mogla promijeniti samo križanjem i zato mutacija ima ulogu obnavljanja izgubljenog genetskog materijala. Potencijalno dobro rješenje ima manju vjerojatnost biti izgubljeno ako koristimo mutaciju.

3. Problem rasporeda sati

Stvaranje rasporeda sati zahtijeva koordinaciju sljedećih elemenata: učenika, predavača, predavaonica i trajanja predavanja. Skupina učenika svrstava se u grupe koje se nazivaju razredi (npr. razred 7b).

Problemi koji se javljaju su mnogobrojni. [4]

- Neki predmeti zahtijevaju “blok-satove”, dva predavanja jedno za drugim. To je osobito važno za predmete kao što su kemija i fizika čiji pokusi mogu potrajati i duže od vremena predviđenog za jedno predavanje.
- Bitno je dodijeliti predavača nekom razredu; budući postoje različiti predavači koji mogu predavati isti predmet nekom razredu, mora se odlučiti koji će predavač predavati dotičnom razredu.
- Pridjeljivanje predavaonica predmetu je podjednako bitno. Predmet kemija može biti održan samo u predavaonici koja ima aparaturu za pokuse.
- Mora se paziti da se ne prekorači broj predavanja nekog predmeta u jednom tjednu. (na primjer tijekom jednog tjedna mora biti četiri predavanja iz matematike)
- Pojedini predavači nisu dostupni određenim danima, i zato se moraju uračunati samo u njima slobodne dane.
- Ponekad određene škole dijele prostorije. Njihovi rasporedi sati moraju biti koordinirani kako ne bi došlo do preklapanja.
- Moguće je da se razred dijeli u dvije grupe jer je dio učenika izabrao predavanja iz različitih predmeta (npr. dio sluša njemački jezik, dio engleski jezik).

Navedeni problemi se gomilaju za posebnosti svake pojedine škole, pa se rješavanje problema mora moći vrlo brzo i jednostavno prilagoditi. Prilagođivanje je jedna od karakteristika genetskog algoritma. Primjena genetskog algoritma na problem rasporeda sati opisana je u sljedećem poglavlju.

4. Primjena genetskog algoritma na problem rasporeda sati

Za opis genetskog algoritma koji rješava problem rasporeda sati koristi se već postojeća implementacija. U toj implementaciji zahtjevi koji se postavljaju prilikom izrade rasporeda mogu se podijeliti u dvije kategorije: strogi i blagi zahtjevi. Razlika između zahtjeva je u tome što kršenjem strogih zahtjeva raspored sati postaje neupotrebljiv, dok blagi zahtjevi služe za ocjenu kvalitete rasporeda. [2]

Strogi zahtjevi su sljedeći:

1. Profesor ne može predavati dvama razredima odjednom
2. Razred ne može imati dva predavanja odjednom
3. U jednoj prostoriji se može istovremeno održavati samo jedno predavanje
4. Prostorija mora zadovoljiti zahtjeve predmeta
5. Prostorija mora biti dovoljno velika
6. Prostorija ne smije biti prevelika

Neki primjeri blagih zahtjeva su sljedeći:

1. Predavanja istog kolegija/predmeta ne smiju biti istog dana
2. Predavanja istog kolegija/predmeta ne smiju biti sljedeći dan

4.1. Prikaz rješenja

U opisanoj implementaciji kromosomi su predstavljeni kao dvodimenzionalne matrice pokazivača na profesore. Svaki red matrice odgovara rasporedu jednog razreda. Time je implicitno riješen strogi zahtjev da jedan razred ne može istovremeno slušati dva profesora. Na slici 4.1. dan je primjer nekog kromosoma u rasporedu s tri profesora, tri razreda, dva dana i tri predavanja dnevno.

	uto			sri		
razred a	1	2	3	3	2	1
razred b	3	3	2	1	1	2
razred c	2	1	1	2	3	3

prof Perić – 1, prof Milić – 2, prof Ivić – 3

Slika 4.1. : prikaz kromosoma (dvodimenzionalna tablica)

Stupac dvodimenzionalne tablice predstavlja određeni sat određenog dana u tjednu. Stupac u crvenom predstavlja drugi sat utorka. Svako ponavljanje profesora u stupcu kršenje je strogog zahtjeva jer isti profesor ne može biti u isto vrijeme na dva različita mjesta.

4.2. Selekcija

U postupku odabira koristi se turnirska selekcija. Od k slučajno odabranih jedinki uzimaju se dvije najbolje i njihovim križanjem se stvaraju nove jedinke. Vrlo je bitan i

elitizam. Najbolje jedinke imaju šansu da postanu promijenjene, i zato se one zaštićuju i u potpunosti jednake prelaze u novu generaciju. Elitizam osigurava da najbolje jedinke prenose svoj genetski materijal dalje kako bi i kasnije sudjelovale u križanju i stvorile potencijalno bolja rješenja.

4.3. Križanje

Križanje se izvodi s vjerojatnošću $P(\text{križanje})$, inače se kao dijete uzima roditelj koji ima veći faktor dobrote. Samo križanje se izvodi tako da se iz kromosoma prvog roditelja za svaki razred sa vjerojatnošću 0.5 uzima dio dvodimenzionalne tablice koji predstavlja zapis rasporeda za taj razred, inače se uzima iz drugog roditelja.

4.4. Mutacija

Mutacija se izvodi na način da se svako predavanje s vjerojatnošću $P(\text{mutacija})$ mijenja sa nekim nasumično odabranim ako nakon zamjene nema preklapanja. Svako predavanje algoritam će na opisan način pokušati nekoliko puta zamijeniti s nekim drugim predavanjem.

4.5. Dobrota jedinke

Dobrota se određuje na temelju broja preklapanja i na temelju kazni koje je kromosom zaslužio jer ne zadovoljava određene blage zahtjeve. Točna formula kojom se dobiva dobrota je $1/(1+BSP+BZP/DIV)$. BSP označava broj preklapanja u rasporedu odnosno broj prekršenih strogih zahtjeva. BZP označava kaznu zbog prekršaja blagih zahtjeva, DIV označava djelitelj koji određuje koliki utjecaj na dobrotu imaju blagi zahtjevi. U idealnom slučaju bi zbroj $BSP+BZP/DIV$ bio jednak nula pa se zbog toga u nazivnik dodaje 1. Prema tome, za idealan raspored bi dobrota bila jednaka 1. Kod onih ne idealnih, ali zadovoljavajućih rješenja, faktor dobrote varira između 0,5 i 0,85. Za broj generacija oko 10000 faktor dobrote jedinki teži prema 0,85.

5. Zaključak

Genetski algoritam služi za rješavanje NP složenih problema. U seminarskom radu pokazana je primjena genetskog algoritma na problem rasporeda sati. Složen problem rasporeda sati uspijeva se riješiti primjenom genetskog algoritma. U prethodno opisanoj implementaciji genetski algoritam uspijeva pronaći zadovoljavajuća rješenja u malom broju generacija, oko 400. Povećanjem broja generacija s 400 na 10000, rješenja postaju idealnija. Vrijeme rada genetskog algoritma za oko 400 generacija je otprilike 7 sekundi, a za 10000 generacija otprilike 3 minute. Zadovoljavajuća rješenja su ona koja nemaju kršenje strogih zahtjeva, u rasponu broja generacija od 400 do 10000 stvaraju se rješenja koja nemaju kršenje strogih zahtjeva, razlika je u broju prekršenih blagih zahtjeva. Kod manjeg broja generacija broj prekršenih zahtjeva je oko 90, za veći broj generacija broj blagih prekršaja smanjuje se na otprilike 40. Koliko je to prihvatljivo to se ostavlja korisniku da ocijeni. Genetski i srodni algoritmi počeci su stvaranja strojeva koji mogu samostalno rješavati raznolike probleme.

6. Literatura

- [1] Golub M., Genetski algoritam, Prvi dio, 27. rujna 2004., verzija 2.3
- [2] Bedek V., Stvaranje rasporeda sati genetskim algoritmom, (predmet Projekt) 2008.
- [3] Wikipedia, the free encyclopedia, Genetic algorithm , 15. veljače 2008.
http://en.wikipedia.org/wiki/Genetic_algorithm
- [4] Wikipedia, the free encyclopedia, Timetabling , 20. travnja 2008.,
http://en.wikipedia.org/wiki/School_timetable

7. Sažetak

Postoje složeni problemi koji zahtijevaju mnogo uloženog truda i vremena kako bi se riješili. Za rješavanje takvih problema služe različiti algoritmi. Jedan od njih je i genetski algoritam. Rješava skoro u potpunosti samostalno, imitirajući evoluciju, takve probleme.

U radu su opisana načela rada i konstrukcije genetskog algoritma. Od problema koji se mogu riješiti genetskim algoritmom opisan je problem rasporeda sati i njegovo rješavanje genetskim algoritmom.