UNIVERZITET U BEOGRADU

MATEMATIČKI FAKULTET

Jasmina Fijuljanin

GENETSKI ALGORITAM ZA REŠAVANJE OUPŠTENOG 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

PREVOD!!!

PREDGOVOR

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 funkcije f na skupu S, tj. rešiti zadatak:

$$\min_{x \in S} f(x) \qquad (1)$$

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 $f(x^{\circ}) = \min f(x)$. 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 ([KomOp]).

Ako, kao u pomenutom primeru, ne postoji efikasna egzaktna metoda pribegava se približnom rešavanju problema uz pomoć heurističkih metoda: genetski algoritmi (genetic algorithms - GA), simulirano kaljenje (simulated annealing), neuralne mreže (neural networks), tabu pretraživanje (tabu search) i Lagrangeova relaksacija.

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 memeorijskog 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 1.5 Problem pripada klasi složenosti **P**, i nazivamo ga problemom *polinomske složenosti*, ako se rešava, u opštem slučaju, nekim od poznatih algoritama polinomske složenosti.

Definicija 1.6 Problem pripada klasi **NP**, i nazivamo ga problemom *nedeterminističke polinomske složenosti*, ako se algoritmom polinomske složenosti može veifikovati. 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 [KomOp], [Pap82], [Ynn97].

1.1.2 NP-kompletni problemi

Definicija 7: Za problem Q_1 kažemo da je svodiv u polinomskom vremenu na problem Q_2 ako postoji algoritam polinomske složenosti koji pretvara svaku interpretaciju problema Q_1 u interpretaciju problema Q_2 tako da imaju analogno zajedničko rešenje.

Definicija 8: Problem odlučivanja nazivamo ***NP****-kompletnim* ako:

- pripada klasi NP
- svi ostali NP problemi se algoritmom polinomske složenosti mogu 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.

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 je dokazano da je **NP**-kompletan 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 [OprIs] . Ono što u okviru klasifikacije treba pomenuti je konstruktivna metoda i u okviru nje princip "proždrljivosti"("greedy") kod koga se u svakoj iteraciji od trenutno mogućih izbora bira onaj koji je "najbolji" po nekom lokalnom kriterijumu. Ovaj "greedy" algoritam je korišćen u predloženom genetskom algoritmu u poglavlju 3.
Od sredine 80-tih godin razvija se i primenjuje niz tvz. *opštih heuristika* koja daju opšta uputstva za rešavanje problema, nezavisno od structure 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 austavljanja. 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], [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], [Kov08], [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.

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 dati su osnovni koraci genetskog algoritma:

```
Unošenje_Ulaznih_Podataka();

Generisanje_Početne_Populacije();

while(!Kriterijum_Zaustavljanja_Genetskog_Algoritma())

        {

           for (i=1;i<=N_populacije;i++ )
```

```
            pi=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 [HII75], 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 su 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.

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. Jedinka će biti izabrana ukoliko je bolja od nekoliko slučajno izabranih protivnika. Broj turnira jednak je broju jedinki, a veličina turnira N_{tur} (celobrojni parametar) se obično unapred zadaje. Jedinke se u svakom turniru upoređuju i bira se najbolja koja će se učestvovati u ukrštanju. 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:

??? $F_{tur} \approx \frac{k1\cdot\lfloor F_{tur}\rfloor + k2\cdot[F_{tur}]}{Nnnel}$, 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 deca. Najbitnija osobina ukrštanja je da deca 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 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.

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 nit iste dužine kao genetski kod jedinki, tvz. „masku“. 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. Ovaj operator proverava da li je došlo do mutacije. 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 binarnu 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: $F(Xmut) \leq F(s) < F(Xmut + 1)$, gde je F funkcija raspodele.

Binomna raspodela aproksimira normalnom raspodelom

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

kada je proizvod dužine genetskog koda i nivoa mutacije dovoljno veliki.
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 direktni 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].

[KomOp] **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.

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

[Brs88] **Brassard G., Bratley P.**, "Algorithms: Theory and Practice", Prentice-Hall Int., Englewoods Cliffs NJ (1988).

[Crm90] **Cormen T.H., Leiserson C.E., Rivest D.L.**, "Introduction to Algorithms", MIT Press, Cambridge MA (1990).

[Man91] **Manber U.**, "Introduction to Algorithms: A Creative Approach", Addison Wesley Publ. Comp., Reading, MA (1991).

[Uro96] **Urošević D.**, "Algoritmi u programskom jeziku C", Mikro knjiga, Beograd (1996).

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

[Ognj04a] **Ognjanović Z., Krdžavac N.,** "Uvod u teorijsko računarstvo", Fakultet organizacionih nauka, Beograd (2004).

[Pap82] **Papadimitriou C.D., Steiglitz K.**, "Combinatorial Optimisation: Algorithms and Complexity", Prentice-Hall, Englewood Cliffs, NJ (1982).

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

[Gar79] **Garey M.R.**, **Johnson D.S.**, "Computers and Intractibility: A Guide to the Theory of NP Completness", W.H. Freeman and Co. (1979).

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

[Lam88] **Lam J.**, "An Efficient Simulated Annealing Schedule", PhD dissertation, Yale University (1988).

[Alv92] **Alves M.L., Almeida M.T.**, "Simulated Annealing Algorithm for the Simple Plant Location Problem: A Computational Study", *Revista Investigação*, Vol. 12 (1992).

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

[Sum02] **Suman B.**, "Multiobjective simulated annealing—a metaheuristic technique for multiobjective optimization of a constrained problem", Foundations of Comput Decision Sci 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 57 pp. 1143–1160 (2006).

[Krp83] **Krarup J., Pruzan P. M.**, "The simple plant location problem: Survey and synthesis", European Journal of Operational Research, Vol. 12, pp. 36-81 (1983).

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

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

[Glo77] **Glover, F.,** "Heuristics for Integer Programming Using Surogate Constraints", Decision Sciences, Vol. 8 (1), pp. 156-166 (1977).

[Glo90] **Glover F.**, "Tabu search: A Tutorial", Interfaces 20, pp. 74-94 (1990).

[Glo97] **F. Glover, F. Laguna**, "Tabu search", Kluwer Academic Publishers, Norwell, MA, USA (1997).

[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).

[Gui88] **Guignard M.**, "A Lagrangean dual ascent algorithm for simple plant location problems", *European Journal of Operational* Research, Vol. 35, pp. 193-200 (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).

[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).

[Han99] **Hansen P., Mladenović N.**, "An Introduction to Variable Neighborhood Search", In: *Meta-Heuristics: 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).

[Han01] **P. Hansen and N. Mladenović**, "Variable neighborhood search: Principles and applications," European Journal of Operational Research 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 19, pp. 552-564 (2007).

[Kov08] **Kovačević, J.**, "Hybrid Genetic Algorithm For Solving The Low-Autocorrelation Binary Sequence Problem", Yugoslav Journal of Operations Research (2008).

[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).

[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).

[Dor96] **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 (February 1996).

[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).

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

[Kra96b] **Kratica J., Radojević S., Filipović V., Šćepanović A.**, "Primenaepsilont 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*

[Dar59] **Darwin C.**, "The origin of species", London (1859).

[Bow89] **Bowler P.**, "The Mendelian Revolution: The Emergence of Hereditarian Concepts in Modern Science and Society", BaltimoreGenetics: The life of DNA Johns Hopkins University Press (1989).

[Hll75] **Holland J.H.**, "Adaptation in Natural and Artificial Systems", The University of Michigan Press, Ann Arbor (1975).

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

[Bok87] **Booker L.**, "Improving Search in Genetic Algorithms", In: *Genetic Algorithms and Simulated Annealing*, Pitman Publishing, London, pp. 61-73(1987).

[Gol89] **Goldberg D.E.**, "Genetic Algorithms in Search, Optimization and Machine Learning", Addison-Wesley Publ. Comp., Reading, Mass., 412 pp (1989).

[Dav91] **Davis L.**, "Handbook of Genetic Algorithms", Van Nostrand Reinhold, New York (1991).

[BeD93a] **Beasley D.**, **Bull D.R.**, **Martin R.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.R.**, **Martin R.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

[Yur94] **Yuret D.**, "From Genetic Algorithms to Efficient Optimization", *MSc Thesis*, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science (1994).
http://www.dai.ed.ac.uk/groups/evalg/Local_Copies_of_Papers/Yuret.MSc_ Thesis.From_Genetic_Algorithms_to_Efficient_Optimization.ps.gz

[Mic96] **Michalewicz Z.**, "Genetic Algorithms + Data Structures = Evolution Programs", Third Edition, Springer Verlag, Berlin Heideleberg (1996).

[Mit96] **Mitchell M.,** "An Introduction to Genetic Algorithms", MIT Press (1996).

[M**ü**h97] **Mühlenbein H.**, "Genetic algorithms", *Local Search in Combinatorial* Optimization, eds. Aarts E.H.L., Lenstra J.K., John Wiley & Sons Ltd., pp. 137-172 (1997).

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

[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).

[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

[Toš97] **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).

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

[BeD93a] **Beasley D.**, **Bull D.R.**, **Martin R.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*

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

[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, pp. 331-336, Belgrade, 21.-24.01. (2001).

[Fil03] Filipović, V. "Fine-Grained Tournament Selection Operator in Genetic Algorithms", Computing and Informatics 22(2), 143-161.(2003).

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

[Toš04] **Tošić D., Mladenović N., Kratica J., Filpović V.,** "Genetski algoritmi", Matematički institut SANU, Beograd (2004).