# 4. Genetički algoritam

## *4.1. Uvod*

U ovom poglavlju je predstavljen genetički algoritam (*genetic algorithm* - GA) i ideja na kojoj se on zasniva. Ukratko su predstavljeni osnovni elementi koji čine GA, nakon toga je predstavljen GA sa realnim kodiranjem, te dat osvrt na kompleksnije šeme kodiranja hromozoma. Predstavljeni su i operatori ukrštanja i mutacije koji se koriste kod takvih varijanti GA. Isto tako, predstavljene su i neke od varijanti GA koje se često susreću, te trenutno aktuelni pristupi teorijskoj analizi GA.

## *4.2. Razvoj genetičkog algoritma*

Problem traženja optimalnog (najboljeg) rješenja za svakodnevne probleme prati ljudsku vrstu praktično od njenog nastanka. Kako je već više puta napomenuto u prethodnim poglavljima, do danas se razvio niz vrlo jakih metoda baziranih na matematičkom aparatu, kao i veliki broj numeričkih metoda koje se mogu primjeniti na određene klase problema. Međutim, još uvijek izazov predstavljaju problemi za koje su takve metode ili neprimjenjive ili neefikasne. Rečeno je da klasične metode optimizacije postavljaju prilično grube zahtjeve u pogledu modalnosti funkcije, njene glatkosti, konveksnosti skupa dozvoljenih vrijednosti u prostoru stanja i slično. Nezadovoljenje samo jednog od pomenutih zahtjeva može bitno degradirati performansu algoritma optimizacije (multimodalnost kriterija uslovljava da će gradijentni algoritmi optimizacije najvjerovatnije konvergirati ka lokalnom ekstremumu) ili potpuno isključuje mogućnost njegove primjenene (npr. klasične numeričke metode bazirane na gradijentu traže glatkost kriterijalne funkcije).

U slučajevima kada klasične determinističke metode optimizacije nisu primjenjive, najčešće se primjenjuju metode pretraživanja prostora stanja bazirane na slučaju. Ove metode ne garantuju konvergenciju ka globalnom optimumu, a za pretraživanje velikih prostora su izrazito neefikasne.

Posljednjih desetljeća veliko interesovanje izazivaju tehnike optimizacije bazirane na evoluciji, kao prirodnom procesu koji u uslovima krajnje neodređenosti permanentno poboljšava karakteristike postojećih bioloških organizama i postepeno kreira nove organizme, uz stalnu težnju za poboljšanjem. Mehanizam evolucije je na sistematičan način prvi predstavio Charles Darwin [1]: *"Može se metaforično reći da prirodno odabiranje svakodnevno i svakoga časa istražuje po cijelome svijetu i najmanje varijacije; ono odbacuje*

*loše, a održava i sabire one koje su dobre; ono radi mirno i neprimjetno, kad god i gdje god se ukaže prilika, na usavršavanju svakog organskog bića u odnosu na njegove organske i neorganske uslove života..."* Ovaj princip je postao ideja na kojoj su zasnovani svi evolucioni algoritmi, čiji je jedan od u praksi najzastupljenijih predstavnika genetički algoritam.

Genetički algoritam je razvio John Holland [2] zajedno sa svojim kolegama i studentima na Univerzitetu Michigen, sa ciljem istraživanja fenomena adaptacije u prirodi, iako se sama ideja javila par desetljeća ranije. Prema definiciji jednog od svojih tvoraca, Davida Goldberga [3], GA predstavlja stohastički algoritam organizovanog pretraživanja baziran na mehanizmu prirodne selekcije i prirodne genetike. GA kombinuje princip opstanka najboljih jedinki (predstavljenih u formi binarnog niza) sa strukturiranom ali još uvijek prilično slučajnom razmjenom informacija između pojedinačnih jedinki. Na taj način se tvori algoritam pretraživanja koji efikasno iskorištava informaciju prikupljenu u populaciji i kreira nove tačke u prostoru pretraživanja, koje u prosjeku imaju bolje performanse. Do danas je razvijen veliki broj različitih varijacija GA, sve sa ciljem poboljšanja njegove performanse. Jedan broj verzija GA teži ka uvođenju dodatnih elemenata zasnovanih na otkrićima u okviru prirodne genetike (diploidnost, različite šeme ukrštanja i mutacije iz prirode, životni vijek, migracije i sl.). Druge verzije idu dalje od modeliranja prirodnih pojava i procesa i uvode elemente i mehanizme koji za efekat imaju bitno poboljšanje performanse, bez postojanja ekvivalenata tih elemenata i mehanizama u prirodi.

## *4.3. Elementi genetičkog algoritma*

Genetički algoritam (slika 4.1) oponaša evoluciju tako što na populaciju vještačkih jedinki, od kojih svaka predstavlja tačku iz prostora stanja koja je potencijalno rješenje problema, primjenjuje operatore koji pokušavaju stvoriti nove jedinke sa poboljšanom performansom.

Svaka od vještačkih jedinki se kodira u formu hromozoma i pretraživanje se izvodi na hromozomima primjenom operatora GA. Najvažniji operatori koji se susreću gotovo u svakom GA su ukrštanje i mutacija. Hromozomi na koje se primjenjuju ovi operatori se biraju tako da oni sa većom vrijednošću fitnessa imaju statistički veću šansu da budu izabrani.

Kod većine implementacija GA (tzv. generacioni GA), operatori se primjenjuju u ciklusima na sve jedinke (slika 4.2). Svaki od ciklusa predstavlja jednu generaciju.

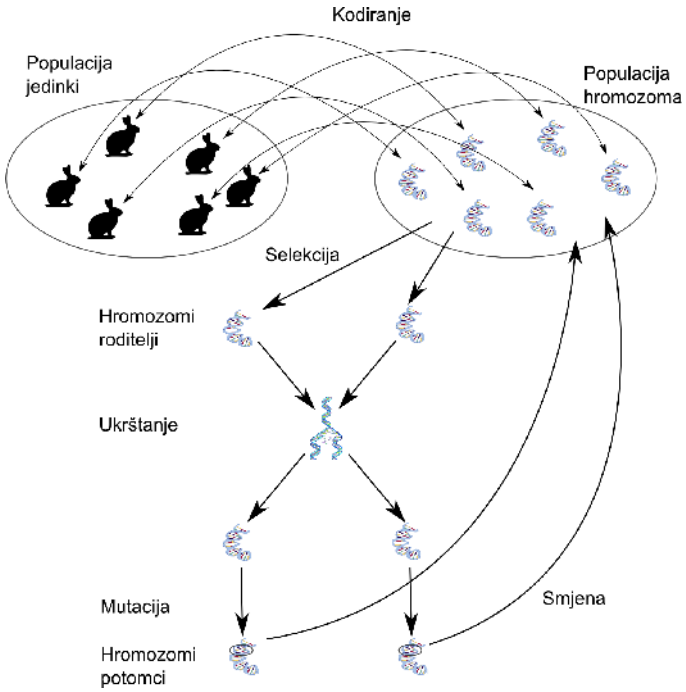
Izvođenje GA započinje generiranjem početne populacije hromozoma. Početna populacija se najčešće generira slučajno, ali ju je moguće generirati i koristeći heuristiku koja će poboljšati performansu algoritma. Za svaki od generiranih hromozoma se odredi vrijednost fitnessa, koji predstavlja mjeru kvaliteta potencijalnog rješenja, radi čega je neophodno predstavu potencijalnog rješenja u formi

hromozoma dekodirati u tačku problemskog prostora.

Bez obzira na raznovrsnost operatora i drugih elemenata koji mogu postojati u GA, može se reći da su osnovni elementi genetičkog algoritma:

- Populacija hromozoma,
- Šema kodiranja hromozoma,
- Mehanizam evaluacije fitnessa,
- Mehanizam selekcije,
- Ukrštanje,
- Mutacija,
- Mehanizam smjene.

Obzirom da svaki od ovih elemenata bitno određuje i karakteristike GA, svaki od njih je u nastavku ukratko razmotren.

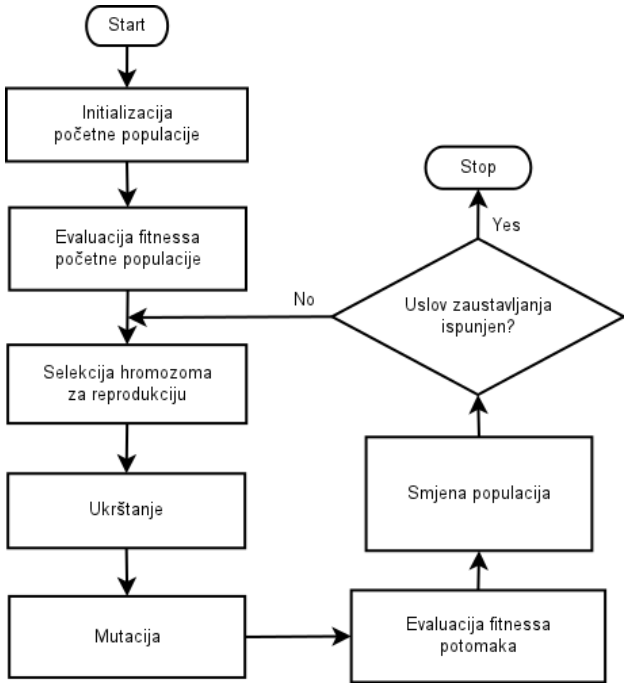



Slika 4.1. Osnovna ideja genetičkog algoritma

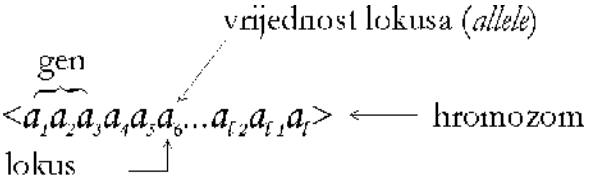
### *4.3.1. Populacija hromozoma*

Pod pojmom hromozoma [4] podrazumijevamo kodiranu jedinku (dakle tačku prostora stanja) predstavljenu, korištenjem podesne šeme kodiranja, nizom elemenata $<a_1a_2...a_l>$ određenog alfabeta $\mathbf{A}$, odnosno $a_i \in \mathbf{A}$. Jednostavni genetički algoritam (*Simple Genetic Algorithm* - SGA) za predstavljanje

hromozoma koristi binarni alfabet $\boldsymbol{A}=\{0, 1\}$, dok GA sa realnim kodiranjem za predstavljanje hromozoma koristi realne brojeve. Hromozom predstavlja osnovni objekat na koji se primjenjuju mehanizmi i operatori GA, a sastoji se od niza gena, kao elemenata koji kodiraju određenu spoljašnju osobinu jedinke (slika 4.3). Svaka pozicija (*locus*) hromozoma sadrži element alfabeta koji kodira određenu vrijednost lokusa (*allele*).




Slika 4.2. Osnovna organizacija genetičkog algoritma




Slika 4.3. Struktura hromozoma

Populaciju $\mathbf{P}$ čini određen i za većinu GA konstantan broj hromozoma $\boldsymbol{a}=<a_1a_2...a_l>$, odnosno $\mathbf{P}=\{\boldsymbol{a}^1, \boldsymbol{a}^2, ..., \boldsymbol{a}^m\}$, gdje $m$ predstavlja broj hromozoma u populaciji. Na hromozome populacije $k$-te generacije $\mathbf{P}^k$ se primjenjuju operatori genetičkog algoritma, na osnovu čega se formira populacija $k+1$-ve generacije $\mathbf{P}^{k+1}$.

### *4.3.2. Šema kodiranja hromozoma*

Kako je prethodno navedeno, hromozom $\boldsymbol{a}=<a_1a_2...a_l>$ se sastoji od niza gena, od kojih svaki kodira određenu vrijednost spoljašnje osobine (vrijednost parametra) jedinke. Cjelokupan genetski materijal

jedinke (svi hromozomi jedinke uzeti zajedno) čini genom jedinke. Treba napomenuti da se kod većine implementacija GA koje su u praktičnoj upotrebi genom jedinke sastoji od jednog hromozoma, tako da se praktično termini hromozom i jedinka koriste kao sinonimi. Operatori GA se primjenjuju na nivou genotipa, odnosno modificiraju vrijednosti gena sadržanih u hromozomu. Način na koji se genotip jedinke prevodi u njene spoljašnje osobine određen je šemom kodiranja. Šema kodiranja predstavlja obostrano jednoznačno preslikavanje $C : \boldsymbol{x} \leftrightarrow \boldsymbol{a}$, gdje je $\boldsymbol{x} = [x_1,x_2,...,x_n]^T$ tačka iz prostora stanja $E^n$, a $\boldsymbol{a}=<a_1a_2...a_l>$ hromozom koji se sastoji od niza elemenata alfabeta $\boldsymbol{A}$, kojim je predstavljena tačka $\boldsymbol{x}$. Dekodirani hromozom jedinke daje njene spoljašnje osobine, odnosno fenotip. Tako npr. za gore pomenuti slučaj binarne reprezentacije hromozoma, gen se sastoji od niza bita koji kodiraju elemenat $x_i$ vektora $\boldsymbol{x}$ (slika 4.4). Dekodiranje binarno kodiranih hromozoma u vektor $\boldsymbol{x}$ se najčešće vrši na osnovu izraza [5]:

$$\begin{aligned} I_i &= \sum_{j=1}^{l(i)} a_{ij} \cdot 2^{j-1} \\ x_i &= \underline{x_i} + \frac{\overline{x_i} - \underline{x_i}}{2^{l(i)} - 1} \cdot I_i \end{aligned} \tag{4.1}$$

gdje su $\overline{x_i}$ i $\underline{x_i}$ gornja i donja granica vrijednosti elementa $x_i$ , $I_i$ binarna vrijednost dijela stringa $\boldsymbol{a}$ kojim je predstavljen element $x_i$ , a $l(i)$ dužina pomenutog dijela stringa $\boldsymbol{a}$. Broj bita kojima će biti predstavljen element $x_i$ se najčešće određuje na osnovu izbora koraka $\varepsilon$ kojim se uzorkuje prostor stanja [5]:

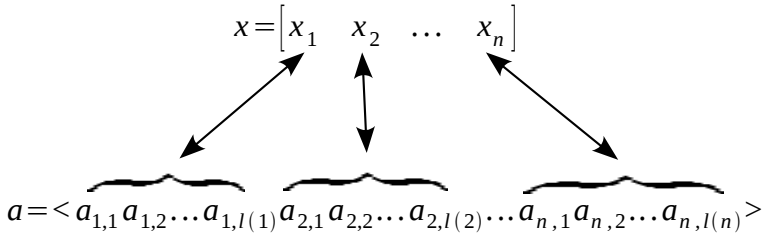
$$l(i) = \left\lceil \log_2 \frac{(\overline{x_i} - \underline{x_i})}{\varepsilon} \right\rceil \tag{4.2}$$

Slično, kodiranje vrijednosti $x_i$ u binarni string se vrši prema izrazu:

$$I_i = \left\lfloor (2^{l(i)} - 1) \cdot \frac{x_i - \underline{x_i}}{\overline{x_i} - \underline{x_i}} \right\rfloor \tag{4.1}$$

Kao primjer, posmatrajmo dio hromozoma, koji kodira opseg vrijednosti realnih brojeva od $\underline{x_i} = 1$ do $\overline{x_i} = 2$ , uz korak $\varepsilon = 0,1$. Potreban broj bita za kodiranje ovih vrijednosti će biti:

$$l(i) = \left\lceil \log_2 \frac{(\overline{x_i} - \underline{x_i})}{\varepsilon} \right\rceil = \left\lceil \log_2 \frac{(2-1)}{0,1} \right\rceil = \left\lceil \log_2 10 \right\rceil = 4$$




Slika 4.4. Predstavljanje vektora $\boldsymbol{x}$ binarnim hromozomom

Tako, ako je potrebno kodirati vrijednost 1,7 u binarni string, cijeli broj na osnovu kojega ćemo odrediti dio hromozoma će biti:

$$I_i = \left\lfloor (2^{l(i)} - 1) \cdot \frac{x_i - \underline{x_i}}{\overline{x_i} - \underline{x_i}} \right\rfloor = \left\lfloor (2^4 - 1) \cdot \frac{1{,}7 - 1}{2 - 1} \right\rfloor = \left\lfloor 15 \cdot \frac{0{,}7}{1} \right\rfloor = \lfloor 15 \cdot 0{,}7 \rfloor = \lfloor 10{,}5 \rfloor = 10$$

Vidimo da će traženi dio hromozoma biti 1010.

Isto tako, ako je binarni string koji predstavlja dio hromozoma 0111, dekodirani realni broj će biti:

$$x_i = \underline{x_i} + \frac{\overline{x_i} - \underline{x_i}}{2^{l(i)} - 1} \cdot I_i = 1 + \frac{2 - 1}{2^4 - 1} \cdot 7 = 1 + \frac{7}{15} = 1 + 0{,}4667 = 1{,}4667 \approx 1{,}5$$

Istorijski gledano, binarno kodiranje je korišteno u GA od njegove pojave, dok su se kasnije javile i verzije GA sa kodiranjem korištenjem alfabeta veće kardinalnosti, uključujući i kodiranje realnim brojevima [6], kao i verzije GA sa kompleksnijim šemama kodiranja i strukturiranim hromozomima [7]. Binarno kodiranje korišteno u jednostavnom GA (*Simple Genetic Algorithm* - SGA) je još uvijek osnovna šema kodiranja za pokušaje teorijske analize genetičkog algoritma.

### *4.3.3. Fitness*

Selekcija hromozoma populacije *k*-te populacije $\mathbf{P}^k = \{\boldsymbol{a}_1^k, \boldsymbol{a}_2^k, \dots, \boldsymbol{a}_m^k\}$, gdje je $m$ broj hromozoma u populaciji, na koje će biti primjenjeni operatori GA i koji će učestvovati u stvaranju hromozoma *k+1*-ve populacije $\mathbf{P}^{k+1} = \{\boldsymbol{a}_1^{k+1}, \boldsymbol{a}_2^{k+1}, \dots, \boldsymbol{a}_m^{k+1}\}$ se odvija na bazi njihovog kvaliteta. Kvalitet jedinke predstavljene hromozomom $\boldsymbol{a}_j^k$ se ocjenjuje na osnovu njenog fitnessa (*fitness*), koji u stvari predstavlja pogodno transformiranu kriterijalnu funkciju polaznog problema traženja optimuma. Jedinke čiji hromozomi imaju bolji fitness imaju statistički gledano veću šansu da učestvuju u kreiranju populacije naredne generacije. Relacija između fitnessa $f(\boldsymbol{a})$ i kriterija $y(\boldsymbol{x})$ može imati različite forme [3] i obično uključuje linearno skaliranje u obliku:

$$f(\boldsymbol{a}) = c_1 \cdot y(\boldsymbol{x}) + c_2 \tag{4.4}$$

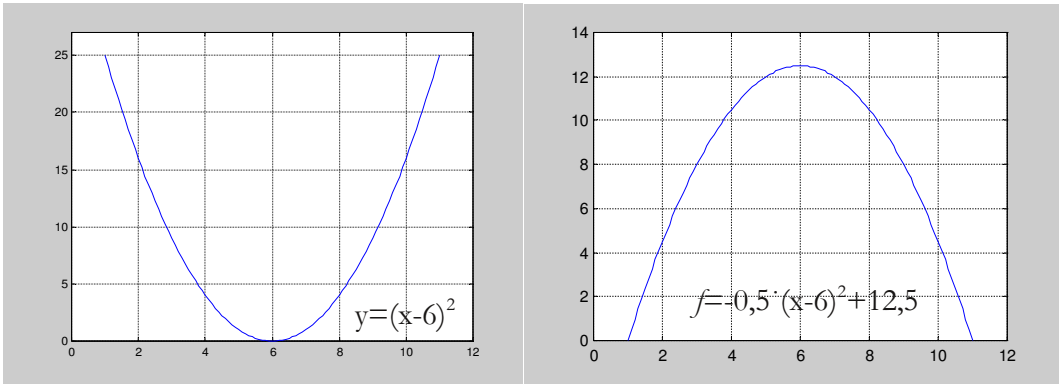
gdje $c_1$ i $c_2$ predstavljaju pogodno izabrane realne brojeve, a eventualno se javlja i dodatno, u opštem slučaju nelinearno, preslikavanje između kriterija $y(\boldsymbol{x})$ i neke mjere korisnosti $u(\ .\ )$. Jasno je da u nekim slučajevima sam kriterij $y(\boldsymbol{x})$ može biti funkcija fitnessa. Najčešći razlog zbog kojeg se vrši transformacija kriterija u fitness je dovođenje u formu pogodnu za primjenu u GA, npr. na način kako je to predstavljeno na slici 4.5. U svakom slučaju, hromozom se vrednuje u prostoru fenotipa, te ga je radi toga potrebno dekodirati korištenjem poznate i jednoznačne šeme kodiranja.

Uvođenjem pojma fitnessa i razdvajanjem genotipa i fenotipa jedinke se obezbjeđuje izvjesna apstrakcija nivoa traženja optimuma od nivoa realiteta u kome egzistira jedinka koja je objekat traženja. Upravo u ovoj apstrakciji se ogleda generalnost GA u odnosu na druge problemski specifične algoritme.

### *4.3.4. Mehanizam selekcije*

Kako je već više puta naglašeno, GA predstavlja u osnovi stohastički algoritam u kome bolje jedinke imaju veću šansu da učestvuju u kreiranju jedinki naredne populacije. Mehanizam selekcije je izrazitno bitan element svakog GA koji direktno određuje korespodenciju između fitnessa hromozoma i vjerovatnoće njegovog učešća u reprodukciji.

Postoji niz različitih mehanizama selekcije [8], od kojih većina modelira različite prirodne procese i pojave. Razlog postojanja više mehanizama selekcije leži u različitom uspostavljanju odnosa između fitnessa hromozoma i vjerovatnoće njegove selekcije. U nastavku će biti kratko predstavljeni najčešće korišteni mehanizmi selekcije.




Slika 4.5. Transformacija između kriterijalne funkcije i fitnessa

*Selekcija skraćivanjem*

Selekcija skraćivanjem (*truncation selection*) [5] predstavlja vrlo jednostavan način selekcije, koji predstavlja model vještačkog uzgoja različitih vrsta životinja i biljaka.

Provodi se tako što se populacija prvo sortira po opadajućem fitnessu, a zatim se određeni dio najboljih jediniki populacije odabire za reprodukciju. Ovaj dio određen pragom skraćivanja (*truncation threshold*) se obično kreće između 10% i 50% veličine populacije. Što je prag manji, veći udio u kreiranju potomaka imaju bolje jedinke, dok za veće vrijednosti praga i lošije jedinke dobijaju udio u kreiranju potomaka.

Iz ovog odabranog dijela populacije, sastavljenog od najboljih jedinki, se onda potpuno slučajno (sa uniformnom vjerovatnoćom) vrši odabir jedinki za reprodukciju.

Selekcija skraćivanjem je vrlo jednostavna za implementaciju, ali bezuvjetno odbacivanje jednog velikog dijela jedinki je čini manje sofisticiranom i rijetko se koristi pri rješavanju praktičnih problema primjenom GA. Izuzetak je uzgajivački GA (*BGA-Breeder Genetic Algorithm*), o kome će biti riječi nešto kasnije.

*Selekcija proporcionalno fitnessu*

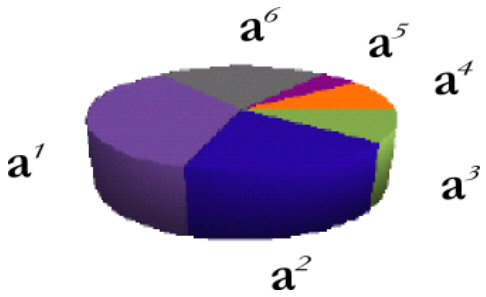
Istorijski gledano, prvi mehanizam selekcije koji je korišten još u Hollandovom GA je selekcija proporcionalno fitnessu hromozoma, poznata još pod imenom selekcija na bazi točka ruleta (*Roullete Wheel Selection* – RWS). Kod ovog načina selekcije, koji modelira prirodno odabiranje, vjerovatnoća selekcije hromozoma $\boldsymbol{a}^j \in \mathbf{P}$ je predstavljena izrazom [3], [6], [8]:

$$p_s(\boldsymbol{a}^j) = \frac{f(\boldsymbol{a}^j)}{\sum_{k=1}^{m} f(\boldsymbol{a}^k)} \tag{4.5}$$

Na taj način je obezbjeđeno da hromozomi sa natprosječnim fitnessom imaju i natprosječnu šansu da budu izabrani za reprodukciju. Realizacija ove šeme selekcije se zasniva na implementaciji ruletskog točka (odatle i naziv Roulette Wheel selekcija), tako što se svakom hromozomu dodjeljuje sektor točka proporcionalan njegovom fitnessu (slika 4.6).

Mehanizam selekcije se provodi sljedećim redom:

- za svaki hromozom $\boldsymbol{a}^j \in \mathbf{P}$ se računa $p_s(\boldsymbol{a}^j)$,
- u skladu sa $p_s(\boldsymbol{a}^j)$ se svakom hromozomu dodijeli sektor točka,
- točak se zavrti $m$ puta i svaki put kada se točak zaustavi, hromozom na čijem se sektoru točak zaustavio se uzima za reprodukciju.




Slika 4.6. Selekcija proporcionalno fitnessu

Pseudokod na osnovu kojega se može implementirati selekcija proporcionalno fitnessu je predstavljen na slici 4.7 [4]. Parametar *pop* predstavlja hromozome tekuće populacije, kako bi se mogla izračunati vjerovatnoća selekcije svakog od njih. Vraća se indeks odabranog hromozoma.

Funkcija koja implementira selekciju proporcionalno fitnessu pri svakom pozivu bira jedan hromozom, što znači da postoji izvjesna vjerovatnoća da čak i najbolji hromozom u populaciji ne bude izabran za ukrštanje. Stohastičko univerzalno uzorkovanje (*Stohastic Universal Sampling* - SUS), kao varijanta selekcije proporcionalno fitnessu, koristi $m$ pokazivača raspoređenih u skladu sa uniformnom slučajnom raspodjelom umjesto da se $m$ puta zavrti točak. Pseudokod ove varijante mehanizma selekcije je predstavljen na slici 4.8 [4]. $m_{sel}$ predstavlja broj hromozoma iz populacije koje je potrebno odabrati, a $m$ ukupan broj hromozoma u populaciji. Vraća se vektor $\boldsymbol{c}$ koji sadrži indekse odabranih hromozoma.

$$
\begin{aligned}
&RWS\,(pop)\\
&\quad j \leftarrow 1\\
&\quad suma \leftarrow p_s(\boldsymbol{a}^j)\\
&\quad u \leftarrow rand\,(0,1)\\
&\quad while\ suma < u\ do\\
&\qquad j \leftarrow j+1\\
&\qquad suma \leftarrow suma + p_s(\boldsymbol{a}^j)\\
&\quad end\\
&\quad return\ j
\end{aligned}
$$

Slika 4.7. Pseudokod selekcije proporcionalno fitnessu

*Selekcija na bazi ranga*

Da bi se izbjegla težnja mehanizma selekcije proporcionalne fitnessu da prenaglašava šansu hromozoma sa boljim fitnessom da budu izabrani za reprodukciju čime se umanjuje raznolikost populacije, koriste se drugi mehanizmi selekcije kao što je selekcija bazirana na rangu hromozoma (*ranking selection*).

```
SUS ( pop , m_sel)
    u ← rand (0, 1/m_sel)
    suma ← 0
    for j = 1 to m do
        c_j ← 0
        suma ← suma + p_s(a^j)
        repeat
            c_j ← c_j + 1
            u ← u + 1/m_sel
        while u < suma
    end
    return c
```

Slika 4.8. Pseudokod stohastičkog univerzalnog uzorkovanja

Selekcija na bazi ranga se zasniva na modificiranju originalnog fitnessa u njegovu modificiranu vrijednost, na način da se ujednači razlika u vrijednosti fitnessa između boljih i lošijih jedinki populacije. Postoji više načina na koji se može provesti selekcija na bazi ranga i oni se uglavnom razlikuju po tome kako se izračunava modificirana vrijednost fitnessa. Ovi izrazi za izračunavanje mogu biti:

- linearni,
- nelinearni.

Ukoliko se modificirani fitness određuje na bazi linearnog izraza, tada će i vjerovatnoće selekcije opadati linearno od jedinke sa najboljim fitnessom do jedinke sa najlošijim fitnessom.

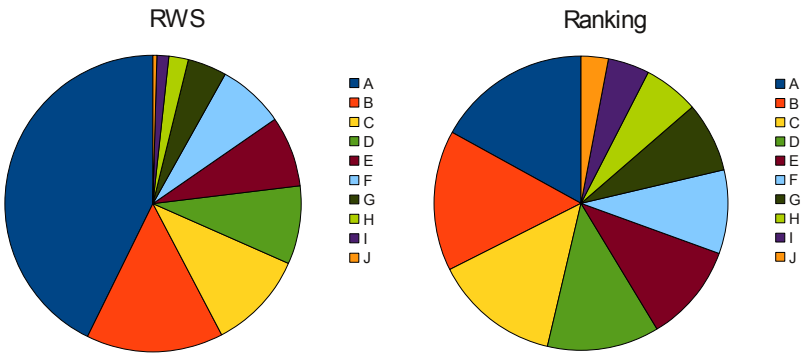
Modificirani fitness hromozoma $\boldsymbol{a}^j \in \mathbf{P}$ pri linearnoj selekciji na bazi ranga se obično određuje na osnovu izraza [8]:

$$f_{mod}(\boldsymbol{a}^j) = (2 - SP) + 2 \cdot (SP - 1) \cdot \frac{rank(\boldsymbol{a}^j) - 1}{m - 1} \tag{4.6}$$

gdje je $rank(\boldsymbol{a}^j)$ redoslijedni broj hromozoma $\boldsymbol{a}^j$ u populaciji $\mathbf{P}$ poredanoj po rastućim vrijednostima fitness-a, a $SP$ parametar koji se naziva selekcijski pritisak i usvaja se iz opsega (1, 2]. Mehanizam selekcije se onda provodi na isti način kao što je bio slučaj kod selekcije proporcionalne fitnessu, samo što se umjesto stvarnog koristi modificirani fitness. Kada selekcijski pritisak ima maksimalnu vrijednost, onda je razlika između vjerovatnoće selekcije boljih i lošijih jedinki najveća. Kada bi selekcijski pritisak imao vrijednost 1, onda bi sve jedinke imale istu vjerovatnoću selekcije, bez obzira na njihov fitness. Drugim riječima, u tom slučaju bi GA vršio potpuno slučajno pretraživanje problemskog prostora.

Obzirom da se selekcija na bazi ranga od selekcije proporcionalno fitnessu razlikuje samo u određivanju modificiranog fitnessa, način implementacije može biti identičan onome predstavljenom na slici 4.7.

Na slici 4.9 se mogu vidjeti različite ovisnosti vjerovatnoće selekcije o fitnessu jedinki, kada se koriste selekcija proporcionalno fitnessu i selekcija na bazi ranga.




a) b)

Slika 4.9. Razlika u dodjeljivanju vjerovatnoće selekcije za: a) selekciju proporcionalno fitnessu i b) selekciju na bazi ranga

*Turnirska selekcija*

Još jedan način da se izbjegnu nedostaci selekcije proporcionalno fitnessu, kao i dodatna kompleksnost selekcije bazirane na rangu zbog potrebe za sortiranjem svake populacije $\mathbf{P}^k$, predstavlja turnirska selekcija (*tournament selection*) [8]. Kod ove selekcije se po dvije jedinke iz populacije $\mathbf{P}^k$ potpuno slučajno biraju za turnir. Zatim se sa određenom vjerovatnoćom $p_t$, koja predstavlja parametar selekcije bira bolja od dvije jedinke. Turnir je potrebno ponoviti $m$ puta. Moguće je da turnir uključuje i više od dvije jedinke.

## *4.3.5. Ukrštanje*

Operator ukrštanja (*crossover*) predstavlja binarni operator:

$$(\boldsymbol{a}^p, \boldsymbol{a}^q) \xrightarrow[\Phi]{} (\boldsymbol{a}^r, \boldsymbol{a}^s) \tag{4.7}$$

koji primjenjen na hromozome-roditelje $(\boldsymbol{a}^p, \boldsymbol{a}^q)$, izabrane mehanizmom selekcije, daje hromozome-potomke $(\boldsymbol{a}^r\ \boldsymbol{a}^s)$. Realizacija operatora $\varphi$ ovisi o vrsti GA i šemi kodiranja, a moguće je i proširenje operatora ukrštanja na $n$-arni operator, kada u kreiranju hromozoma potomaka učestvuje više od dva hromozoma roditelja. Od svih evolucionih algoritama ovaj operator se izvorno javlja samo u GA i dio GA praktičara i teoretičara smatra da je operator ukrštanja primarni operator genetičkog algoritma.

Njegova osnovna funkcija je da, kombinujući informaciju sadržanu u populaciji (odnosno u izabranim jedinkama populacije), kreira nove jedinke-potomke. Obzirom da su među hromozomima na koje se primjenjuje ovaj operator uglavnom bolje jedinke sa stanovišta fitnessa, postoji vjerovatnoća da će se među hromozomima potomcima pojaviti oni koji sadrže dijelove hromozoma roditelja koji su suštinski za performansu jedinke, te da će na taj način postići bolji fitness. Ovo je osnova hipoteze gradivnih blokova (*building block hypothesis*), kao jednog od fundamentalnih principa funkcionisanja GA. Obzirom da binarno kodiranje predstavlja osnovnu varijantu kodiranja korištenu u GA, u nastavku će biti predstavljeni osnovni operatori ukrštanja binarnih stringova.

Najjednostavniju formu operatora ukrštanja predstavlja ukrštanje u jednoj tački (*single point crossover*). Ovaj operator (slika 4.9), koji je John Holland koristio u izvornom GA, potomke kreira prema sljedećem izrazu:

$$\begin{aligned}\boldsymbol{a}^r &= \langle a_1^p\, a_2^p \ldots a_k^p\, a_{k+1}^q a_{k+2}^q \ldots a_l^q \rangle \\ \boldsymbol{a}^s &= \langle a_1^q\, a_2^q \ldots a_k^q\, a_{k+1}^p a_{k+2}^p \ldots a_l^p \rangle\end{aligned} \tag{4.8}$$

gdje su:

$$\begin{aligned}\boldsymbol{a}^p &= \langle a_1^p\, a_2^p \ldots a_l^p \rangle \\ \boldsymbol{a}^q &= \langle a_1^q a_2^q \ldots a_l^q \rangle\end{aligned} \tag{4.9}$$

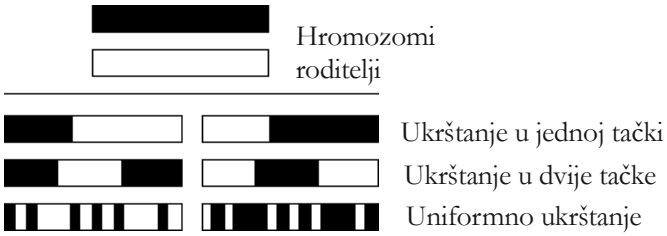
hromozomi roditelji, a vrijednost $k \in \{1,2,\ldots,l\text{-}1\}$ slučajno izabrano mjesto ukrštanja. Negativna osobina ukrštanja u jednoj tački je da uvijek razdvaja krajeve hromozoma-roditelja, što u nekim primjenama GA nije poželjno. Naime, ako se povoljne vrijednosti lokusa hromozoma roditelja nalaze na krajevima, ukrštanje u jednoj tački će ih svaki put rastavljati i time umanjivati vrijednost fitnessa.

Radi ovoga je uveden operator ukrštanja u dvije tačke (slika 4.10):

$$\begin{aligned}\boldsymbol{a}^r &= \langle a_1^p\, a_2^p \ldots a_k^p\, a_{k+1}^q a_{k+2}^q \ldots a_h^q\, a_{h+1}^p\, a_{h+2}^p \ldots a_l^p \rangle \\ \boldsymbol{a}^s &= \langle a_1^q\, a_2^q \ldots a_k^q\, a_{k+1}^p a_{k+2}^p \ldots a_h^p\, a_{h+1}^q a_{h+2}^q \ldots a_l^q \rangle\end{aligned} \tag{4.10}$$

gdje su $k \in \{1,2,\ldots,l\text{-}2\}$ i $h \in \{1,2,\ldots,l\text{-}1\}$, $k<h$ slučajno izabrana mjesta ukrštanja. Povećavajući dalje broj tačaka ukrštanja dolazimo do operatora ukrštanja u $n$ tačaka, odnosno za $n=l$ do uniformnog ukrštanja. Kod operatora uniformnog ukrštanja se za svaki lokus hromozoma slučajno bira vrijednost lokusa jednog od hromozoma-roditelja [5]:

$$\begin{aligned}a_i^r &= a_i^p \cdot z_i + a_i^q \cdot (1 - z_i) \\ a_i^s &= a_i^p \cdot (1 - z_i) + a_i^q \cdot z_i \\ z_i &\in (0,1) \\ i &\in (1,2,\ldots,l)\end{aligned} \tag{4.11}$$




Slika 4.10. Ukrštanje u jednoj tački, u dvije tačke i uniformno ukrštanje

Treba istaći da sve navedene varijante operatora ukrštanja kombinuju samo već postojeće vrijednosti bita hromozoma populacije $\mathbf{P}$ , i ukoliko ni jedan hromozom populacije ne posjeduje vrijednost bita $a_i$ [1], ona se ne može pojaviti ni u jednoj generaciji GA kada se primjenjuje samo operator ukrštanja. To znači da ni jedna dekodirana jedinka neće posjedovati performansu koju je kodirala vrijednost $a_i$ .

Bitno je reći da se u GA kao parametar javlja vjerovatnoća ukrštanja $p_c$ , što znači da će određeni dio izabranih roditelja na generaciji $k$ u generaciju $k+1$ preći neizmijenjen.

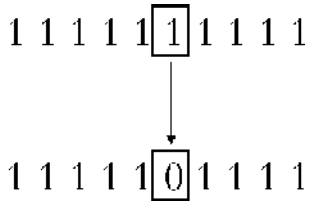
### *4.3.6. Mutacija*

Operator mutacije (*mutation*) je unarni operator :

$$\boldsymbol{a}^p \xrightarrow[\mu]{} \boldsymbol{a}^r \tag{4.12}$$

koji slučajno modificira hromozom na koji je primjenjen. Za GA sa binarnim kodiranjem osnovni operator mutacije predstavlja negacija vrijednosti bita na lokusu $i$ sa vjerovatnoćom $p_m$ koja ima malu vrijednost (slika 4.11.) [5]:

$$\begin{aligned} &a_i^r = a_i^p \cdot z_i + \overline{a_i^p} \cdot (1 - z_i) \\ &z_i \in \{0,1\} \quad \text{vrijednost izabrana u skladu sa } p_m \\ &i \in \{1,2,\ldots,l\} \end{aligned} \tag{4.13}$$




Slika 4.11. Binarna mutacija

1 Do ove pojave dolazi kada početna populacija nije dobro generirana, kada je broj hromozoma izrazito mali u odnosu na prostor koji se pretražuje, te kada djelovanje operatora ukrštanja u prethodnim generacijama potpuno eliminira određenu vrijednost bita iz populacije.

Osim ovako definiranog operatora mutacije, mogu se sresti i druge verzije, kao što su mutacija pomakom, inverzijom dijela hromozoma i slično. Obično se smatra da operator mutacije, iako je neophodan, u GA predstavlja sekundarni operator, mada mu se u posljednje vrijeme pridaje sve više značaja. Njegova osnovna funkcija je da osigura raznolikost populacije GA, odnosno da permanentno potpuno slučajno modificirajući hromozome neovisno o njihovom fitnessu sprečava gubitak vrijednosti bita $a_i$, što bi u protivnom dovelo do preuranjene konvergencije GA ne samo prema lokalnom optimumu, nego čak i prema bilo kojoj tački prostora koji se pretražuje. Druga funkcija mutacije je da kreira eventualno vrijednost $a_i$ koju ne posjeduje ni jedan hromozom. Obzirom da će u populaciji od $m$ hromozoma dužine $l$ svaki od bita biti mutiran sa vjerovnoćom $m \cdot p_m$, za $N$ generacija GA svaki bit će biti mutiran sa vjerovatnoćom $N \cdot m \cdot p_m$ . Da bi se omogućilo da se tokom izvršavanja GA svaki bit što izvjesnije bar jednom mutira, i time uvede eventualno nepostojeća vrijednost $a_i$ u populaciju, preporučeno je da broj generacija GA bude izabran u skladu sa:

$$N \geq \frac{1}{m \cdot p_m} \tag{4.14}$$

Kako je u praktičnoj primjeni GA obično $m \geq 10$, a $p_m \geq 0{,}001$, slijedi da je za mutiranje svakog bita hromozoma barem jednom tokom izvođenja GA potrebno da je $N \geq 100$. Iz iste nejednakosti se može zaključiti da se sa smanjenjem broja hromozoma u populaciji treba povećati vjerovatnoća mutacije, kako bi se umanjio neželjen uticaj gubitka određenih vrijednosti bita u populaciji, pri čemu treba imati u vidu da pomenuta nejednakost samo ukazuje na vezu između veličine populacije, vjerovatnoće mutacije i broja generacija, a ne da vrijedi egzaktno.

I na kraju, treća funkcija mutacije je slučajno pretraživanje prostora stanja.

### *4.3.7. Mehanizam smjene*

Mehanizam smjene vrši zamjenu jedinki populacije stare generacije i formiranje nove populacije, što omogućava gradnju različitih verzija GA, od generacijskog GA do stabilnog GA (Steady-State GA) kao krajnjih varijanti.

Kod generacijskog GA se cjelokupna populacija roditelja mijenja populacijom potomaka. S druge strane, kod stabilnog GA se u populaciju uvodi samo mali broj novih jedinki dok se većina roditelja zadržava.

GA koji koristi samo selekciju, ukrštanje i mutaciju neprestano otkriva i (zbog stohastičkih elemenata sadržanih u operatorima ukrštanja i mutacije) uništava hromozome sa visokom vrijednošću fitnessa. Zbog toga će populacije takvog GA neprekidno oscilirati u okolini tačke optimuma i vrlo je mala

vjerovatnoća da će u posljednjoj populaciji GA biti i hromozom čiji je fitness za praktičnu primjenu dovoljno blizu optimalnom.

Radi toga se u GA pri rješavanju problema optimizacije uvodi elitizam. Pod elitizmom[2] se podrazumijeva zadržavanje na svakoj generaciji jednog ili više hromozoma sa najboljim fitnessom i njihovo bezuslovno prenošenje u narednu generaciju. Time se bitno povećava performansa GA, što je od naročitog značaja za njegovu primjenu u rješavanju problema optimizacije.

### *4.3.8. GA sa realnim kodiranjem*

U dosadašnjim razmatranjima je korišteno kodiranje potencijalnih rješenja u binarne stringove. Iako je ovaj način kodiranja korišten u izvornom GA i vrlo jednostavan, performansa GA sa binarnim kodiranjem nije najbolja u svakoj primjeni. To je razlog korištenja drugih načina kodiranja, od kojih je najrašireniji u primjeni kodiranje potencijalnog rješenja u vektor realnih brojeva. Michalewicz, Bäck, Davis i drugi [4] su pokazali da za većinu realnih primjena performansa GA sa realnim kodiranjem bitno nadmašuje performansu jednostavno GA. Kod GA sa realnim kodiranjem je $\boldsymbol{A} = \boldsymbol{R}$, odnosno kao vrijednost lokusa se može pojaviti bilo koji realni broj.

Najjednostavniji pristup je da se za preslikavanje $C : \boldsymbol{x} \leftrightarrow \boldsymbol{a}$ koristi identitet, odnosno da se direktno kodiraju komponente vektora $\boldsymbol{x}$. U takvom slučaju je $\boldsymbol{x} = \boldsymbol{a}$.

Korištenje kodiranja realnim brojevima zahtijeva i definiranje adekvatnih operatora ukrštanja i mutacije, te se u praksi koristi čitav niz ovih operatora. U nastavku će biti predstavljeni najčešće korišteni.

#### *4.3.8.1 Operatori ukrštanja korišteni u GA sa realnim kodiranjem*

Operatori ukrštanja hromozoma roditelja predstavljeni binarnim stringovima se mogu koristiti i u slučaju predstavljanja hromozoma vektorima realnih brojeva. Međutim, obzirom da je dužina hromozoma u slučaju realnog kodiranja dosta manja nego u slučaju binarnog kodiranja, efikasnost opisanih operatora ukrštanja u pogledu sposobnosti kreiranja novih hromozoma potomaka je manja. Radi toga će biti predstavljeni operatori ukrštanja koji se češće koriste u GA sa realnim kodiranjem.

*Diskretna rekombinacija*

Ovaj operator ukrštanja se može koristiti neovisno o šemi kodiranja (binarni string, cijeli brojevi, realni brojevi, simboli), kada je hromozom predstavljen vektorom. Svaki od elemenata vektora hromozoma

2 Elitizam je u GA 1975. godine uveo Kenneth De Jong.