UNIVERZITET U BEOGRADU

MATEMATIČKI FAKULTET

Jasmina Fijuljanin

GENETSKI ALGORITAM ZA REŠAVANJE UOPŠTENOG PROBLEMA BOJENJA GRAFA SA OGRANIČENJIMA ŠIRINE OPSEGA I NJEGOVA PRIMENA U NASTAVI

Diplomski – master rad

Beograd

2010.

Mentor: **Doc dr Vladimir Filipović**

Matematički fakultet

u Beogradu

Komentor: **dr Jozef Kratica**

Viši naučni savetnik

Matematički institut SANU

Članovi komisije: **dr Aleksandar Savić**

Matematički fakultet

u Beogradu

Datum odbrane: _____

Genetski algoritam za rešavanje uopštenog problema bojenja grafa sa ograničenjima širine opsega i njegova primena u nastavi

Rezime

U ovom radu je opisan genetski algoritam za rešavanje uopštenog problema bojenja grafa sa ograničenjima širine opsega. Ovaj problem je uopštenje poznatog **NP**-teškog problema bojenja grafa i ima značajnu primenu u praksi. Koristi se pri rešavanju problema dodeljivanja frekvencija kod mobilnih i radio mreža, optimizacije prevoza opasnih materija, kao i u obrazovanju prilikom pravljenja rasporeda polaganja ispita na fakultetima, rasporeda kontrolnih i pismenih zadataka za mesec dana, polugodište ili za celu školsku godinu, rasporeda korišćenja video bima, projektora i drugog školskog pribora u školama.

Pri rešavanju uopštenog problema bojenja grafa korišćeno je specijalno projektovano celobrojno kodiranje rešenja i genetski operatori koji su prilagođeni prirodi rešavanog problema: fino-gradiranu turnirsku selekciju, jednopoziciono ukrštanje, prostu mutaciju sa zaleđenim genima. Time je ostvareno da je broj nedopustivih rešenja zanemarljiv. Primenjene su razne strategije za sprečavanje preuranjene konvergencije genetskog algoritma, kao i tehnika keširanja za poboljšavanje performansi algoritma.

Genetski algoritam je testiran na javno dostupnim instancama iz literature. Predloženi genetski algoritam je za svaku instancu postigao mnogo bolja rešenja od izračunate gornje granice. Za date instance u literaturi nisu poznata optimalna rešenja, ali dobijeni rezultati pokazuju da je predloženi pristup vrlo efikasan u rešavanju datog problema na instancama velikih dimenzija – za grafove sa 110 i 120 čvorova i 1366 i 1611 grana.

Ključne reči: Problem bojenja grafa, Kombinatorna optimizacija, Metaheuristike, Genetski algoritmi

A genetic algorithm for the Bandwidth Coloring Problem and his application in teaching

Abstract

In this paper, a genetic algorithm for the Bandwidth Coloring Problem is proposed**.** This problem is generalization of well-known NP-hard graph coloring problem and it has significant application in practice. It is used in solving the frequency assignment problems in mobile and radio networks, to optimized delivering dangerous items, as well as in education, when it is necessary to organize the timetable of examinations of a university, the timetable of tests and written tests for one month, semestar or entire school year, use of video projectors and other school supplies.

For solving the Bandwidth Coloring Problem the specially designed integer representation and genetic operators are adapted to the problem: fine-grained tournament selection, one-point crossover, simple mutation with frozen genes. In this way is achieved that the number of infeasible solutions is negligible. Various strategies to prevent premature convergence of genetic algorithm is applied as well as cashing techniques to improve the performance of the algorithm.

Genetic algorithm is tested on the publicly-available instances from the literature. Proposed genetic algorithm has achieved a much better solution than the calculated upper limit for a given problem. The given instances are unsolved in the literature so far, but obtained solutions shows that proposed approach is very effective in solving the large-scaled problem instances – for graphs with 110 and 120 verteces and 1366 and 1611 edges.

Keywords: Graph Coloring Problem, Combinatorial Optimization, Metaheuristics, Genetic Algorith

PREDGOVOR

Rad se sastoji od pet poglavlja. U uvodnom poglavlju prikazane su osnovne informacije o NP-kompletnim problemima, heuristikama, metaheuristikama i genetskim algoritmima. U drugom poglavlju formulisan je NP-težak uopšteni problem bojenja grafa, izložen je kratak pregled raznih varijanti ovog problema i data je njegova matematička formulacija. Opis predloženog genetskog algoritma za rešavanje uopštenog problema bojenja grafa je dat u trećem poglavlju. Četvrto poglavlje sadrži eksperimentalne rezultate genetskog algoritma. Zaključak sadrži kratak pregled primenjenih metoda i opis naučnog doprinosa ovog rada.

Želela bih da se zahvalim:

Mentoru doc dr Vladimiru Filipoviću i komentoru dr Jozefu Kratici na rukovođenju pri izradi ovog rada. Posebnu zahvalnost dugujem komentoru dr Jozefu Kratici koji me je zainteresovao za genetske algoritme.

Članu komisije dr Aleksandru Saviću na korisnim savetima koji su doprineli kvalitetu celog rada.

Kolegi Denisu Ćoriću na strpljenju i pomoći pri obradi teksta i crteža.

Najviše se zahvaljujem svojim roditeljima i bratu na bezgraničnom razumevanju, ljubavi i podršci.

Kandidat

Beograd, 2010.

Jasmina Fijuljanin
profesor matematike i računarstva

1 Uvod

Naglo širenje računarske industrije poslednjih godina doprinelo je napretku programerskih kao i nekih matematičkih disciplina: kombinatorna optimizacija, diskretna analiza, numerička analiza, teorija algoritama.

Oblast na kojoj se intenzivno radi i koja beleži brz napredak je *kombinatorna optimizacija*. Razvoj računara dao je mogućnost testiranja programa na primerima relativno velike dimenzije. Problem kombinatorne optimizacije se može definisati na sledeći način:

Dat je konačan ili beskonačan prebrojiv diskretan skup S i funkcija $f: S \to \mathbb{R}$. Naći minimum ili maksimum funkcije f na skupu S, tj. rešiti zadatak:

$$\min_{x \in S} f(x) \quad \text{ili} \quad \max_{x \in S} f(x) \tag{1}$$

Skup S se naziva *dopustiv skup*, funkcija f *funkcija cilja* (o kojoj će više reči biti u narednim poglavljima). Za tačku $x \in S$ se kaže da je dopustivo rešenje problema (1). Potrebno je naći sva dopustiva rešenja x^* (ili bar jedno od njih) takvo da je $f(x^*) = \min f(x)$ ili $f(x^*) = \max f(x)$ (u slučaju da je problem maksimizacije). Takva rešenja se nazivaju *optimalna rešenja.*

Za problem kombinatorne optimizacije sa konačnim skupom S postoje algoritmi potpune pretrage. Ako je kardinalnost skupa S velika nemoguće je primeniti potpunu pretragu. Na primer, ako bi trebalo pretražiti svih 2^n temena, n – dimenzione kocke i ako je za funkciju cilja u jednom temenu potrebno samo 10^{-10} sekundi, za $n = 100$ pretraga bi trajala više od $3*10^{12}$ godina ([Kom96]).

1.1 Složenost algoritama i NP-kompletni problemi

Često se dešava da programi na test-primerima korektno rade, ali da za veće dimenzije problema koje se javljaju u praksi, njihovo izvršavanje traje neprihvatljivo dugo ili je nedostižno. Zbog toga se javlja ideja o klasifikaciji algoritama po brzini izvršavanja. Pored vremena izvršavanja (vremenske složenosti) vrlo važan aspekt svakog algoritma je i potrošnja memorijskog prostora (prostora složenost). Detaljnije u [Brs88], [Crm90], [Man91], [Uro96], [Pau97].

1.1.1 Vremenska složenost algoritma

Da bi se izbegle razlike u konstrukciji i performansama konkretnih računara vremenska i prostorna složenost algoritma ocenjuju se asimptotski. Na taj način vrednujemo kvalitet algoritma bez obzira na kojoj se platformi izvršava.

Vreme složenosti nekog algoritma za rešavanje zadatka $g(n)$ je maksimalno vreme koje je potrebno algoritmu za nalaženje rešenja zadatka koji ima dimenziju n. Za merenje efektivnosti algoritma najčešće se koristi Landauov simbol O (veliko O).

Definicija 1: Za dati problem, čiji su ulazni podaci dimenzije n, kažemo da je određeni algoritam vremenske složenosti $O(g(n))$, ako vreme izvršavanja algoritma u *najgorem slučaju* ne prelazi vrednost $c \cdot g(n)$, gde je c konstanta.

Definicija 2: Algoritam je *polinomske složenosti* po vremenu izvršavanja, ako je vremenske složenosti najviše $O(n^k)$, za neku konstantu k.

Neki od algoritama polinomske složenosti su algoritmi za:
• pretragu uređenog niza, nalaženje Fibonacci-jevih brojeva, ... (složenosti $O(\log n)$);
• pretragu neuređenog niza, zbir elemenata niza, ... (složenosti $O(n)$);
• sortiranje elemenata niza, ...(složenosti $O(n \log n)$);
• sabiranje matrica, množenje matrice vektorom, nalaženje najkraćeg puta ...
(složenosti $O(n^2)$) itd.

Definicija 3: Algoritam je *eksponencijalne složenosti* po vremenu izvršavanja, ako nije polinomske složenosti.

S obzirom da vrednost eksponencijalnih funkcija, kao i funkcije *n!* mnogo brže raste sa porastom *n* od vrednosti stepenih funkcija (polinoma), algoritmi polinimske složenosti za veliko *n* su jedini efikasni u praksi. U [Ognj04a] se može pronaći detaljnija klasifikacija složenosti algoritama.

Problem kod koga je rešenje oblika DA ili NE, se naziva problem *odlučivanja* (decision problem).

Definicija 4: Problem pripada klasi složenosti **P**, i nazivamo ga problemom *polinomske složenosti*, ako se rešava, u opštem slučaju, nekim od poznatih algoritama polinomske složenosti.

Definicija 5: Problem pripada klasi **NP**, i nazivamo ga problemom *nedeterminističke polinomske složenosti*, ako se algoritmom polinomske složenosti može verifikovati. Za unapred dato rešenje se utvrđuje se da li su ispunjeni svi uslovi problema. Da bi odgovor bio i formalno (matematički) korektan potrebno da on bude zadat kao problem odlučivanja.

Za svaki problem polinomske složenosti postoji poznat algoritam polinomske složenosti koji ga rešava, pa time trivijalno verifikuje dato rešenje, pa na osnovu toga zaključujemo da je klasa problema **P** potklasa problema **NP**, tj. $\mathbf{P} \subseteq \mathbf{NP}$. Pitanje da li je $\mathbf{P} \subset \mathbf{NP}$ ili $\mathbf{P} = \mathbf{NP}$ je jedno od ključnih pitanja savremene matematike na koje još uvek nemamo odgovor.

Rezultati višegodišnjih istraživanja pokazuju da klasa **NP** sadrži neke probleme koji ne pripadaju klasi **P**, mada za to još uvek nemamo matematički dokaz.

Problemi za koje možemo konstruisati eksponencijali algoritam i za koji možemo dokazati da se isti ne može izbeći ne pripadaju ni klasi **P** ni klasi **NP**. Detaljnije u [Kom96], [Pap82], [Ynn97].

1.1.2 NP-kompletni problemi

Definicija 6: Za problem $\boldsymbol{Q_1}$ kažemo da je *svodiv u polinomskom vremenu* na problem $\boldsymbol{Q_2}$ ako postoji algoritam polinomske složenosti koji pretvara svaku interpretaciju problema $\boldsymbol{Q_1}$ u interpretaciju problema $\boldsymbol{Q_2}$ tako da imaju analogno zajedničko rešenje.

Definicija 7: Problem odlučivanja nazivamo ***NP-kompletnim*** ako:

- pripada klasi **NP**

- bar jedan poznati **NP**-kompletni problem se algoritmom polinomske složenosti može svesti na dati problem

Ako unapred prihvatimo neki problem za **NP**-kompletan, korišćenjem metoda svođenja u polinomskom vremenu relativno lako nalazimo ostale **NP**-kompletne probleme.

Tokom 70-tih godina za više od 300 problema je dokazano da pripadaju klasi **NP**-kompletnih problema. Prvi pregled **NP**-kompletnih problema dat je u [Gar79]. Ovi problem su podeljeni u 12 tematskih celina i ova podela se održala do danas. Detaljnije informacije o trenutnom spisku **NP**-kompletnih problema se mogu videtu u [NPc98].

Složenost algoritama se ocenjuje asimptotski i računa za najnepovoljniju varijantu njegovog izvršavanja.
NP-kompletan problem velike dimenzije se ne može rešiti pretragom ni na savremenim računarima sa najboljim karakteristikama. Zbog toga su za rešavanje ovih problema pronađene razne približne metode. Međutim, te metode ne garantuju pronalaženje optimalnog rešenja. Pomenute metode se nazivaju *heuristike*, a rešenja koja se uz pomoć njih dobijaju su *suboptimalna rešenja*.

Za problem bojenja grafa sa ograničenjima širine opsega koji je uopštenje poznatog problema bojenja grafa je dokazano da je **NP**-kompletan (u [GaJo79]) u opštem slučaju, mada mogu postojati određeni specijalni slučajevi kada je problem polinomske složenosti. To je slučaj kada graf bojimo samo sa dve boje.

1.2 Heuristike i metaheuristike

Kao što je već pomenuto heuristike ne koriste klasične matematičke postupke bazirane na teoriji pa zato njihova primena ne garantuje nalaženje optimalnog rešenja. Heuristike omogućavaju primenu zdravorazumskih pravila koja često imitiraju proces ljudskog mišljenja.

U početku heuristike nisu bile toliko popularne jer su davale rešenja slabijeg kvaliteta (sa relativno velikim odstupanjima od optimalnog rešenja), konstruisane su samo za jedan ili više sličnih problema, zaustavljale su se u prvom lokalnom ekstremu, bez mogućnosti da iz njega izađu, i dostignu globalno optimalno rešenje. U poslednjih 10-20 godina dolazi do velikog pomaka u ovoj oblasti, pa se heuristke sve češće koriste.
Termin heuristika potiče od starogrčke reči “heuriskein” što znači “naći” ili “otkriti”.

Heuristički algoritmi treba da rade u razumnom vremenu, tj. da budu računski efikasni. Heuristika treba da bude jednostavna, odnosno razumljiva za onog ko je koristi, takođe treba da bude robusna, tj. da ne menja drastično svoje ponašanje za male promene parametara.
Klasifikacija heurističkih metoda se najčešće vrši prema njihovom opštem pristupu rešavanja i može se naći u [Opr04]. Ono što u okviru klasifikacije treba pomenuti je konstruktivna metoda i u okviru nje „pohlepni“ (greedy) algoritam kod koga se u svakoj iteraciji od trenutno mogućih izbora bira onaj koji je “najbolji” po nekom lokalnom kriterijumu. Ovaj „pohlepni“ (greedy) algoritam je korišćen u predloženom genetskom algoritmu za dobijanje gornje granice rešenja u poglavlju 3.
Od sredine 80-tih godina razvija se i primenjuje niz tvz. *opštih heuristika* koja daju opšta uputstva za rešavanje problema, nezavisno od strukture konkretnog problema. Ove heuristike nazivamo *metaheuristikama.*
One se uz određene izmene mogu priimeniti na široku klasu problema, a najpoznatije metaheuristike su:

- ***Genetski algoritmi (genetic algorithms)*** – detaljnije o genetskim algoritmima u narednim poglavljima
- ***Simulirano kaljenje (Simulated annealing***) – metoda zasnovana na principima statičke termodinamike. Ova metoda u odnosu na trenutnu tačku pretraživanja generiše na slučajan način suseda i nezavisno od poboljšanja ili pogoršanja vrednosti funkcije cilja prihvata tog suseda sa određenom verovatnoćom. Na ovaj način se poboljšava kvalitet rešenja i sprečava se preuranjena konvergencija. Unapred zadat maksimalan broj iteracija je kriterijum zaustavljanja. Detaljnije o simuliranom kaljenju videti u [Art97b], [Lam88], [Alv92], [Krp83], [Sum06] i [Sum02].
- ***Tabu pretraživanje (Tabu search)*** – prvi je uveo Glover u 1986. godine u radu [Glo86]. Tabu pretraživanje kao svoju najvažniju komponentu koristi adaptivnu memoriju. Adaptivna memorija sadrži podatke o prethodnim fazama pretraživanja i utiče na sledeće izbore u procesu. Detaljnije u [Her97], [Joh96], [Kno89], [Mis05], [Glo97], [Glo90] i [Glo77].
- ***Lagranževa relaksacija (Lagrangian relaxation)*** - nekim od uslova zadatka, dodeljuje odgovarajuće faktore koji se nazivaju Lagranževi množioci, čime se oni uključuju u vrednosnu funkciju. Dobijeno optimalno rešenje na ovaj način definisanog problema predstavlja donju granicu za ocenu rešenja polaznog problema. Može se dobiti jednostavniji problem sa mnogo kraćim vremenom optimalnog rešavanja ako se pogodno izaberu uslovi koji se uključuju u vrednosnu funkciju. Detaljnije u [BeJ88], [Gui88], [BeJ90b], [BeJ93] i [Glv93].
- ***Metoda promenljivih okolina (Variable neighborhood search - VNS)*** – nastala 1996. godine, zasniva se na lokalnom pretraživanju. Detaljnije u [Han99], [Han01], [Han07], [Mla95], [Mla97].
- ***Programiranje sa ograničenjima*** (***Constraint programming)***

- Brojni metaheuristički algoritmi koji se razvijaju kao što su: neuralne mreže ([Fan90]), mravlji sistemi (ant systems) opisani u [Dor96], epsilon transformacija ([Zha96], [Pem96] i [Kra96b]).

1.3 Genetski algoritmi

Genetski algoritmi su metaheuristike koje simulirajući mehanizam prirodne evolucije rešavaju računarske probleme. Zasnovani su na Darvinovoj teoriji o postanku vrsta (nastaloj krajem 19. veka) [Dar59] i Mendelovim zakonima [Bow89].

Prvi radovi koji se mogu klasifikovati u ovu oblast su nastali 60-tih godina. Bez obzira na to idejnim tvorcem genetskih algoritama smatra se John Holland, koji je 70-tih godina u knjizi *„Adaptacija u prirodnim i veštačkim sistemima" („Adaptation in natural and artificial systems")* [HII75] postavio temelje ove metode.

Postoji veliki broj radova o genetskim algoritmima. Neki od njih su: [DJo75], [Bok87], [Gol89], [Dav91], [BeD93a], [BeD93b], [Yur94], [Mic96], [Mit96] i [Müh97]. Detaljnije o genetskim algoritmima se može naći i kod domaćih autora: [Čan96], [Fil97], [Kra97a], [Toš97] i [Fil98].

1.3.1 Opis genetskih algoritama

Kao što je već rečeno, genetski algoritmi su adaptivne metode kojima se rešavaju različiti problemi kombinatorne optimizacije. Genetski algoritam se primenjuje na konačnom skupu jedinki koji se naziva *populacija.*

U praksi uobičajeno je da je broj jedinki u populaciji između 5 i 500. Za dati problem svaka jedinka predstavlja moguće rešenje u pretraživačkom prostoru. Jedinka se predstavlja genetskim kodom nad određenom konačnom azbukom. Najčešće se koristi binarno kodiranje kod koga se genetski kod sastoji od niza bitova. Nekada su azbuke veće kardinalnosti pogodnije.

Neadekvatno kodiranje može dovesti do loših rezultata, pa je zbog toga način kodiranja veoma bitan za rešavanje problema ([Bor09]).

Da bismo dobili što raznovrsniji genetski materijal, početna populacija se najčešće generiše na slučajan način. Generisanje cele (ili dela) početne populacije nekom pogodno izabranom heuristikom nekada dovodi do boljih rezultata. U ovom slučaju treba obratiti pažnju na to da se što manje smanjuje raznovrsnost genetskog materijala i da vreme izvršavanja date heuristike bude relativno kratko.

Kvalitet jedinke se ocenjuje na taj način što svakoj jedinki dodeljuje funkcija prilagođenosti (fitness function). Operatori, koji obezbeđuju da se iz generacije u generaciju poboljšava apsolutna prilagođenost svake jedinke u populaciji su operatori *selekcije, ukrštanja i mutacije* i njihovom uzastopnom primenom poboljšava se i srednja prilagođenost celokupne populacije. Na ovaj način se za konkretni primer dobijaju sve bolja rešenja.

Operator *selekcije* bira natprosečno prilagođene jedinke tako da one dobijaju veću šansu za reprodukciju pri formiranju nove generacije. Slabije prilagođene jedinke postepeno "izumiru".

Razmenu gena jedinki vrši operator *ukrštanja.* Rezultat ukrštanja je razmena genetskog materijala između jedinki, sa mogućnošću da dobro prilagođene jedinke generišu još bolje. Svoju šansu da rekombinacijom dobrih gena proizvedu dobro prilagođene jedinke dobijaju i relativno slabije prilagođene jedinke. Verovatnoća ukrštanja se unapred zadaje. Broj jedinki koje učestvuje u ukrštanju proizvodeći nove jedinke i broj jedinki koji se prenosi u sledeću generaciju bez modifikacija određuje upravo ova verovatnoća.

Neki regioni pretraživačkog prostora postaju nedostupni višestrukom primenom operatora selekcije i mutacije pa se javlja mogućnost gubljenja genetskog materijala.
Operator *mutacije* sa datom malom verovatnoćom p_{mut} vrši slučajnu promenu određenog gena, čime je moguće vratiti izgubljeni genetski materijal u populaciju. Na taj način se sprečava preuranjena konvergencija genetskog algoritma u lokalnom ekstremu.

Na *slici 1.1* ([Dju10]) dati su osnovni koraci genetskog algoritma:

```
Unošenje_Ulaznih_Podataka();
Generisanje_Početne_Populacije();
while(!Kriterijum_Zaustavljanja_Genetskog_Algoritma())
    {
        for (i=1;  i<=N_populacije;  i++ )
            p_i=Funkcija_Cilja(i);
        Funkcija_Prilagođenosti();
        Selekcija();
        Ukrštanje();
        Mutacija();
    }
Štampanje_Izlaznih_Podataka();
```

Slika 1.1 Shematski prikaz GA

1.3.2 Prost genetski algoritam

Prost genetski algoritam *(simple genetic algorithm -SGA)* je najjednostavnijji koncept genetskog algoritma. Sastoji se od proste rulet selekcije, jednopozicionog ukrštanja i proste mutacije. Opis prostog genetskog algoritma može se naći u *Holland-ovoj* knjizi [Hll75], a u narednim odeljcima su opisane osobine navedenih genetskih operatora.

Prilikom primene prostog genetskog algoritma problem koji se najčešće javlja je *preuranjena konvergencija.* Do nje dolazi ukoliko jedna ili nekoliko relativno dobrih jedinki, ali ne i optimalnih, postepeno preovlada u populaciji i proces konvergira u lokalnom ekstremu. Tada su mogućnosti za poboljšanje genetskog algoritma veoma male. Preuranjena konvergencija najčešće nastaje kao posledica primene proste rulet selekcije.
Lošije jedinke će biti izbačene iz populacije jer selekcija i ukrštanje u populaciji sa istim jedinkama nemaju efekat. Teorijski, u ovakvoj situaciji jedino mutacija može da pomogne, međutim u praksi ni ona u većini slučajeva ne daje efekat. Ako je nivo mutacije relativno veliki, genetski algoritam se pretvara u slučajnu pretragu, a ako je nivo mutacije relativno mali, dominantne jedinke vrlo brzo eliminišu sve ostale jedinke iz populacije jer su u tom slučaju promene genetskog materijala neznatne.

Spora konvergencija je takođe jedan od nedostataka prostog genetskog algoritma. Ona predstavlja problem suprotan problemu preuranjene konvergencije. Javlja se uglavnom u kasnoj fazi izvršavanja prostog genetskog algoritma. Ako nije dostignuto optimalno rešenje, a populacija je postala dovoljno slična, tada su razlike između najbolje jedinke i ostalih jedinki u populaciji male, a srednja prilagođenost svih jedinki u populaciji je velika. Tada za genetski algoritam ne postoji dovoljni gradijent u funkciji prilagođenosti, koji bi mu pomogao da se približi optimalnoj vrednosti u dostižnom broju generacija.

1.3.3 Složeniji koncept genetskog algoritma

Samo se manji broj problema kombinatorne optimiacije mogu rešiti uz pomoć prostog genetskog algoritma upavo zbog nedostataka navedinh u prethodnom poglavlju. Za rešavanje složenijih problema koriste se poboljšane verzije genetskog algoritma. Ta poboljšanja ogledaju se u korišćenju raznih vrsta kodiranja i odgovarajućih vrednosnih funkcija, više vrsta funkcija prilagođenosti, različitih politika zamene generacija, zatim u fiksnoj ili adaptivnoj promeni parametara tokom izvršavanja genetskog algoritma.

1.3.3.1 Kodiranje i funkcija prilagođenosti

Postoje samo dve komponente genetskog algoritma koje zavise od konkretnog problema: reprezentacija rešenja i funkcija cilja.

Genetski algoritmi zahtevaju neku meru kvaliteta, tj. ispravnosti predloženog rešenja. Ovu meru daje funkcija cilja, odnosno funkcija *prilagođenosti* (*fitness).*U literaturi se još može naći pod nazivima funkcija sposobnosti ili funkcija ocene kvaliteta.

Funkcija prilagođenosti treba da bude neprekidna i glatka da bi jedinke sa sličnim genetskim kodom imale sličnu vrednost prilagođenosti. Za dobre rezultate takođe je važno da funkcija prilagođenosti nema suviše izolovan globalni ekstremum ili mnogo lokalnih ekstremuma ([BeD93a]).

Načini računanja funkcije prilagođenosti koji se najčešće koriste su: *direktno preuzimanje, linearno skaliranje, skaliranje u jedinični interval i sigma odsecanje.*

Kod direktnog preuzimanja se za vrednost funkcije prilagođenosti neke jedinke uzima njena vrednost funkcije cilja, dok kod linearnog skaliranja prilagođenost neke jedinke se računa kao linearna funkcija vrednosti funkcije cilja te jedinke. Funkcija prilagođenosti uzima vrednosti iz intervala [0, 1] kada je u pitanju skaliranje u jedinični interval. Izbor načina računanja funkcije prilagođenosti zavisi od konkretnog problema, a često se dešava da se kombinuju različiti načini ([Sta04]).

Najbolje je uspostaviti bijektivnu vezu između rešenja problema i njihovih genetskih kodova, mada kodiranje ne mora biti bijektivno, čak ne mora biti ni preslikavanje. Tada se javljaju tvz. nekorektne jedinke koje se mogu izbaciti iz populacije postavljanjem njihovih vrednosti na nulu.

Za poboljšanje genetskog algoritma i funkcije prilagođenosti koriste se Gray kodovi kod kojih se susedni genetski kodovi razlikuju samo u jednom bitu ([Kra00]).

1.3.3.2 Operator selekcije

Odabir jedinki koje će učestvovati u postupku stvaranja nove generacije je zadatak operatora selekcije. Najjednostavniji oblik operatora selekcije je *prosta rulet selekcija*. Ona je u direktnoj vezi sa funkcijom prilagođenosti jer je verovatnoća selekcije proporcionalna njenoj prilagođenosti. Zbog postepenog preovlađivanja visoko prilagođenih jedinki u populaciji koje ne odgovaraju globalnom optimumu dolazi do preuranjene konvergencije koja zapravo predstavlja osnovni problem ove metode.
Selekcija zasnovana na *rangiranju* genetskih kodova prema prilagođenosti se koristi da bi se prevazišao problem preuranjene konvergencije.

Još jedan od oblika selekcije je *turnirska selekcija*. Turniri su takmičenja dve ili više jedinki koje se nadmeću da bi učestvovale u sledećoj generaciji. Učesnici na turniru se biraju na slučajan način. Broj turnira jednak je broju jedinki, a veličina turnira N_{tur} (celobrojni parametar) se obično unapred zadaje. Jedinka se bira ako je pobedila na turniru. Međutim, veliki nedostatak turnirske selekcije je nemogućnost izbora pogodnog parametra N_{tur}. Upravo zbog ovog nedostatka često se događa da za jedan parametar konvergencija bude veoma spora, a da povećanje parametra za jedan dovede do preuranjene konvergencije, odnosno vezivanja rešenja za lokalni ekstremum. Iz ovih razlog za predloženi genetski algoritam korišćeno je unapređenje standardnog operatora turnirske selekcije, poznato kao *fino – gradirana turnirska selekcija*. Umesto celobrojnog parametra N_{tur}, fino – gradirana turnirska selekcija zavisi od parametra F_{tur} - željena srednja veličina turnira, koji uzima realne vrednosti. Kao i kod turnirske selekcije, jedinka će biti izabrana ukoliko je bolja od određenog broja slučajno izabranih protivnika, ali

veličina izbornih turnira nije jedinstvena u okviru populacije. Koriste se dva tipa turnira: prvi se sprovodi k_1 puta i njegova veličina je [F_{tur} +1], dok se drugi sprovodi k_2 puta, gde je broj jedinki koje učestvuju u turniru [F_{tur}]. Pri tome važi: $F_{tur} \approx \frac{k_1 \cdot \left[F_{tur} \right] + k_2 \left[F_{tur} \right]}{N_{nnel}}$, gde je $N_{nnel} = k_1 + k_2$.

Detaljnije o ostalim tipovima operatora selekcije se može naći u: [Fil98], [Fil06], [Fil00], [Fil01], [Fil03].

1.3.3.3 Operator ukrštanja

Operator ukrštanja je binarni operator koji se primenjuje nad jedinkama koje se nazivaju roditelji pri čemu nastaje jedna ili dve nove nove jedinke koje se nazivaju potomci. Najbitnija osobina ukrštanja je da potomci nasleđuju osobine svojih roditelja. Ukrštanje može biti *jednopoziciono, dvopoziciono, višepoziciono ili uniformno*, ali postoje i složeniji oblici ovog operatora [Müh97].

Jednopoziciono ukrštanje predstavlja najprostiji način ukrštanja, gde se na slučajan način bira ceo broj *k* iz intervala [0, *l*-1], gde je *l* dužina jedinke roditelja. Izabrani ceo broj *k* predstavlja tačku ukrštanja gena. Svi geni, počev od pozicije *k*+1, do poslednje pozicije *l*-1 u genetskim kodovima roditelja uzajamno menjaju mesta, pri čemu se stvaraju dva nova potomka.

Kod *dvopozicionog* ukrštanja, slučajno se biraju dve pozicije k_1 i k_2 iz intervala intervala [0, *l*-1] i uzajamno se razmenjuju delovi kodova roditelja od pozicije $k_1 + 1$ do pozicije k_2.

U slučaju *uniformnog* ukrštanja za svaki roditeljski par se generiše binarni niz iste dužine kao genetski kod jedinki, tvz. „maska“. Na mestima gde maska uzima vrednost 1, roditelji zadržavaju svoje gene, a na pozicijama gde maska ima vrednost 0, roditelji razmenjuju gene.

Kod ukrštanja različiti pristupi koji se koriste mogu doneti relativno male promene u performansama genetskog algoritma. Ako su geni međusobno nezavisi obično se koristi uniformno ukrštanje. Dvopoziciono ili višepoziciono ukrštanje se koristi ako želimo da u većoj meri izmešamo blokove u genetskom kodu, a jednopoziciono ukrštanje koristimo kada želimo da sačuvamo strukturu genetskog koda.

Primer 1: Neka je data populacija od pet jedinki: 00110011, 10001010, 01011010, 00110010, 11011001. Neka su izabrani prvi i treći i drugi i peti par i neka se ukrštanje vrši u oba izabrana

para. Neka je nivo ukrštanja 0.85. Ako je pozcija prvog para za ukrštanje k=2 i pozicija drugog k=4, tada se ukrštanjem dobijaju sledeće nove jedinke:

I par roditelja (prvi i treći)		II par roditelja (drugi i peti)	
PRE UKRŠTANJA	POSLE UKRŠTANJA	PRE UKRŠTANJA	POSLE UKRŠTANJA
001|10011	00101110	10001|010	10001111
010|11010	01000110	11011|001	11011101

Nakon ukrštanja dobijamo sledeću populaciju: 00101110, 10001111, 01000110, 00110010, 11011101.

Primer 2: uniformno ukrštanje

maska: 11001010

pre ukrštanja	posle ukrštanja
XXXXXXXX	XXYYXYXY
YYYYYYYY	YYXXYXYX

1.3.3.4 Operator mutacije

Najvažniji operator genetskog algoritma koji menja slučajno izabrane jedinke i vraća koristan genetski materijal izgubljen u selekciji i ukrštanju naziva se mutacija. Kako se primenjuje nad samo jednom jedinkom mutacija je unarni operator. Najčešće se koriste *prosta mutacija, mutacija pomoću normalne raspodele i mutacija pomoću binomne raspodele.*

Operator *proste mutacije* se najčešće upotrebljava kada genetski algoritam koristi binarno kodiranje i populacija nema nekorektnih jedinki. Na početku algritma zadaje se nivo mutacije p_{mut} − koji zapravo predstavlja verovatnoću kojom se svaki bit mutira. Nekada se za prostu mutaciju koristi „maska“ – slučajno generisan binarni niz koji nosi informaciju o tome na kojoj poziciji u genetskom kodu dolazi do promene gena.
Da bismo ubrzali izvršavanje operatora proste mutacije koristimo binomnu ili normalnu raspodelu.

Mutacija pomoću *binomne raspodele* se zasniva na tome da slučajno generisana promenljiva X_{mut} koja predstavlja broj mutiranih gena jedinke ima binomnu raspodelu B(n, p_{mut}), gde je n dužina genetskog koda, a p_{mut} nivo mutacije. Na slučajan način se bira $s \in \{[0,1]\}$, a zatim se pronalazi X_{mut} za koje važi: $F(X_{mut}) \leq F(s) < F(X_{mut} + 1)$, gde je F funkcija raspodele.

Binomna raspodela se aproksimira normalnom raspodelom

$$N(N_{bit} \cdot p_{mut} ; N_{bit} \cdot p_{mut} \cdot (1-p_{mut})) \quad (2)$$

kada je proizvod dužine genetskog koda i nivoa mutacije dovoljno veliki ([Mar08]). Detaljnije o navedenim mogućnostima za mutaciju pogledati u [Kra00] i [Toš04].

Kada geni genetskog koda nisu ravnopravni neke delove koda jedinke je potrebno mutirati sa manjom ili većom verovatnoćom. Tada se obično koristi *normalna mutacija* (primenjuje se u skladu sa normalnom raspodelom) ili *eksponencijalna mutacija* (gde broj mutiranih gena u kodu eksponencijalno opada).

1.3.3.5 Kriterijum zaustavljanja

Nad početnom populacijom se izvršava proces koji se sastoji od selekcije, ukrštanja i mutacije dok se ne zadovolji neki uslov zaustavljanja. Kada se ispuni uslov zaustavljanja algoritam se završava. Teško je formalno definisati kriterijum zaustavljanja s obzirom da su genetski algoritmi sholastička metoda pretrage. Kriterijumi zaustavljanja koji se najčešće primenjuju su:

- ✓ dostignuti maksimalni broj generacija
- ✓ sličnost jedinki u populaciji
- ✓ ponavljanje najbolje jedinke određeni (maksimalni) broj puta
- ✓ dostignuto optimalno rešenje ako je ono unapred poznato
- ✓ dokazana optimalnost najbolje jedinke (ukoliko je to moguće)
- ✓ ograničavanje vremena izvršavanja genetskog algoritma
- ✓ prekid od strane korisnika itd.

U praksi najbolje pokazalo kombinovanje ovih kriterujuma zaustavljanja jer svaki od njih ima dobre i loše strane. Na taj način se smanjuje mogućnost loše procene prekida genetskog algoritma.

1.3.3.6 Ostali aspekti genetskog algoritma

Politika zamene generacija je jedan od bitnih aspekata genetskog algoritma. Strategije koje se najčešće primenjuju su:

- ✓ *Generacijska (generational)* kod koje se sve jedinke u populaciji menjaju u svakoj generaciji
- ✓ *Stacionarna (steady- state)* kod koje se generiše samo deo populacije u svakoj generaciji, a preostale jedinke se prenose iz prethodne generacije.
- ✓ *Elitistička strategija (elitist strategy)* – ova strategija omogućava da bez primene genetskih operatora selekcije, ukrštanja i mutacije i bez računanja funkcije prilagođenosti jedna ili više najboljih (elitnih) jedinki prođe direktno u narednu generaciju.

Za poboljšavanje početne populacije kao i svake naredne generacije, primenom na celu populaciju, jedan njen deo ili na pojedinačnu jedinku genetski algoritmi se kombinuju sa drugim heuristikama. Paralelizacijom i keširanjem performanse genetskog algoritma bitno se mogu poboljšati, o čemu se detaljnije može videti u [Kra00].

2 Uopšteni problem bojenja grafa sa ograničenjima širine opsega

Problem bojenja grafova je bio jedan od prvih problema za koji se pokazalo da je NP-kompletan (NP-težak). Prvi algoritmi za rešavanje problema bojenja grafa datiraju iz 60-tih godina 20. veka ([ChN71], [WePo67]) i od tada se problem intenzivno proučava. U literaturi se nalazi veliki broj algoritama za rešavanje ovog problema koji spadaju u tri glavna pristupa rešenju ([PoHk10]): sekvencijalna metoda (brza, ali ne tako efikasna), metode lokalnog pretraživanja (tabu pretraživanje ([Blz03], [DoH98a], [FlFe96], [GaHa99], [HeWe87], [PoHa10]), simulirano kaljenje ([ChW87], [JaMa91]), metode promenljivih okolina ([Avh03], [HePl08], [TrYi07])) i metode zasnovane na populaciji ([Brs88], [DoH98], [DoMa96], [FlFe96], [GaHa99], [Gahz08], [LüHa10], [MoTo08]).

U praksi bojenje grafova ima široku primenu u oblastima kao što su raspoređivanje (scheduling), pravljenje rasporeda časova ili ispita (timetabling), alokacija registara u prevodiocima programskih jezika (register allocation in compilers), dodela frekvencija u ćelijskim mrežama (frequency assignment in cellular networks).

Definišimo pojmove:

- *Minimalna udaljenost između čvorova grafa* je unapred zadata vrednost za svaki par čvorova, kao i za svaki čvor u odnosu na samog sebe
- *Udaljenost između čvorova grafa* je apsolutna vrenost razlike boja dodeljenih svakom paru čvorova

Uopšteni problem bojenja grafa sa ograničenjima širine opsega se sastoji u dodeljivanju boja za svaki čvor na grafu, tako da udaljenost između čvorova grafa bude manja ili jednaka od minimalne udaljenosti.

Problem bojenja grafa (The Graph Coloring Problem - GCP) se javlja u četiri varijante :

- ***Problem bojenja čvorova grafa (The Vertex Coloring Problem – VCP***) – svaki čvor se boji jednom bojom, a minimalna udaljenost između svih čvorova je jednaka jedan
- ***Problem bojenja čvorova grafa sa ograničenjima širine opsega (The bandwidth coloring problem – BCP)*** – u ovom slučaju za svaka dva čvora je udaljenost veća ili jednaka od unapred zadate minimalne udaljenosti
- ***Problem bojenja čvorova grafa nizom boja (The multicoloring problem – MCP)*** – u ovom slučaju definiše se pozitivan broj za svaki čvor i taj broj predstavlja broj boja koji se mora dodeliti svakom čvoru, tako da ne postoje nikoja dva čvora koja su obojena istom bojom.
- ***Problem bojenja čvorova grafa nizom boja sa ograničenjima širine opsega (The bandwidth multicoloring problem – BMCP)*** – predstavlja kombinaciju predhodna dva problema: svakom čvoru mora biti dodeljen određen broj boja i svaka boja mora

poštovati minimalnu udaljenost između boja ostalih čvorova. U ovom slučaju se takođe mora obratiti pažnja na minimalnu udaljenost različitih boja dodeljenih jednom istom čvoru.

1979. godine Garey i Johnson ([GaJo79]) su dokazali da je VCP NP-težak, a kako je VCP specijalan slučaj BMC-a, MCP-a i BMCP-a sledi da su i oni NP-teški. Pored toga BCP i MCP su specijalni slučajevi BMCP.

MCP se može koristiti, na primer, prilikom raspoređivanja poslova sa različitim vremenskim zahtevima, gde dodeljivanje boja čvorovima zapravo predstavlja raspoređivanje poslova slobodnim radnicima. Svakom čvoru se dodeljuju boje tako da njegovim susedima bude dodeljen različit skup boja.

Jedna od najbitnijih primena BCP-a, MCP-a i BMCP-a je pri rešavanju problema dodeljivanja frekvencija (the frequency assignment problems – FAP). Ovaj problem se često javlja kod mobilnih i radio mreža kada kod operatera koji posluju u istom regionu postoji potreba dodeljivanja frekencija, sa različitih predajnika, koje mogu da ometaju jedna drugu ([MaTo08]).
Problem se može posmatrati kao MCP gde je svakom čvoru grafa dodeljen ceo broj koji predstavlja broj poziva koje odgovarajuća bazna stanica mora opslužiti. Nastoji se da se svakom čvoru dodeli skup boja tako da on bude disjunktan sa skupovima boja koji su dodeljeni susednim čvorovima. Cilj je da minimizira broj boja se koristi za takav zadatak. Zbog ometanja signala u mrežama susedni predajnici moraju da rade na odvojenim frekvencijama odnosno treba odvojiti frekvencije do određene granice tako da smetnje budu svedene na minimum za fiksnu raspodelu frekvencija.
Ako FAP posmatramo kao BMCP tada broju čvorova odgovara broj predajnika koji mora da primi odgovarajući broj frekvencija, a udaljenost odgovara minimalnoj udaljenosti medju frekvencijama koje mogu koristiti susedni transmiteri. Dakle, raspon frekvencija mora biti minimiziran.
Ovaj problem u literaturi je poznat kao MS – FAP (The minimum span frequency assignment problem). Za rešavanje MS- FAP-a predloženi su različiti istraživački pristupi ([ZoBe77]) : metoda lokalnog pretraživanja ([TsVo98], [WaRu96]), tabu pretraživanje ([Co93], [Hdog98]), simulirano kaljenje ([Co93]), genetski algoritmi ([Vhu99], [BoČ10], [Aha77]).

2.1 Postojeći načini rešavanja

Od svog nastanka problem bojenja grafa je konstantno proučavan i postoji veliki broj metoda za njegovo rešavanje, kako egzaktnih tako i heurističkih. Zbog toga ćemo navesti samo neke od najznačajnijih rezultata.

Najjednostavnija heuristika za rešavanje VCP-a je metodom *sekvencijalnog bojenja*. Primena ove metode sastoji se u tome što se prvo sortiraju čvorovi, a na vrhu se nalazi čvor sa brojem jedan. Preostali čvorovi seređaju tako što se svaki boji prvom sledećom bojom kojom nisu obojeni njegovi susedni čvorovi. Prvi ovakav pristup rešavanju problema imali su Welsh i Powell 1967. godine ([MaGo09]). Ovu metodu je veoma lako implementirati i veoma je brza, ali ona daje rešenje koje je daleko od optimalnog, tako da se od nje vrlo brzo odustalo. Lim je 2005. godine ([LiL05]) predložio više metoda koje u osnovi prilagođavaju metodu sekvencijalnog bojenja.

Veliki broj radova opisuju algoritme napravljene da reše slučajeve koji predstavljaju realne MS – FAP instance nastale u oblasti telekomunikacija. Tako su dobijene poznate Filadelfija instance, koje je prvi uveo Anderson 1973. godine [And97]. U svim ovim slučajevima n je jednako 21 i svaki predajnik tj. čvor mora da primi veliki broj frekvencija, tj. boja tako da ovaj problem pripada multicoloring verziji.
2002. godine je organizovan *Computational Symposium on Graph Coloring and its Generalizations* (simpozijum sa temom: Bojenje grafa i njegova uopštenja) kako bi se promovisala istraživanja na polju ovog problema ([MaGo09]). Za vreme sipozijuma predstavljena su različita rešenja problema i predložen je skup instanci za VCP, BCP, MCP i BMCP.
Phan i Skiena (videti u [PhSk02]) su predložili da se VCP i BCP reše uz pomoć opštih heuristika, zvanih Discropt (napravljenih za optimizaciju po principu "crne kutije") koje bi bile prilagođene problemu bojenja grafa.
Prestwich ([Spre02]) je za rešavanje ovog problema predložio metodu zvanu pretraživanje okolina proveravanjem unapred (Forward Checking Neighborhood Search - FCNS).

Lim ([LiL05]) je 2005. godine predložio kombinaciju više metoda za rešavanje BCP-a u kojima se početno rešenje dobija korišćenjem "pohlepne" metode (greedy method) . S obzirom na redosled čvorova dati algoritam sekvencijalno dodeljuje najmanju boju svakom čvoru pri čemu vrši proveru ograničenja propustnosti opsega. U svakoj iteraciji metod prvo generiše redosled čvorova, a zatim nastavlja tako što pokušava da smanji broj boja koje se koriste u prethodnoj iteraciji. Lim je ovaj metod primenio u geometrijskim grafovima i dobio rešenja sličnog kvaliteta kao i Prestwich 2002. godine.

Sekvencijalna metoda je zapravo adaptacija poznatog algoritma za stepen zasićenosti (Degree of Saturation - DSATUR) algoritma implementiranog 1979. godine (videti u [DBr79] i [JaMa91]) gde su čvorovi u svakoj fazi izabrani na osnovu njihove ocene, odnosno stepena zasićenja, tj. na osnovu broja boja kojima su bili obojeni. U svakom koraku bira se čvor sa maksimalnim stepenom zasićenja i on se prvi boji. Malaguti i Toth su 2008. godine su za rešavanje BMCP-a predložili (videti u [MaTo08]) kombinaciju DSATUR algoritma i tabu pretraživanja.

Jedna od značajnih metoda za rešavanje problema bojenja grafa je GRASP (Greedy Randomized Adaptive Search Procedure), predstavljen u [ReRi03] . GRASP je iterativni proces u kome se svaka iteracija sastoji iz dve faze: izgradnje i lokalnog pretraživanja. Faza izgradnje gradi moguće rešenje čiji se susedi istražuju dok se ne pronadje lokalni optimum nakon primene faze lokalnog pretraživanja. GRASP je u kombinaciji sa metodom tabu pretraživanja korišćen 2009. godine u [MaGo09] kao predlog rešenja za BCP.

Eksperimetalna studija iz rada [ChS07] uz pomoć metode lokalnog pretraživanja predlaže algoritam za rešavanje GSTCP-a (graph set *T*-colouring problem) za veliki broj instanci.

U [ErTo09] je za rešavanje BCP-a i BMCP-a predložen algoritam zasnovan na neregularnim ćelijskim automatima (irregular cellular learning automata - ICLA), gde se rezultat dobija tako što se BCP ili BMCP prvo uproste, odnosno svedu na VCP, gde je svaki čvor grafa obojen samo jednom bojom.

2.2 Matematička formulacija

Neka je $G = (V;E)$ graf, gde je V skup svih njegovih čvorova, a E skup svih njegovih grana. Neka je $|V|=n$ i $|E|=m$ (skup V ima n članova, a skup E ima m članova). Kao što je već rečeno, problem bojenja grafa zahteva da se svakom čvoru dodeli boja tako da su svi susedni čvorovi obojeni različitim bojama i da se broj boja koje se koriste svede na minimum.

Egzaktni algoritmi predloženi za rešavanje problema bojenja grafa mogu naći optimalno rešenje ovog problema samo za grafove sa malim brojem čvorova, ne više od 100 čvorova. Sa druge strane, realni problemi sadrže grafove sa više stotina ili više hiljada čvorova, koje se rešavaju uz pomoć heuristika ili metaheuristika.

Koristeći celobrojno linearno programiranje (Integer Linear Programming - ILP) za rešavanje problema bojenja grafa moramo definisati sledeća dva skupa binarnih promenljivih: promenljivu x_{ih} ($i \in V$, $h=1,\ldots,n$), gde je $x_{ih}=1$, ako je čvoru i dodeljena boja h i promenljivu y_h ($h=1,\ldots,n$)

koja nosi informaciju da li se boja h koristi. Koristeći uvedene oznake, problem bojenja grafa matematički se može predstaviti na sledeći način ([Ema03]):

$$\min \sum_{h=1}^{n} y_h \tag{1.1}$$

$$\sum_{h=1}^{n} x_{ih} = 1 \qquad \forall i \in V \tag{1.2}$$

$$x_{ih} + x_{jh} \leq y_h \qquad \forall (i,j) \in E;\ h = 1,\ldots, n \tag{1.3}$$

$$x_{ih} \in \{0,1\} \qquad \forall i \in V,\ h = 1,\ldots, n \tag{1.4}$$

$$y_h \in \{0,1\} \qquad h = 1,\ldots, n \tag{1.5}$$

Funkcija cilja (1.1) minimizuje broj boja koje se koriste. Ograničenja (1.2) zahtevaju da je svaki čvor obojen, dok (1.3) nameće da je najviše jedan par od susednih čvorova obojen, bojom koja se koristi. Iz uslova (1.4) i (1.5) se vidi das u promenljive x_{ih} i y_h binarne.

Ovakav oblik ima problem sletanja aviona na aerodrum. Kontrolni toranj dodeljuje avionima vremenski interval u kome oni čekaju sletanje. Ako se vreme dolaska dva aviona poklapa, oni ne mogu koristiti isti vremenski interval čekanja. Čvorovi predstavljaju avione, a grana spaja dva čvora samo ako se vremena dolaska aviona poklapaju ([Ema03]).

U praksi se često dešava da je broj korisnika koji pristupa određenom resursu ograničen ili se može desiti da svaki korisnik troši određeni deo resursa, a da je ukupan kapacitet resursa oganičen. Kada imamo ovakvu situaciju svakom čvoru dodeljujemo pozitivni broj w_i i namećemo ograničenje za ukupan zbir w_i svih čvorova i koji se boje istom bojom. Ovaj problem je poznat kao problem bojenja grafa sa ograničenjem (Bounded Vertex Coloring Problem – BVCP). Ako je C kapacitet svake boje, možemo nametnuti ograničenja kapaciteta na sledeći način (videti u [MaTo90]):

$$\sum_{i=1}^{n} w_i\, x_{ih} \leq C \qquad \forall\ h = 1,\ldots, n \tag{1.6}$$

Problem koji ima matematičku formulaciju koja zadovoljava (1.1)-(1.4) i (1.6) zapravo predstavlja BVCP.

Prilično je očigledno da se problem pravljenja rasporeda korišćenja video bima, projektora ili druge opreme u školama ili učionica koje sadrže posebnu opremu može modelirati na vrlo sličan način, jer je količina pomenute opreme u školama u najvećem broju slučajeva ograničena. Na isti način možemo modelirati problem korišćenja školskog pribora za matematiku, fiziku, hemiju, biologiju, fizičko, muzičko...

Problem pravljenja rasporeda ispita na fakultetima je BVCP. Svaki ispit mora imati svoj termin održavanja, pri čemu fakultet želi da organizuje što više ispita u isto vreme tako da za svaki ispit bude slobodnih učionica, a sve u cilju smanjenja broja termina u kojima će se održavati ispiti. Budući da studenti mogu polagati više od jednog ispita u ispitnom roku, raspored mora biti takav da oni budu u mogućnosti da izađu na svaki ispit koji su slušali, dakle dva ispita ne mogu imati isti termin polaganja, ako je postojao bar jedan student koji je slušao oba ispita. Svakom čvori *i* se dodeljuje boja *h,* ako je ispit *i* zakazan u terminu *h,* pri čemu su dva čvora susedna ako odgovarajući ispiti ne mogu biti raspoređeni u isto vreme (jer postoji bar jedan student koji je slušao oba ispita). Svaki termin ima maksimalni kapacitet *C* koji odgovara broju slobodnih učionica ([Ema03]).

Problem prevoza opasnih materija je takođe BVCP. Određene materije mogu da budu štetne i opasne i zbog toga zahtevaju vozila sa posebnom opremom. Broj takvih vozila je ograničen. Čvorovi predstavljaju materije koje se moraju prevesti; svakom vozilu dodeljuje boja, dok broj vozila koja imaju posebnu opremu predstavljaju maksimalni kapacitet *C*. ([Ema03])

U BCP-u je dato minimalno rastojanje između susednih čvorova i ono zamenjuje uslov (1.3). Minimalno rastojanje između čvorova $d(i,j)$ se definiše za svaku ivicu $(i,j)\in E$ i apsolutna vrednost razlike između boja dodeljnih čvorovima *i* i *j* mora biti veće ili jednako od ovog zadatog minimalnog rastojanja, tj. $|c(i)-c(j)|\geq d(i,j)$. U ovom slučaju može biti potrebno više od *n* boja. Zbog toga neka *H* bude skup svih dostupnih boja. Matematička formulacija BCP-a je sledeća:

$$\min k \qquad (1.7)$$

$$k \geq y_h h \qquad h\in H \qquad (1.8)$$

$$\sum_{h\in H} x_{ih}=1 \qquad i\in V \qquad (1.9)$$

$$x_{ih} + x_{jl} \leq 1 \qquad (i,j)\in E;\ h\in H;\ l\in \{h - d(i,j) + 1,\ldots, h + d(i,j) + 1\} \qquad (1.10)$$

$$x_{ih}\leq y_h \qquad i\in V;\quad h\in H \qquad (1.11)$$

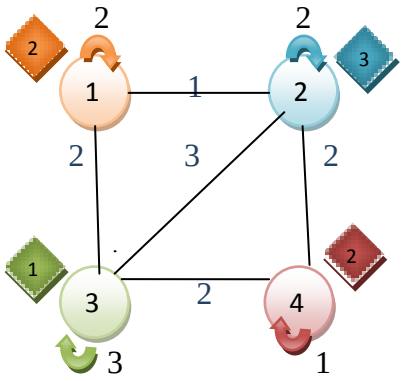
$$x_{ih} \in \{0,1\} \qquad i\in V;\ h\in H \qquad (1.12)$$

$$y_h \in \{0,1\} \qquad h\in H \qquad (1.13)$$

Uslov (1.10) zapravo predstavlja uslov $|c(i)-c(j)|\geq d(i,j)$.

U slučaju MPC-a definiše se pozitivan broj r_i za svaki čvor $i\in V$ i taj broj predstavlja broj boja koji se mora dodeliti svakom čvoru *i*, tako da ne postoje nikoja dva čvora koja su obojena istom bojom. Drugim rečima, za svako $i\in V$ treba pronaći minimalno *k* i skup $S(i)\subset \{1,\ldots,k\}$ tako da je $|S(i)|=k(i)$, $\forall i\in V$ (gde $|S(i)|$ predstavlja kardinalni broj skupa $S(i)$) i gde važi $S(i)\cap S(j)=\emptyset$, $\forall (i,j)\in E$.

Kao što je već poznato BMCP predstavlja kombinaciju prethodna dva problema: svakom čvoru *i* mora biti dodeljen određen broj boja r_i i svaka boja mora poštovati minimalnu udaljenost *d(i,j)* između boja ostalih čvorova *j*. U ovom slučaju se takođe mora obratiti pažnja na minimalnu udaljenost *d(i,i)* različitih boja dodeljenih jednom istom čvoru *i*. Drugim rečima, za svako $i \in V$ treba pronaći minimalno *k* i skup $S(i) \subset \{1, \dots, k\}$ tako da je $|S(i)|=k(i)$, $\forall i \in V$ (gde $|S(i)|$ predstavlja kardinalni broj skupa *S(i)*), gde važi $S(i) \cap S(j) = \emptyset$, $\forall (i,j) \in E$ i gde $\forall p \in S(i)$ i $\forall q \in S(j)$ važi $|p-q| \geq d(i,j)$, $\forall (i,j) \in E$.
Kao što sam već napomenula problem dodeljivanja frekvencija (the frequency assignment problems – FAP) je BMCP i jedan je od najviše proučavanih problema u ovoj oblasti.

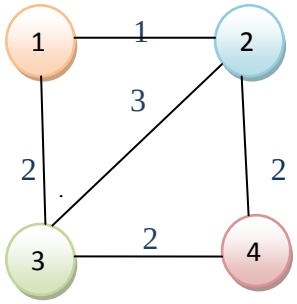


BMCP

c(1)=4,6
c(2)=3,5,1
c(3)=10
c(4)=7,8

Vred=10

Slika 2.1 Primer bojenja grafa za BMCP



BCP

c(1)=3
c(2)=1
c(3)=5
c(4)=3

Vred=5

Slika 2.2 Primer bojenja grafa za BCP

Problem pravljenja rasporeda kontrolnih i pismenih zadataka za mesec dana, polugodište ili za celu školsku godinu je BMCP. Čvorovi predstavljaju predmete, grane su minimalna rastojanja između kontrolnih ili pismenih iz različitih predmeta u određenom vremenskom periodu, dok boje predstavljaju datume (tj. dane) određene za kontrolne i pismene zadatke. Vodi se računa da učenici nemaju dva ili više pismenih zadataka u istoj nedelji i dva ili više kontrlona zadatka u toku dana iz različitih predmeta ($d(i,j)$). Takođe se vodi se računa o razmaku pismenih i kontrolnih zadataka iz jednog istog predmeta ($d(i,i)$).

3 Predloženi genetski algoritam za rešavanje problema bojenja grafa sa ograničenjima širine opsega

3.1 Reprezentacija i funkcija cilja

U implementaciji genetskog algoritma primenjeno je celobrojno kodiranje jedinki. Neka je $G = (V;E)$ graf, gde je V skup svih njegovih čvorova, a E skup svih njegovih grana. Neka je $|V|=n$ i $|E|=m$ (skup V ima n članova, a skup E ima m članova).

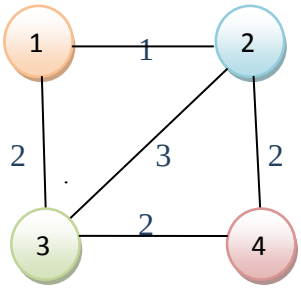
Za svaki čvor nam trebaju kandidati, pa pre genetskog algoritma se računa do kog kadidata ćemo ići. To dobijamo primenom najjednostavnijeg „pohlepnog“ (greedy) algoritma. „Pohlepni“ (greedy) algoritam se koristi samo za određivanje broja boja, ne i u genetskom algoritmu. Neka je broj boja koje nam je „pohlepni“ (greedy) algoritam izračunao t. Tada svakom od n čvorova dodeljujemo za njega zadat broj boja na taj način što prilikom dodeljivanja boja uzimamo prvi slobodan kandidat, a zatim u svakoj generaciji za svaku jedinku primenjujemo genetski algoritam. Broj gena je ukupan broj boja koje se raspoređuju, a svaki gen ima vrednost od 1, ..., t.

Svakom čvoru i mora biti dodeljen određen broj boja r_i i svaka boja mora poštovati minimalnu udaljenost $d(i,j)$ između boja ostalih čvorova j. U najsloženijem slučaju (BMCP) mora obratiti pažnja na minimalnu udaljenost $d(i,i)$ različitih boja dodeljenih jednom istom čvoru i. Ako utvrdimo da za neki čvor nemamo nijednog kandidata, onda kažemo da je jedinka nekorektna. Funkcija cilja nam daje broj boja potrebnih za bojenje.

3.2 Primeri

3.2.1 Primer genetskog algoritma za BCP

Podsetimo se, BCP (the bandwidth coloring problem) je problem bojenja čvorova grafa sa ograničenjima širine opsega gde je za svaka dva čvora udaljenost veća ili jednaka od unapred zadate minimalne udaljenosti (*slika 3.1*).



BCP

Slika 3.1 Primer bojenja grafa za BCP

U ovom primeru videćemo kako predloženi genetski algoritam raspoređuje boje na grafu. Pre samog raspoređivanja boja, prostim „pohlepnim“ (greedy) algoritmom ćemo dobiti najveći broj boja koje možemo koristiti.

„POHLEPNI“ algoritam:

I čvor: jedna boja: **1**

II čvor: jedna boja na rastojanju 1 od boje I čvora: **2**

III čvor: jedna boja na rastojanju 2 od boje I čvora i na rastojanju 3 od boje II čvora: **5**

IV čvor: jedna boja na rastojanju 2 od boje II čvora i na rastojanu 2 od boje III čvora: **7**

Dakle, maksimalan broj boja koje možemo koristiti je 7.

Sada kada znamo koliko boja možemo koristiti, za genetski kod: 3, 1, 1, 1 raspoređivanje boja predloženim genetskim algoritmom će ići ovako:

I čvor: sve boje su slobodne: 1,2,3,4,5,6,7, treća slobodna boja je 3

II čvor: slobodne su boje: 1,2,4,5,6,7, prva slobodna boja je 1

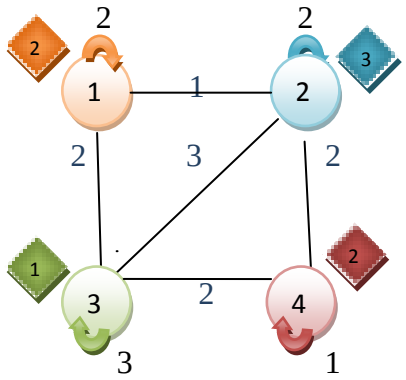
III čvor: boja kojom će biti obojen ovaj čvor mora biti na rastojanju 2 od boje I čvora i na rastojanju 3 od boje II čvora, pa su kandidati za boju ovog čvora: 5,6,7, prva slobodna je 5

IV čvor: boja kojom će biti obojen ovaj čvor mora biti na rastojanju 2 od boje II čvora i na rastojanju 2 od boje III čvora, pa su kandidati za boju ovog čvora: 3,7, prva slobodna je 3

Imamo $c(1)=3$, $c(2)=1$, $c(3)=5$, $c(4)=3$, funkcija cilja bira maksimalnu od ovih vrednosti, to je 5 (*Vred*=5), što znači da nam je 5 boja dovoljno za bojenje ovog grafa.

3.2.2 Primer genetskog algoritma za BMCP

Podsetimo se, problem bojenja čvorova grafa nizom boja sa ograničenjima širine opsega (The bandwidth multicoloring problem – BMCP) – predstavlja kombinaciju predhodna dva problema: svakom čvoru mora biti dodeljen određen broj boja i svaka boja mora poštovati minimalnu udaljenost između boja ostalih čvorova. U ovom slučaju se takođe mora obratiti pažnja na minimalnu udaljenost različitih boja dodeljenih jednom istom čvoru (*slika 3.2*).



BMCP

Slika 3.2 Primer bojenja grafa za BMCP

U ovom primeru videćemo kako predloženi genetski algoritam raspoređuje boje na grafu. Pre samog raspoređivanja boja, prostim „pohlepnim“ (greedy) algoritmom ćemo dobiti najveći broj boja koje možemo koristiti.

„POHLEPNI“ algoritam:

I čvor: dve boje na minimalnom međusobnom rastojanju 2: **1, 3**

II čvor: tri boje na minimalnom međusobnom rastojanju 2 i na rastojanju 1 od boja I čvora: **2, 4, 6**

III čvor: jedna boja na rastojanju 2 od boja I čvora (1,3) i na rastojanju 3 od boja II čvora (2,4,6): **9**

IV čvor: dve boje na minimalnom međusobnom rastojanju 1, na rastojanju 2 od boje II čvora (2,4,6) i na rastojanu 2 od boje III čvora (9): **11, 12.**

Maksimalan broj boja koje možemo koristiti je 12.

Sada kada znamo koliko boja možemo koristiti, za genetski kod: 4, 3, 3, 2, 1, 3, 1, 1 raspoređivanje boja predloženim genetskim algoritmom će ići ovako:

I čvor:

I boja: sve boje su slobodne: 1,2,3,4,5,6,7,8,9,10,11,12, četvrta slobodna boja je 4

II boja: slobodne boje su: 1,2,3,5,6,7,8,9,10,11,12, treća slobodna boja na rastojanju 2 od prve boje je 6

II čvor:

I boja: slobodne su boje: 1,2,3,5,7,8,9,10,11,12, treća slobodna boja na rastojanju 1 od boja I čvora je 3

II boja: slobodne boje: 1,5,7,8,9,10,11,12 (2 otpada je mora da bude na rastojanju 2 od dva), druga slobodna boja na rastojanju 2 od I boje i na rastojanju 1 od boja I čvora je 5

III boja: slobodne boje: 1,7,8,9,10,11,12, prva slobodna boja na rastojanju 2 od prve dve boje i na rastojanju 1 od boja I čvora je 1

III čvor:

I boja: slobodne boje: 8,9,10,11,12 (7 otpada jer mora da se razlikuje bar 2 od boja I čvora i bar 3 od boja II čvora), treća slobodna boja je 10

IV čvor:

I boja: slobodne boje: 7,8,12 (9 i 11 otpadaju jer moraju da se razlikuju bar 2 od boja III čvora), prva slobodna boja je 7

II boja: slobodne boje: 8,12, prva slobodna je 8.

Imamo $c(1)=4,6$; $c(2)=3,5,1$; $c(3)=10$; $c(4)=7,8$ funkcija cilja bira maksimalnu od ovih vrednosti, to je 10 ($Vred=10$), što znači da nam je 10 boja dovoljno za bojenje ovog grafa.

Napomena: Ako broj boja koji ostane bude manji od vrednosti gena u genetskom kodu, onda računamo ostatak pri deljenju.

3.3 Genetski operatori

3.3.1 Selekcija

Prilikom odabira roditelja koji daju potomke za sledeću generaciju koristi se mehanizam koji se naziva *operator selekcije.* Selekcijom se favorizuju dobre jedinke nad lošim. Na taj način se dobri geni prenose na sledeće populacije, a loši odumiru.

Predloženi genetski algoritam koristi fino – gradiranu turnirsku selekciju (opisanu u [Fil98], [Fil03], [Fil06]). Turniri su takmičenja dve ili više jedinki koje se nadmeću da bi učestvovale u sledećoj generaciji. Učesnici na turniru se biraju na slučajan način. Broj turnira jednak je broju jedinki, a veličina turnira N_{tur} (celobrojni parametar) se obično unapred zadaje. Jedinka se bira ako je pobedila na turniru. Međutim, veliki nedostatak turnirske selekcije je nemogućnost izbora pogodnog parametra N_{tur} . Upravo zbog ovog nedostatka često se događa da za jedan parametar konvergencija bude veoma spora, a da povećanje parametra za jedan dovede do preuranjene konvergencije, odnosno vezivanja rešenja za lokalni ekstremum. Iz ovih razlog za predloženi genetski algoritam korišćeno je unapređenje standardnog operatora turnirske selekcije, poznato ka Fino – gradirana turnirska selekcija. Umesto celobrojnog parametra N_{tur} , fino – gradirana turnirska selekcija zavisi od parametra F_{tur} - željena srednja veličina turnira, koji uzima realne vrednosti. Kao i kod turnirske selekcije, jedinka će biti izabrana ukoliko je bolja od određenog broja slučajno izabranih protivnika, ali veličina izbornih turnira nije jedinstvena u okviru populacije. Koriste se dva tipa turnira: prvi se sprovodi k_1 puta i njegova veličina je [F_{tur} +1], dok se drugi sprovodi k_2 puta, gde je broj jedinki koje učestvuju u turniru [F_{tur}]. Pri tome važi: $F_{tur} \approx \frac{k_1 \cdot \left[F_{tur} \right] + k_2 \left[F_{tur} \right]}{N_{nnel}}$, gde je $N_{nnel} = k_1 + k_2$.

U ovoj implementaciji je F_{tur} = 5.4, sa pridruženim vrednostima k_1=*30* i k_2=*20* za N_{nnel} = 50. Složenost izračunavanja ovog operatora je konstantna *O(1).*

3.3.2 Ukrštanje

Na jedinke odabrane operatorom selekcije se primenjuje operator ukrštanja. Kao što je već pomenuto operator ukrštanja je binarni operator koji se primenjuje nad jedinkama koje se nazivaju roditelji pri čemu nastaje jedna ili dve nove nove jedinke koje se nazivaju potomci. Najbitnija osobina ukrštanja je da potomci nasleđuju osobine svojih roditelja.

Pri rešavanju uopštenog problema bojenja grafa sa ograničenjima širine opsega korišćeno je jednopoziciono ukrštanje. Ono predstavlja najprostiji način ukrstanja, gde se na slučajan način bira ceo broj k iz intervala [0, l-1], gde je l dužina jedinke roditelja. Izabrani ceo broj k predstavlja tačku ukrštanja gena. Svi geni, počev od pozicije k+1, do poslednje pozicije l-1 u genetskim kodovima roditelja uzajamno menjaju mesta, pri čemu se stvaraju dva nova potomka.

U predloženom genetskom algoritmu nivo ukrštanje je 0.85, što znači da 85% parova razmeni genetski materijal.

3.3.3 Mutacija

Najvažniji operator genetskog algoritma koji menja slučajno izabrane jedinke i vraća koristan genetski materijal izgubljen u selekciji i ukrštanju naziva se mutacija. Kako se primenjuje nad samo jednom jedinkom mutacija je unarni operator.

Može se desiti da prilikom izvršavanja genetskog algoritma (skoro) sve jedinke u jednoj generaciji imaju istu vrednost na istoj bit-poziciji. Takve gene definišemo kao "zaleđene", a njih može biti više u istoj generaciji. Ako je broj zaleđenih gena l, tada pretraživački prostor postaje $l!$ puta manji pa mogućnost preuranjene konvergencije rapidno raste. Operatori selekcije i ukrštanja ne mogu promeniti vrednost zaleđenog bita. Jedino operator mutacije može promeniti vrednost zaleđenog bita, ali osnovni nivo mutacije u najvećem broju slučajeva nije dovoljan da bi se povratili izgubljeni regioni prostora pretrage. Sa druge strane ni osnovni nivo mutacije ne sme biti veliki jer se onda gubi smisao samog genetskog algoritma i on postaje slučajna pretraga (random search). Iz ovog razloga nivo mutacije se smanjuje samo na zaleđenim genima.

Sa sum_w označimo zbir težina svih čvorova, tj. ukupan broj boja koje se raspoređuju i neka je gro najveća numeracija (gornja granica) za funkciju cilja. Za nezaleđene gene prvi bit se mutira

sa 0.2/sum_w, drugi sa 0.1/sum_w itd. Za zaleđene bitove se koristi 2.5 puta veći nivo mutacije, tj prvi bit se mutira sa 0.5/sum_w, drugi sa 0.25/sum_w itd.

3.3.4 Politika zamene generacija

U predloženom genetskom algoritmu je na slučajan način izabrana populacija koja broji 150 jedinki, pri čemu se u velikoj meri obezbedila raznovrsnost genetskog materijala. Jedna trećina, odnosno 50 jedinki se zamenjuje u svakoj generaciji, dok dve trećine, odnosno 100 najboljih tvz. elitnih jedinki prelaze u sledeću generaciju. Funkcija cilja se računa samo za prvu generaciju pri čemu se štedi na vremenu izračunavanja. Ovakva politika zamene generacija je u literaturi poznat pod nazivom *stacionarni genetski algoritam sa elitističkom strategijom* ([Kra00], [Kra01]).

Da bi se izbegla mogućnost preuranjene konvergencije i da bi se obezbedila raznovrsnost genetskog materijala, sve višestruke pojave iste jedinke u populaciji se uklanjaju. Postavljajući funkciju prilagođenosti date jedinke na nulu, uklanjanje se vrši implicitno, čime ona gubi mogućnost da se pojavi u narednoj generaciji. Može se desiti da jedinke sa istom funkcijom cilja, a različitim genetskim kodovima dominiraju u populaciji. Ako su takvim jedinkama kodovi slični, genetski algoritam se može završiti u lokalnom optimumu. Zato se pojavljivanje ovakvih jedinki u populaciji ograničava na neku konstantu N_{rv}.

U implementaciji ovog genetskog algoritma broj jedinki sa istom funkcijom prilagođenosti, a različitim genetskim kodom ograničen je na 40, tj. $N_{rv}=40$.

3.3.5 Keširanje genetskog algoritma

U predloženom genetskom algoritmu u cilju poboljšanja njegovih performansi koristi se tehnika keširanja prvi put primenjena u [Kra99] i [Kra01]. Kada je u pitanju vreme izvršavanja genetskog algoritma, najveći deo vremena je potreban za je izračunavanje funkcije cilja. Nakon dobijanja vrednosti funkcije cilja, one se zajedno sa odgovarajućim genetskim kodom, smeštaju u memoriju. Za vreme izvršavanja genetskog algoritma proverava se da li se tekuća jedinka pojavljivala u prethodnim generacijama, odnosno da li postoji u keš memoriji. Ako postoji u keš memeoriji, njena funkcija cilja se ne izračunava, već se vrednost uzima iz keš memorije. U suprotnom, računa se vrednost funkcije cilja i smešta se u keš memoriju zajedno sa genetskim kodom.

Keš memorija se sastoji iz blokova, pri čemu svaki blok služi za memorisanje genetskog koda jedne jedinke i vrednosti njene funkcije cilja ili samo za očitavanje vrednosti funkcije cilja. Broj blokova u keš memoriji je ograničen i kod genetskih algoritama veličine blokova su jednake. U slučaju da se napuni keš memorija izbacuje se najstariji nekorišćeni blok.

Tehnika keširanja se zasniva na strategiji najstarijeg nekorišćenog bloka (*Least Recently Used Strategy-LRU*). Za implementaciju ove strategije koriste se heš-red (*hash-queue*) strukture (kao u [Kra00a]), u koje su smešteni pokazivači na sve blokove keš memorije i ona omogućava:

- ✓ Pronalaženje tražene jedinke uz pomoć heš tabele pretraživanjem keš memorije
- ✓ Izbacivanje najstarijeg nekorišćenog bloka iz keš memorije (korišćenjem reda) u slučaju da je keš memorija potpuno popunjena.
- ✓ Ubacivanje aktuelne jedinke u keš memoriju ako se ona već ne nalazi u keš memoriji.

Red pokazivača na keš memoriju je u skladu nalaženjem /nenalaženjem jedinki populacije u keš memoriji. Na vrhu reda se nalaze blokovi kojima se najskorije pristuplo, dok se na kraju reda nalaze blokovi kojima se pristupalo najdalje u prošlosti.

U implementaciji ovog genetskog algoritma, broj keširanih vrednosti funkcije cilja u heš-red tabeli je ograničen i iznosi *5000* ($N_{cache}=5000$).

4 Rezultati

U ovom delu prikazani su dobijeni rezultati genetskog algoritma za rešavanje uopštenog problema bojenje grafa sa ograničenjima širine opsega. Sva testiranja izvedena su na računaru sa Intel-ovim Core 2 Duo procesorom, radnog takta 2.6 GHz i 4 GB radne memorije. Algoritam je kodiran u programskom jeziku C.

Genetski algoritam je testiran na GEOM instancama iz rada [Ema03]. Sam naziv instance govori o broju čvorova koji testirani graf ima, dok se broj grana kod grafova razlikuje što pokazuje sledeća tabela:

Tabela 4.1. Prikaz broja čvorova i broja grana kod korišćenih instance

NAZIV INSTANCE	BROJ ČVOROVA GRAFA	BROJ GRANA GRAFA	NAZIV INSTANCE	BROJ ČVOROVA GRAFA	BROJ GRANA GRAFA
GEOM 20	20	40	GEOM 70b	70	558
GEOM 20a	20	57	GEOM 80	80	429
GEOM 20b	20	52	GEOM 80a	80	692
GEOM 30	30	80	GEOM 80b	80	743
GEOM 30a	30	111	GEOM 90	90	531
GEOM 30b	30	111	GEOM 90a	90	879
GEOM 40	40	118	GEOM 90b	90	950
GEOM 40a	40	186	GEOM 100	100	645
GEOM 40b	40	197	GEOM 100a	100	1092
GEOM 50	50	177	GEOM 100b	100	1150
GEOM 50a	50	288	GEOM 110	110	748
GEOM 50b	50	299	GEOM 110a	110	1317
GEOM 60	60	245	GEOM 110b	110	1366
GEOM 60a	60	339	GEOM 120	120	893
GEOM 60b	60	426	GEOM 120a	120	1554
GEOM 70	70	337	GEOM 120b	120	1611
GEOM 70a	70	529			

Za date instance u literaturi nisu poznata optimalna rešenja. U svakoj GEOM instanci genetski algoritam je bio pokrenut 20 puta. Maksimalan broj generacija u ovoj implementaciji genetskog algoritma je $N_{gen} = 5000$ Ponavljanje vrednosti najbolje funkcije cilja je ograničeno konstantom $N_{rep} = 2000$

Kolone u tabelama 4.2 i 4.3 sadreže sledeće podatke (redom):

- Ime instance **GEOM**;
- Rešenje „pohlepnog“ (greedy) algoritma $\boldsymbol{Rez_{poh}}$;
- Rešenje genetskog algoritma $\boldsymbol{Rez_{GA}}$;
- Prosečno vreme $\boldsymbol{t(s)}$ u sekundama, potrebno da bi se dobio rezultat genetskog algoritma;
- Prosečno ukupno vreme $\boldsymbol{t_{uk}}$ u sekundama, za završetak genetskog algoritma;
- Prosečan ukupan broj generacija $\boldsymbol{gen}$;
- Prosečna relativna greška $\boldsymbol{Gr}$ u procentima rešenja genetskog algoritma;
- Standardna devijacija $\boldsymbol{\sigma}$ relativne greške u procentima;
- Prosečan broj izračunavanja funkcije cilja $\boldsymbol{eval}$;
- Prosečna vrednost uštede primenom tehnike keširanja $\boldsymbol{Kеš}$ u procentima.

Tabela 4.2: Rezultati genetskog algoritma za BCP

$inst.$	Rez_{poh}	Rez_{GA}	t (s)	t_{uk} (s)	gen	Gr (%)	σ (%)	$eval$	$Keš$ (%)
GEOM20	25	21	0.091	0.682	1118.9	5.714	7.18	39410.6	27.9
GEOM20a	28	21	0.044	1.196	2079.5	0.238	1.065	68866.6	34
GEOM20b	17	14	0	0.604	2003.7	0	0	59529.9	40.7
GEOM30	34	32	0.108	0.916	975.2	0.469	1.145	35522.3	26.3
GEOM30a	32	28	0.089	2.106	2087.1	0	0	74357.8	28.9
GEOM30b	28	26	0.111	1.249	2187.3	1.538	2.9	68938.3	37
GEOM40	34	30	0.229	1.66	1195.4	7.667	3.262	44944.5	24.2
GEOM40a	49	40	0.336	3.572	2204.3	0.125	0.559	81485.6	26.2
GEOM40b	41	36	0.194	2.131	2195.9	6.806	2.774	69869.4	36.5
GEOM50	34	30	0.729	4.382	2295.3	7.5	4.171	86456.6	24.6
GEOM50a	60	53	0.704	5.796	2279.3	3.302	1.205	85238.8	25.2
GEOM50b	55	43	0.574	3.434	2397.1	2.442	1.187	79173.9	34
GEOM60	41	34	0.739	5.189	2328.9	6.176	2.682	87292.5	25.2
GEOM60a	61	55	0.603	7.24	2177.7	1.818	1.022	83484	23.4
GEOM60b	64	51	1.067	5.346	2487.8	6.961	2.807	83251.9	33.1
GEOM70	47	42	0.601	6.413	2202.6	2.024	0.872	83080.3	24.7
GEOM70a	73	65	0.817	9.424	2184.9	1.154	0.846	83626.9	23.6
GEOM70b	77	60	1.445	7.478	2462.4	2.667	1.257	83931.5	31.7
GEOM80	52	45	1.425	8.768	2382.7	6.556	2.219	90352.3	24.1
GEOM80a	80	69	3.477	13.983	2650.1	3.768	2.124	100165.4	24.4
GEOM80b	99	73	2.481	10.938	2566.7	3.425	2.061	90169.2	29.8
GEOM90	54	49	0.731	9.683	2160.4	2.551	1.605	81930.9	24.3
GEOM90a	81	75	3.728	17.239	2537.6	3.333	2.35	95989.2	24.6
GEOM90b	110	86	4.396	16.493	2657.1	5.64	2.779	93179.7	29.8
GEOM100	58	55	1.819	12.501	2331.2	3.818	1.948	90491.6	22.5
GEOM100a	97	83	3.158	20.564	2352.6	2.651	1.593	90520	23.2
GEOM100b	122	93	2.441	17.312	2307.7	3.065	1.683	80204.8	30.7
GEOM110	58	57	1.405	13.444	2230.8	2.719	1.061	85634.9	23.2
GEOM110a	98	88	7.786	28.91	2689.2	4.261	2.328	104786.1	22.4
GEOM110b	122	98	7.177	25.194	2761.7	3.214	1.851	97110.8	29.7
GEOM120	73	66	2.252	17.115	2295.9	6.212	2.021	89902.1	21.9
GEOM120a	108	100	3.134	29.815	2228.8	2.75	1.618	85770.4	23.2
GEOM120b	124	106	5.282	27.951	2450	3.726	2.28	88615.7	27.8

Tabela 4.3: Rezultati genetskog algoritma za BMCP

$inst.$	Rez_{poh}	Rez_{GA}	t (s)	t_{uk} (s)	gen	Gr (%)	σ (%)	$eval$	$Keš$ (%)
GEOM20	195	150	2.826	14.074	2485.6	2.333	1.113	84429.1	32.1
GEOM20a	201	178	4.278	18.459	2562	2.051	1.279	84382.1	34.3
GEOM20b	47	45	0.477	1.708	2741	2.444	1.596	79880.3	41.8
GEOM30	214	168	10.301	27.519	3032.2	4.702	1.93	108808.1	28.3
GEOM30a	292	234	25.074	70.929	3049.5	2.863	2.048	108979.3	28.4
GEOM30b	88	80	0.877	5.87	2335	1.438	1.167	75298.3	35.5
GEOM40	226	180	22.624	60.739	3040	3.75	2.046	109607.3	27.9
GEOM40a	297	246	42.722	109.84	3180.3	5.488	2.824	113042.6	29.1
GEOM40b	121	82	2.16	8.754	2623.1	5.732	2.276	81142.8	38.3
GEOM50	271	243	37.868	113.607	2962.2	3.992	2.157	111719.9	24.6
GEOM50a	440	370	102.564	309.143	2930.1	3.243	1.493	107111	26.8
GEOM50b	126	99	3.152	14.258	2529.9	4.747	2.541	81457.4	35.9
GEOM60	312	268	69.008	166.331	3200.7	2.836	1.688	120341.1	24.6
GEOM60a	460	399	152.433	417.723	2999.3	1.729	0.849	108722.1	27.5
GEOM60b	161	134	7.085	26.998	2677.7	3.209	1.877	88634.4	33.7
GEOM70	374	304	67.98	231.641	2782.5	5.641	2.977	105986.8	23.9
GEOM70a	577	508	262.543	601.959	3330.9	2.776	1.35	126187	24
GEOM70b	176	142	9.506	36.051	2676.5	3.768	2.122	90613.4	32.2
GEOM80	513	421	258.467	528.526	3523.1	3.717	2.199	136854.8	22.4
GEOM80a	461	411	235.69	620.676	3106.8	2.153	0.961	115513.6	25.7
GEOM80b	179	162	9.18	50.136	2414.8	3.241	1.456	81242.9	32.9
GEOM90	418	363	217.427	569.227	3057.3	1.832	1.055	117643.6	23.1
GEOM90a	485	430	388.372	845.052	3280.8	2.57	1.172	118300.6	27.9
GEOM90b	210	180	25.266	80.652	2851.6	3.278	2.101	97084	32
GEOM100	503	449	356.284	870.236	3312.9	2.383	1.451	131584	20.5
GEOM100a	626	541	521.058	1399.933	3089.8	1.59	0.93	118060.4	23.6
GEOM100b	243	198	20.4	88.188	2573	2.121	1.525	88792.4	31
GEOM110	483	434	318.342	957.523	2963.5	2.396	1.199	115519.8	22
GEOM110a	662	585	1050.278	2485.472	3421.3	2.376	1.045	129903.5	24.1
GEOM110b	297	240	43.197	132.244	2924.8	2.125	1.505	103761.4	28.9
GEOM120	515	458	453.186	1174.291	3128.8	2.904	1.727	118775.1	24.2
GEOM120a	702	649	980.06	2711.486	3075.9	1.464	0.766	113393.6	26.5
GEOM120b	268	225	43.118	153.825	2727.4	2.711	1.795	95091.5	30.4

Za problem BCP***:*** Primetimo da je značajno poboljšanje genetskog algoritma u odnosu na gornju granicu koje iznosi do 26.26%. Poboljšanje genetskog algoritma u odnosu na gornju granicu je najmanje za instacu GEOM110 i iznosi 1, procentualno 1.72%. Najveće poboljšanje između pomenuta dva rezultata je za instancu GEOM100b i iznosi 29,a najveće relativno poboljšanje je 26.26% za instancu GEOM80b.

Prosečno vreme potrebno da bi se dobio rezultat genetskog algoritma za prvih 14 instanci, tj. za grafove koji imaju maksimalno 60 čvorova, kao i za instance GEOM70, GEOM70a i GEOM90 je manje od 1 sekunde, a najduže se izvršavala instanca GEOM110a ($7.786s$), koja ima 110 čvorova i 1317 grana. Prosečno ukupno vreme za završetak genetskog algoritma je najmanje za instance GEOM20b ($0.604s$) koja ima 20 čvorova i 52 grane, a najveće za GEOM120a ($29.815s$) koja ima 120 čvorova i 1554 grana.

Najmanji prosečan broj generacija je na instanci GEOM30 ($gen = 975.2$), a najveći na GEOM110a ($gen = 2689.2$). Najveća prosečna relativna greška rešenja genetskog algoritma je za instancu GEOM40 ($Gr = 7.667\%$), a prosečne relativne greške nema na instancama GEOM20b i GEOM30a.

Standardna devijacija relativne greške σ je najveća za instance GEOM20 ($\sigma = 7.18\%$), a $\sigma = 0$ za instance GEOM20b i GEOM30a. Broj izračunavanja funkcije cilja $eval$ se kreće od 35552.3 za GEOM30 do 104786.1 za GEOM110a. Prosečna vrednost uštede primenom tehnike keširanja je najmanja za instancu GEOM120 (21.9%), a najveća za GEOM20b (40.7%).

Za problem BMCP***:*** Primetimo da je poboljšanje genetskog algoritma u odnosu na gornju granicu za ovaj problem nešto veće i kreće se do 32.23%. Poboljšanje genetskog algoritma u odnosu na gornju granicu je najmanje za instacu GEOM20b i iznosi 2, što je procentualno 4.25%. Najveće apsolutno poboljšanje je za instancu GEOM80 i iznosi 92, a najveće relativno poboljšanje je 32.23% za instancu GEOM40b.

Prosečno vreme potrebno da bi se dobio rezultat genetskog algoritma za instace GEOM20b i GEOM30b je manje od 1 sekunde, a najduže se izvršavala instanca GEOM110a ($1050.278s$), koja ima 110 čvorova i 1317 grana. Prosečno ukupno vreme za završetak genetskog algoritma je najmanje za instance GEOM20b ($1.708s$) koja ima 20 čvorova i 52 grane, a najveće za GEOM120a ($2711.486s$) koja ima 120 čvorova i 1554 grana.

Najmanji prosečan broj generacija je na instanci GEOM30b ($gen = 2335$), a najveći na GEOM80 ($gen = 3523.1$). Najveća prosečna relativna greška rešenja genetskog algoritma je za instancu GEOM40b ($Gr = 5.732\%$), a prosečna relativna greška je najmanja za instancu GEOM30b ($Gr = 1.438\%$)

Standardna devijacija relativne greške σ je najveća za instance GEOM70 ($\sigma = 2.977\%$), a $\sigma = 0.766\%$ za instancu GEOM120a. Broj izračunavanja funkcije cilja $eval$ se kreće od 79880.3 za GEOM20b do 136854.8 za GEOM80. Prosečna vrednost uštede primenom tehnike keširanja je najmanja za instancu GEOM100 (20.5%), a najveća za GEOM20b (41.8%).

Iz prikazanih eksperimentalnih rezultata jasno se vidi da je genetski algoritam postigao mnogo bolja rešenja od gornje granice za dati problem u obe tabele. Dobijeni rezultati pokazuju da je predloženi pristup vrlo robustan i efikasan u rešavanju datog problema na instancama velikih dimenzija. Vreme izvršavanja je bilo relativno kratko za oba testirana problema, gde je čak i na instancama najveće dimenzije program radio manje od pola sata.

5 Zaključak

U ovom radu je prikazan genetski algoritam za rešavanje uopštenog problema bojenja grafa sa ograničenima širine opsega koji ima veliku primenu u praksi. Osim primene u radio-difuziji, optimizacije prevoza opasnih materija, dati problemi su i direktno primeljivi i u obrazovanju. Osim što modelira pravljenje rasporeda ispita na fakultetima, uopšteni problem bojenja grafova može da se primeni i u osnovnim i srednjim školama za određivanje rasporeda pismenih vežbi i kontrolnih zadataka.

Primenjeno je modifikovano celobrojno kodiranje koje na adekvatan način oslikava prirodu problema, a sa druge strane mogućnost pojave nekoreknih jedinki je svedena na minimum. Takođe, predložen način kodiranja dozvoljava primenu standardnih operatora, a jedino je nivo mutacije na pojedinačnim genima prilagođen prirodi problema.

Dati problem je težak za rešavanje i to je jedan od razloga što postojeće egzaktne metode mogu rešiti samo instance problema manjih dimenzija. Zbog toga je predloženi evolutivni pristup vrlo značajan, jer robustnost i adaptivnost predloženih genetskih operatora omogućuju vrlo uspešno rešavanje datih problema, čak i na instancama velikih dimenzija.

Dizajnirani su i implementirani modifikovani genetski operatori, koji vode računa o načinu kodiranja i prirodi problema. Ovi operatori, između ostalih funkcija, čuvaju i korektnost jedinki, čime se isključuje pojava nekorektnih jedinki u svakoj generaciji. Primenjeni genetski operatori koji su pokazali najbolje rezultate pri rešavanju datog problema su: fino-gradirana turnirska selekcija, operator jednopozicionog ukrštanja i modifikovani operator mutacije sa zaleđenim genima. Kod implementacije genetskog algoritma keširanje u velikoj meri poboljšava performanse, ali ne utiče na ostale aspekte algoritma. Takođe, primenjene su razne metode za sprečavanje preuranjene konvergencije, gubljenja raznovrsnosti genetskog materijala i spore konvergencije genetskog algoritma. Predložena implementacija primenjuje stacionarni genetski algoritam sa elitističkom strategijom.

Za testiranje predloženog genetskog algoritma su korišćene instance problema predložene u literaturi. Na žalost za date instance nisu poznata optimalna rešenja, pa nije bilo moguće proveriti kvalitet rešenja genetskog algoritma, ali su dobijena rešenja značajno bolja od gornje granice, što

pokazje da su dobijena rešenja verovatno prilično dobrog kvaliteta. Vreme izvršavanja je bilo relativno kratko, gde je čak i na instancama najveće dimenzije program radio manje od pola sata.

Mogući pravci daljeg razvoja su:

- ✓ Rešavanje predloženih problema pomoću nekih drugih metaheuristika
- ✓ Primena datog pristupa za rešavanje nekih drugih sličnih problema, kao i modeliranje i rešavanje nekih realnih problema iz domena obrazovanja.

Najvažniji naučni doprinosi ovog rada su:

- ✓ Nov način kodiranja i odgovarajuća funkcija cilja kojom je očuvana korektnost jedinki i omogućena efikasna implementacija
- ✓ Prilagođavanje genetskih operatora datom načinu kodiranja koji odgovaraju prirodi problema primenjenim načinima kodiranja i čuvaju dopustivost rešenja.

Kao što se može videti iz eksperimentalnih rezultata, predloženi kombinatorni pristup je veoma uspešan pri rešavanju uopštenog problema bojenja grafa sa ograničenjima širine opsega. Zbog svega gore navedenog, naučno istraživanje opisano u ovom radu daje doprinos oblastima kombinatorne optimizacije, problema bojenja grafa i metaheuristika. Dobijeni rezultati su u pripremi za objavljivanje u naučnim publikacijama.

Literatura

[Aha77] **Appel K, Haken W., Koch J.** „Every planar map is four colorable“, Illinois Journal Of Mathematics, Vol. 21, pp. 439-567, 1977.

[Alv92] **Alves M., Almeida M.,** „Simulated annealing algorithm for the simple plant location problem: a computational study“, Revista Investigação, Vol. 12, 1992.

[And97] **Anderson L.,** „A simulation study of some dynamic channel assignment algorithms in a high capacity mobile telecommunications system“, IEEE Transactions On Communications, Vol. 21, pp. 1294–1301, 1973.

[Art97b] **Aerts E., Korst J., Van Laarhoven P.J.M.,** „Simulated annealing“, Local Search In Combinatorial Optimization, Aarts E.H.L. and Lenstra J.K. (eds.), John Wiley & Sons Ltd., pp. 91-120, 1997.

[Avh03] **Avanthay C., Hertz A., Zufferey N.** „A variable neighborhood search for graph coloring“, European Journal Of Operational Research, Vol. 151, No. 2, pp.379–388, 2003.

[BeD93a] **Beasley D., Bull D., Martin R.,** „An overview of genetic algorithms, part 1, fundamentals“, University Computing, Vol. 15, No. 2, pp. 58-69, 1993. ftp://ralph.cs.cf.ac.uk/pub/papers/GAs/ga_overview1.ps

[BeD93b] **Beasley D., Bull D., Martin R.,** „An overview of genetic algorithms, part 2, research topics“, University Computing, Vol. 15, No. 4, pp. 170-181, 1993. ftp://ralph.cs.cf.ac.uk/pub/papers/GAs/ga_overview2.ps

[BeJ88] **Beasley J.E.,** „An algorithm for solving large capacitated warehouse location problems“, European Journal Of Operational Research, Vol. 33, No. 3, pp. 314-325, 1988.

[BeJ90b] **Beasley J.E.,** „A Lagrangean heuristic for set-covering problems“, Naval Research Logistics, Vol. 37, pp. 151-164, 1990.

[BeJ93] **Beasley J.E.,** „Lagrangean heuristic for location problems“, European Journal Of Operational Research, Vol. 65, No. 3, pp. 383-399, 1993.

[Blz03] **Blöchliger I., Zufferey N.,** „A graph coloring heuristic using partial solutions and a reactive tabu scheme“, Computersand Operations Research, Vol. 35, No. 3, pp. 960–975, 2003.

[BoČ10] **Bouchard M., Čangalović M., Hertz A.,** „On a reduction of the interval coloring problem to a series of bandwidth coloring problems“, Springer Science+Business Media, LLC, 2010.

[Bok87] **Booker L.,** „Improving search in genetic algorithms“, Genetic Algorithms And Simulated Annealing, Pitman Publishing, London, pp. 61-73, 1987.

[Bor09] **Borak B.**, „Genetski algoritam za rešavanje lokacijskog problema snabdevača ograničenog kapaciteta u više nivoa“, Diplomski-master rad, Matematički fakultet, Beograd, 2009.

[Bow89] **Bowler P.,** „The mendelian revolution: the emergence of hereditarian concepts in modern science and society“, Baltimore Genetics: The Life Of DNA, Johns Hopkins University Press, 1989.

[Brs88] **Brassard G., Bratley P.,** „Algorithms: theory and practice“, Prentice-Hall Int., Englewoods Cliffs NJ, 1988.

[ChW87] **Chams M., Hertz A., De Werra D.,** „Some experiments with simulated annealing for coloring graphs“, European Journal Of Operational Research Vol. 32, No. 2, pp. 260–266, 1987.

[ChN71] **Christofides N.,** „An algorithm for the chromatic number of a graph“, The Computer Journal, Vol. 14, No. 1, pp. 38–39, 1971.

[ChS07] **Chiarandini M., Stützle T.,** „Stochastic local search algorithms for graph set T-colouring and frequency assignment“, Springer Science + Business Media, LLC, 2007.

[Co93] **Costa D.,** „On the use of some known methods for T-colourings of graphs“, Annals Of Operations Research, Vol. 41, pp. 343–358, 1993.

[Crm90] **Cormen T.H., Leiserson C.E., Rivest D.L.,** „Introduction to algorithms“, MIT Press, Cambridge MA, 1990.

[Čan96] **Čangalović M.,** "Opšte heuristike za rešavanje problema kombinatorne optimizacije", Kombinatorna optimizacija: Matematička teorija i algoritmi, str. 320-350, 1996.

[Dar59] **Darwin C.,** „The origin of species“, London, 1859.

[Dav91] **Davis L.,** „Handbook of genetic algorithms“, Van Nostrand Reinhold, New York, 1991.

[DBr79] **D. Brelaz,** „New methods to color the vertices of a graph“, Communications Of The ACM, Vol. 22, No. 4, pp. 251–256, 1979.

[DJo75] **De Jong K.E.,** „An analysis of the behavior of a class of genetic adaptive systems“, PhD thesis, University of Michigan, 1975.

[Dju10] **Đukić M.,** „Hibridni genetski algoritam za rešavanje hab lokacijskog problema neograničenih kapaciteta sa višestrukim alokacijama”, Diplomski-master rad, Matematički fakultet, Beograd, 2010.

[DoH98a] **Dorne R., Hao JK.,** „A new genetic local search algorithm for graph coloring“, PPSN 98, Lecture Notes In Computer Science, Berlin: Springer; Vol. 1498, pp. 745, 1998.

[DoH98] **Dorne R., Hao JK.,** „Tabu search for graph coloring, t-colorings and set t-colorings“, Voss S, editor. Meta-heuristics Advances And Trends In Local Search Paradigms For Optimization, Dordrecht: Kluwer; pp.77–92, 1998.

[DoMa96] **Dorigo M., Maniezzo V., Colorni A.,** „Ant system: optimizing by a colony of cooperating agents“, IEEE Transactions On Systems, Man And Cybernetics - Part B, Vol. 26, No. 1, pp. 29-41, 1996.

[Ema03] **Malaguti E.,** „The vertex coloring problem and its generalizations“, A.A. 2003-2006.

[ErTo09] **Eraghi A.E., Torkestani J.A., Meybodi M.R.,** „Solving the bandwidth multicoloring problem: a cellular learning automata approach“, Proceedings Of 2009 International Conference On Machine Learning And Computing, 2009.

[Fan90] **Fang L., Li T.,** „Design of competition based neural networks for combinatorial optimization“, International Journal On Neural System, Vol. 3, pp. 221-235, 1990.

[Fil00] **Filipović V., Kratica J., Tošić D., Ljubić I.,** „Fine grained tournament selection for the simple plant location problem“, Proceedings Of The 5th Online World Conference On Soft Computing Methods In Industrial Applications - WSC5, pp. 152-158, 2000.

[Fil01] **Filipović V., Tošić D., Kratica J.,** „Experimental results in applying of fine grained tournament selection“, Proceedings Of The 10th Congress Of Yugoslav Mathematicians, Belgrade, pp. 331-336, 2001.

[Fil03] **Filipović V., „**Fine-grained tournament selection operator in genetic algorithms“, Computing and Informatics, Vol. 22, No. 2, pp. 143-161, 2003.

[Fil06] **Filipović V.,** „Operatori selekcije i migracije i WEB servisi kod paralelnih evolutivnih algoritama“, Doktorska disertacija, Matematički fakultet, Beograd, 2006.

[Fil97] **Filipović V.,** „Određivanje performansi genetskih algoritama u teoriji i praksi“, Prolećna škola o programskim jezicima, Institut za Matematiku PMF, Novi Sad, str. 131-141, 1997.

[Fil98] **Filipović V.,** „Predlog poboljšanja operatora turnirske selekcije kod genetskih algoritama“, Magistarski rad, Univerzitet u Beogradu, Matematički fakultet, 1998.

[FlFe96] **Fleurent C., Ferland J.A.,** „Object-oriented implementation of heuristic search methods for graph coloring, maximum clique, and satisfiability“, American Mathematical Society, 1996.

[GaHa99] **Galinier P., Hao J.K.,** „Hybrid evolutionary algorithms for graph coloring“, Journal Of Combinatorial Optimization, Vol. 3, No. 4, pp. 379–397, 1999.

[Gahz08] **Galinier P., Hertz A., Zufferey N.,** „An adaptive memory algorithm for the k-coloring problem“, Discrete Applied Mathematics, Vol. 156, No. 2, pp. 267–279, 2008.

[GaJo79] **Garey M.R., Johnson D.S.,** „Computers and intractability: a guide to the theory of NP-completeness“, DIMACS Series In Discrete Mathematics And Theoretical Computer Science, Freedman, New York, 1979.

[Gar79] **Garey M.R., Johnson D.S.,** „Computers and intractibility: a guide to the theory of NP-completness“, W.H. Freeman and Co., New York, 1979.

[Glo77] **Glover F.,** „Heuristics for integer programming using surogate constraints“, Decision Sciences, Vol. 8, No. 1, pp. 156-166, 1977.

[Glo90] **Glover F.,** „Tabu search: a tutorial“, Interfaces, Vol. 20, pp. 74-94, 1990.

[Glo97] **Glover F., Laguna F.,** „Tabu search“, Kluwer Academic Publishers, Norwell, MA, USA, 1997.

[Glv93] **Galvao R.D.,** „The use of Lagrangean relaxation in the solution of uncapacitated facility location problems“, Location Science, Vol. 1, No. 1, pp. 57-79, 1993.

[Gol89] **Goldberg D.E.,** „Genetic algorithms in search, optimization and machine learning“, Addison-Wesley Publ. Comp., Reading, Mass., pp. 412, 1989.

[Gui88] **Guignard M.,** „A Lagrangean dual ascent algorithm for simple plant location problems“, European Journal Of Operational Research, Vol. 35, pp. 193-200, 1988.

[Han01] **Hansen P., Mladenović N.,** „Variable neighborhood search: principles and applications“, European Journal Of Operational Research, Vol. 130, pp. 449-467, 2001.

[Han07] **Hansen P., Brimberg J., Urošević D., Mladenović N.,** „Primal-dual variable neighborhood search for the simple plant-location problem“, INFORMS Journal On Computing, Vol. 19, pp. 552-564, 2007.

[Han99] **Hansen P., Mladenović N.,** „An introduction to variable neighborhood search“, metaheuristics: advances and trends in local search paradigms for optimization”, Voss S., Martello S., Osman I.H., Roucairol C. (eds.), Kluwer Academic Publishers, pp. 433-458, 1999.

[Hdog98] **Hao J.K., Dorne R., Galinier P.,** „Tabu search for frequency assignment in mobile radio networks“, Journal Of Heuristics, Vol. 4, pp. 47–62, 1998.

[HePl08] **Hertz A., Plumettaz A., Zufferey N.,** „Variable space search for graph coloring“, Discrete Applied Mathematics, Vol. 156, No. 13, pp. 2551–2560, 2008.

[Her97] **Hertz A., Taillard E., De Werra D.,** „Tabu search“, Local Search In Combinatorial Optimization, Aarts E.H.L. and Lenstra J.K. (eds.), John Wiley & Sons Ltd., pp. 121-136, 1997.

[HeWe87] **Hertz A., De Werra D.,** „Using tabu search techniques for graph coloring“, Computing, Vol. 39, No. 4, pp. 345–351, 1987.

[Hll75] **Holland J.H.,** „Adaptation in natural and artificial systems“, The University Of Michigan Press, Ann Arbor, 1975.

[JaMa91] **Johnson D.S., Aragon C.R., McGeoch L.A., Schevon C.,** „Optimization by simulated annealing an experimental evaluation; part II, graph coloring and number partitioning“, Operations Research, Vol. 39, No. 3, pp. 378–406, 1991.

[Joh96] **Johnson D.S., Trick M.,** „Cliques, coloring and satisfiability second DIMACS implementation challenge, DIMACS series in discrete mathematics and theoretical computer science“, American Mathematical Society, Vol. 26, pp. 619–52, 1996.

[Kno89] **Knox J.,** „The application of TABU search to the symmetric traveling salesman problem“, PhD dissertation, University of Colorado, 1989.

[Kom96] **Cvetković D., Čangalović M., Dugošija Đ., Kovačević- Vujičić V., Simić S., Vuleta J.,** „Kombinatorna optimizacija – matematička teorija i algoritmi”, Društvo operacionih istraživača Jugoslavije, Beograd, 1996.

[Kra00] **Kratica J.,** „Paralelizacija genetskih algoritama za rešavanje nekih NP-kompletnih problema“, Doktorska disertacija, Matematički fakultet, Beograd, 2000.

[Kra96b] **Kratica J., Radojević S., Filipović V., Šćepanović A.,** „Primena epsilont ransformacije u problemu pretrage drveta“, Međunarodni naučno-razvojni simpozijum: Stvaralaštvo kao uslov privrednog razvoja - Nove tehnologije i tehnike u službi čoveka u štampi, Beograd, 1996. http://alas.matf.bg.ac.yu/~kratica/ntt96.pdf

[Kra97a] **Kratica J.,** „Napredne tehnike genetskih algoritama i njihova implementacija“, Prolećna škola o programskim jezicima, Institut za Matematiku PMF, Novi Sad, str. 123-130, 1997.

http://alas.matf.bg.ac.yu/~kratica/pspj97.pdf

[Krp83] **Krarup J., Pruzan P.,** „The simple plant location problem: survey and synthesis“, European Journal Of Operational Research, Vol. 12, pp. 36-81, 1983.

[Lam88] **Lam J.,** „An efficient simulated annealing schedule“, PhD dissertation, Yale University, 1988.

[LiL05] **Lim A., Lou Q., Rodrigues B., Zhu Y., „**Heuristic methods for graph coloring problems“, Proceedings Of The 2005 ACM Symposium On Applied Computing, Santa Fe, pp. 933–939, 2005.

[LüHa10] **Lü Z., Hao J.K.,** „A memetic algorithm for graph coloring“, European Journal Of Operational Research, Vol. 203, No. 1, pp. 241–250, 2010.

[MaGo09] **Marti R., Gortazar F., Duarte A.,** „Heuristics for the bandwidth coloring problem“, 2009.

[Man91] **Manber U.,** „Introduction to algorithms: a creative approach“, Addison Wesley Publ. Comp., Reading, MA, 1991.

[Mar08] **Marić M.,** „Rešavanje nekih *NP*–teških hijerarhijsko–lokacijskih problema primenom genetskih algoritama“, Doktorska disertacija, Matematički fakultet, Beograd, 2008.

[MaTo90] **Martello S., Toth P.,** „Knapsack problems: algorithms and computer implementations“, John Wiley & Sons, Chichester, 1990.

[MaTo08] **Malaguti E., Toth P.,** „An evolutionary approach for bandwidth multicoloring problems“, European Journal Of Operational Research, Vol. 189, pp. 638–651, 2008.

[Mic96] **Michalewicz Z.,** „Genetic algorithms + data structures = evolution programs“, Third Edition, Springer Verlag, Berlin Heideleberg, 1996.

[Mis05] **Misevicius A.,** „A tabu search algorithm for the quadratic assignment problem“, Computational Optimization And Applications, Vol. 30, pp. 95-111, 2005.

[Mit96] **Mitchell M.,** „An introduction to genetic algorithms“, MIT Press, 1996.

[Mla95] **Mladenović N.,** „A variable neighborhood algorithm - a new metaheuristics for combinatorial optimization“, Abstracts Of Papers Presented At Optimization Days, Montreal, 1995.

[Mla97] **Mladenović N., Hansen P.,** „Variable neighborhood search“, Computers Operations Research, Vol. 24, pp. 1097-1100, 1997.

[MoTo08] **Malaguti E., Monaci M., Toth P.,** „A metaheuristic approach for the vertex coloring problem“, INFORMS Journal On Computing, Vol. 20, No. 2, pp. 302, 2008.

[Müh97] **Mühlenbein H.,** „Genetic algorithms“, Local Search In Combinatorial Optimization, Aarts E.H.L., Lenstra J.K., (eds.), John Wiley & Sons Ltd., pp. 137-172, 1997.

[NPc98] **Kann V., Crescenzi P.,** „A compendium of NP optimization problems”, August 31, 1998.

http://www.nada.kth.se/viggo/index-en.html

[Ognj04a] **Ognjanović Z., Krdžavac N.,** „Uvod u teorijsko računarstvo“, Fakultet organizacionih nauka, Beograd, 2004.

[Opr04] **Krčevinac S., Čangalović M., Kovačević-Vujičić V., Martić M., Vujošević M.,** „Operaciona istraživanja”, Fakultet organizacionih nauka, Beograd, 2004.

[Pap82] **Papadimitriou C.D., Steiglitz K.,** „Combinatorial optimisation: algorithms an complexity“, Prentice-Hall, Englewood Cliffs, NJ, 1982.

[Pau97] **Paunić Đ.,** „Strukture podataka i algoritmi“, Univerzitet u Novom Sadu, Prirodno-Matematički fakultet, Novi Sad, 1997.

[Pem96] **Pemberton J.C., Zhang W.,** „Epsilon-transformation: exploiting phase transitions to solve combinatorial optimization problems“, Artificial Intelligence, Vol. 81, pp. 297-325, 1996.

[PhSk02] **Phan V., Skiena S.,** „Coloring graphs with a general heuristic search engine“, Computational Symposium On Graph Coloring And Its Generalization, Ithaca, NY, pp. 92–99, 2002.

[PoHa10] **Porumbel C.D., Hao J.K., Kuntz P.,** „A search space „cartography“ for guiding graph coloring heuristics“, Computers And Operations Research, Vol. 37, pp. 769–778, 2010.

[PoHk10] **Porumbel D., Hao J.K., Kuntz P.,** „An evolutionary approach with diversity guarantee and well-informed grouping recombination for graph coloring“, Computers And Operations Research, Vol. 37, 2010.

[ReRi03] **Resende, M.G.C., Ribeiro C.C.,** „Greedy randomized adaptive search procedures”, Handbook Of Metaheuristic, Kluwer Academic Publishers, pp.219–249, 2003.

[Spre02] **Prestwich S.,** „Constrained bandwidth multicoloration neighborhoods“, Computational Symposium On Graph Coloring And Its Generalization, Ithaca, NY, pp. 126-133, 2002.

[Sta04] **Stanimirović Z.,** „Rešavanje nekih diskretnih lokacijskih problema primenom genetskih algoritama”, Magistarski rad, Matematički fakultet, Beograd, 2004.

[Sum02] **Suman B.,** „Multiobjective simulated annealing - a metaheuristic technique for multiobjective optimization of a constrained problem“, Foundations Of Comput Decision Sci, Vol. 27, pp. 171–191, 2002.

[Sum06] **Suman B., Kumar P.,** „A survey of simulated annealing as a tool for single and multiojbective optimization“, Journal Of The Operational Research Society, Vol. 57, pp. 1143–1160, 2006.

[Tošić04] **Tošić D., Mladenović N., Kratica J., Filpović V.,** „Genetski algoritmi“, Matematički institut SANU, Beograd, 2004.

[Tošić97] **Tošić D.,** „Pregled razvoja i opis osnovnih karakteristika evolucionih (genetskih) algoritama“, Prolećna škola o programskim jezicima, Institut za Matematiku PMF, Novi Sad, str. 115-122, 1997.

[TrYi07] **Trick M.A., Yildiz H.,** „A large neighborhood search heuristic for graph coloring“, CPAIOR 2007, Lecture Notes In Computer Science, Berlin: Springer, Vol. 4510, pp. 346-360, 2007.

[TsVo98] **Tsang E., Voudouris E.,** „Solving the radio link frequency assignment problem using guided local search“, NATO Symposium On Radio Length Frequency Assignment,Aalborg,Denmark,1998.
http://cswww.essex.ac.uk/CSP/papers.html

[Uro96] **Urošević D.,** „Algoritmi u programskom jeziku C“, Mikro knjiga, Beograd, 1996.

[Vhu99] **Velanzuela C, Hurley S., Smith D.,** „A permutation based genetic algorithm for the minimum span frequency assignment“, Lecture Notes In Computer Science, Vol. 1498, pp. 907–916, 1999.

[WaRu96] **Wang W., Rushforth C.K.,** „An adaptive local-search algorithm for the channel-assignment problem (CAP)“, IEEE Transactions On Vehicular Technology, Vol. 45, pp. 459–466, 1996.

[WePo67] **Welsh D.J.A., Powell M.B., „**An upper bound for the chromatic number of a graph and its application to timetabling problems“, The Computer Journal, Vol. 10, No. 1, pp. 85-86, 1967.

[Ynn97] **Yannakakis M.,** „Computational complexity“, Local Search In Combinatorial Optimization, Aarts E.H.L. and Lenstra J.K. (eds.), John Wiley & Sons Ltd., pp. 19-56, 1997.

[Yur94] **Yuret D.,** „From genetic algorithms to efficient optimization“, MSc Thesis, Massachusetts Institute Of Technology, Department Of Electrical Engineering AndComputerScience,1994.
http://www.dai.ed.ac.uk/groups/evalg/Local_Copies_of_Papers/Yuret.MSc_
Thesis.From_Genetic_Algorithms_to_Efficient_Optimization.ps.gz

[Zha96] **Zhang W., Korf R.E.,** „A study of complexity transitions on the asymmetric traveling salesman problem“, Artificial Intelligence, Vol. 81, pp. 223-239, 1996.

[ZoBe77] **Zoellner J.A., Beall C.L.,** „A breakthrough in spectrum conserving frequency assignment technology“, IEEE Transactions On Electromagnetic Compatibility, Vol. 19, pp. 313–319, 1977.

Sadržaj