Genetski algoritam

Pregled

- Uvod (motivacija, koraci)
- Vrste genetskih algoritama
- Prikazi rješenja
- Dobrota
- Selekcija
- Križanje
- Mutacija
- Parametri
- Ostalo

Motivacija

- Prirodna evolucija
- Bolje jedinke preživljavaju i prenose svoje gene na potomke
- Slučajne promjene uzrokuju potencijalna poboljšanja u jedinkama
- Nakon puno generacija, dobivamo jedinke koje su dobro prilagođene svojoj okolini

Genetski algoritam

- Baziran na ideji prirodne evolucije
- Populacijski algoritam -> radi na skupu od više rješenja
- Kombinirati dobra rješenja radi dobivanja još boljih rješenja
- Uvoditi slučajne promjene u nadi da izbjegnemo lokalne optimume
- Ponavljaj određeni broj puta

Koraci GA

- Stvoriti početna rješenja (**jedinke**)
- Na neki način odabrati 2 rješenja (**selekcija**) i pomoću njih stvoriti dijete (**križanje**)
- Provesti nasumične promjene nad djetetom (**mutacija**)
- Evaluirati dijete (**funkcija dobrote**)
- Ubaciti dijete u populaciju umjesto neke druge jedinke
- Ponavljati do kriterija zaustavljanja

Primjer

- imamo n poslova koje moramo rasporediti na m dostupnih strojeva
- Svaki posao i ima ova svojstva:
  - Trajanje izvođenja na stroju j p_i
  - Vrijeme željenog završetka d_i
  - Težina (važnost) posla w_i
- Kriterij, minimizirati težinsko kašnjenje svih poslova

$$Twt = \sum_{i}^{n} w_i * \max(C_i - d_i, 0)$$

- C_i predstavlja vrijeme završetka posla i

Vrste genetskog algoritma

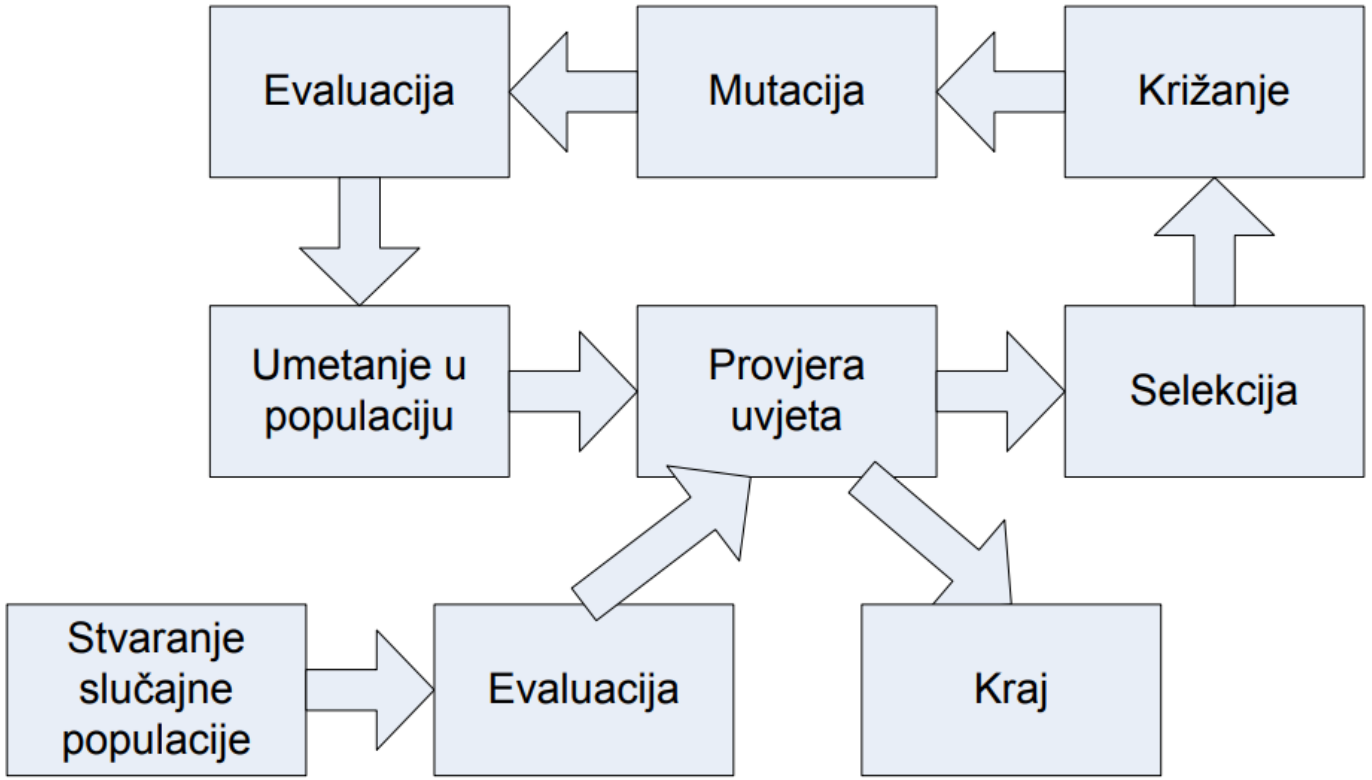
- Eliminacijski
- Generacijski
- Druge varijante

Eliminacijski GA

- Stvara se samo jedna nova jedinka

- Na neki način odaberi dva roditelja (selekcija)
- Križaj oba roditelja i stvori novu jedinku *dijete*
- Mutiraj dijete s određenom vjerojatnošću
- Ubaci dijete u populaciju (izbacivanjem neke druge jedinke)
- Ponavljaj određeni broj iteracija

Eliminacijski GA



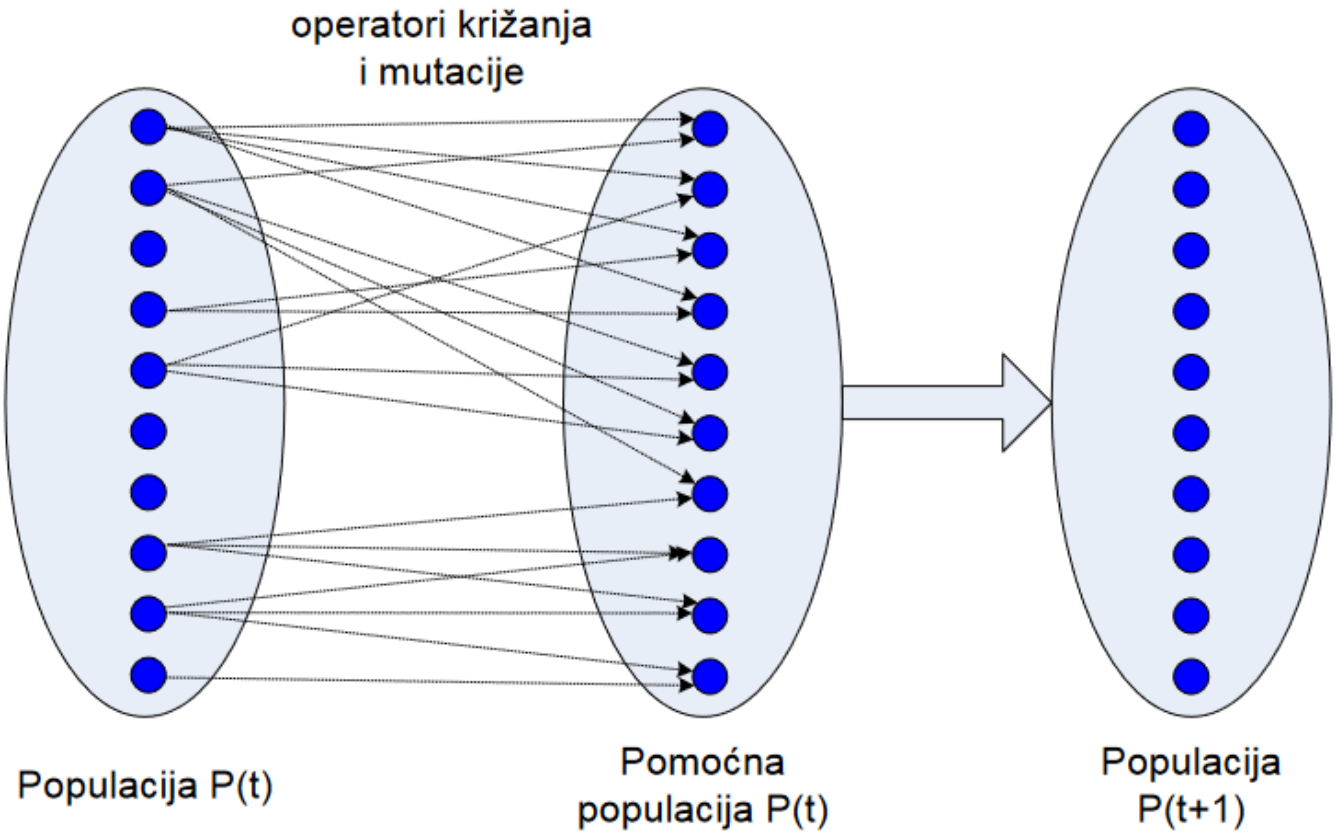
Eliminacijski GA

- Možemo definirati postotak eliminacije koji određuje koliko ćemo jedinki eliminirati iz populacije
- Odabir jedinki koje eliminiramo radimo s nekom od selekcija
- Nad preostalim jedinkama provodimo križanje i mutaciju i djecu umećemo u populaciju

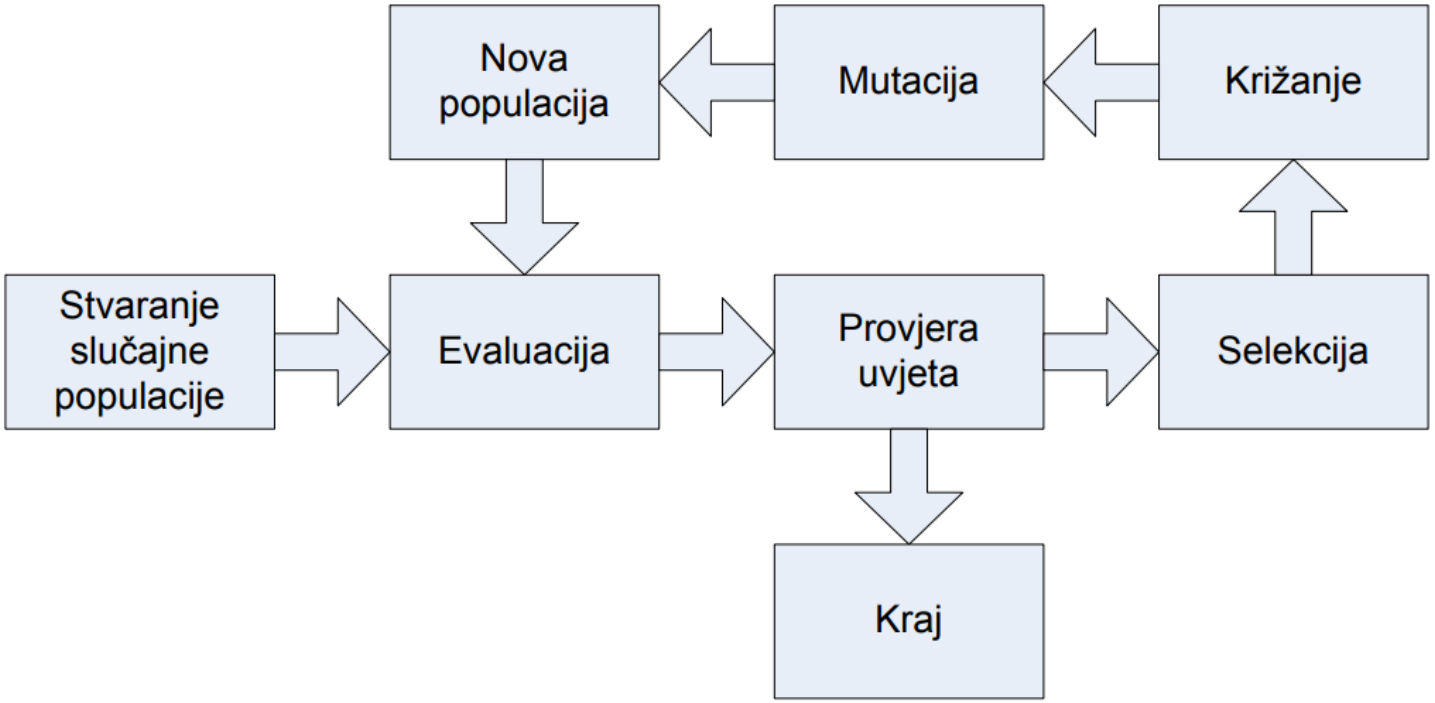
Generacijski GA

- Stvori novu populaciju jedinki (novu generaciju)
- Gradimo novu populaciju preko selekcije, križanja i mutacije
- Novu populaciju možemo graditi na razne načine:
  - Križaj dvije odabrane jedinke, mutiraj dijete i ubaci u populaciju
  - ubaci kopiju jedinke u novu generaciju
  - Mutiraj jedinku i ubaci ju u novu populaciju
- Kada izgradimo novu populaciju, brišemo staru i na njeno mjesto stavljamo novu

Generacijski GA



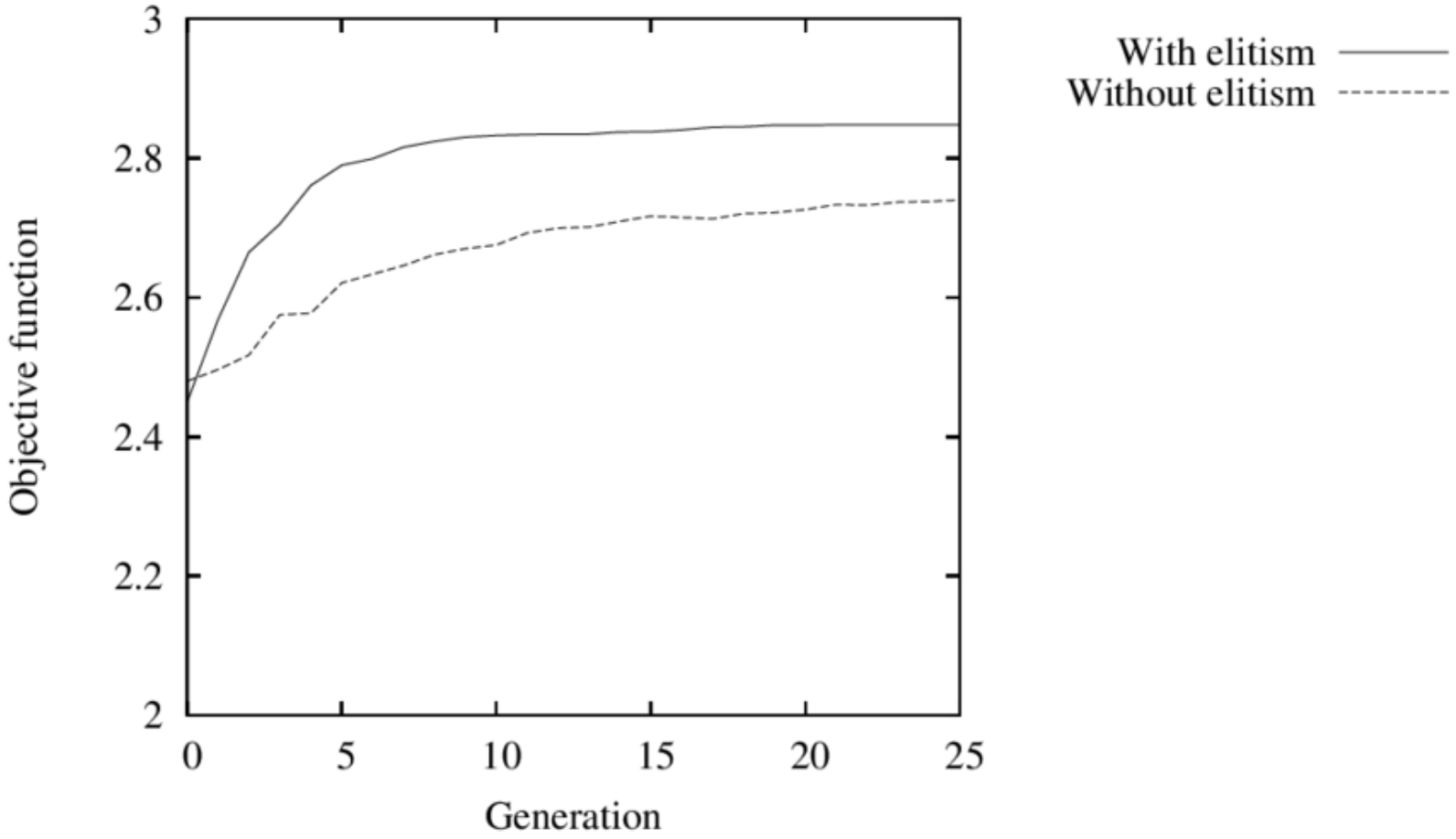
Generacijski GA



Elitizam

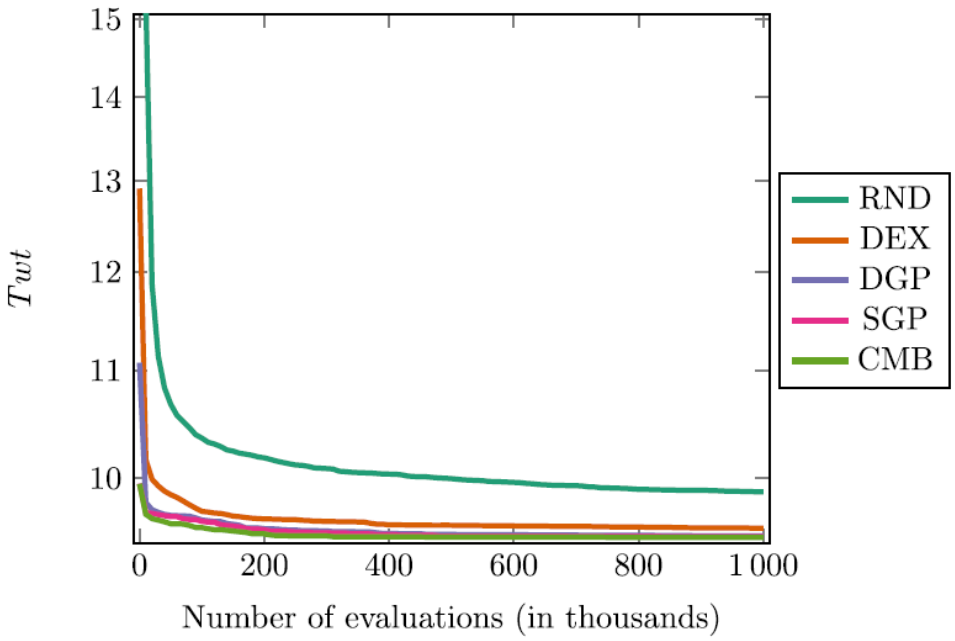
- Svojstvo da najbolja jedinka nikad ne bude eliminirane iz populacije
- Poželjno svojstvo
- Ako nema elitizma dobrota tijekom vremena oscilira
- U eliminacijskim varijantama inherentno uključen
- U generacijskim ga najčešće moramo sami ugraditi
- Može se definirati koliko najboljih jedinki želi sačuvati u populaciji

Elitizam


3
2.8
2.6
2.4
2.2
2
Objective function
0
5
10
15
20
25
Generation
With elitism
Without elitism

Stvaranje početne populacije

- Nasumično
  - Jednostavno, ali sporija konvergencija
- Heuristički
  - Koristimo neke heuristike za stvaranje rješenja
  - Moramo paziti da ne zapnemo u lokalnom optimumu
  - Kako osigurati raznolikost?



Prikazi rješenja

- Ovisan o konkretnom problemu:
  - Binarni prikaz
  - Realni
  - Permutacijski
  - Cjelobrojni
  - Miješani

Binarni prikaz

- Jedan od prvih prikaza
- Geni su 0 ili 1
- Jednostavan prikaz
- Danas se rijetko koristi
- Problem decepcije

1	1	0	1	1	1	0	0	1

Binarni prikaz

- Primjer decepcije
- n=4
- Minimum je u x=8
- Šta ako je većina rješenja >8?
- Algoritam teži prema 7 (binarno 0111)
- Najbolje rješenje je 8 (binarno 1000)
- Potrebno je promijeniti sva 4 bita kako bi došli do boljeg rješenja
- Rješenje: Greyev kod

Preslikavanje u bin. prikaz

- Bilo koji realni broj možemo prebaciti u binarni prikaz s određenom preciznošću
- Ako imamo raspon $[x_{min}, x_{max}]$ i željenu preciznost $\tilde{x}$, broj bitova za prikaz tih brojeva potrebno je $n \geq \frac{\log(\frac{x_{max}-x_{min}}{\tilde{x}}+1)}{\log(2)}$
- Realni broj možemo onda prebaciti u binarni prikaz kao

$$b = \frac{x - x_{min}}{x_{max} - x_{min}} * (2^n - 1)$$

- Binarni broj se pretvara u realni preko

$$x = x_{min} + \frac{b}{2^n} * (x_{max} - x_{min})$$

Binarni prikaz

- Problem diskretizacije prostora -> optimum možda ni ne možemo predstaviti u zadanom prikazu
- S brojem varijabli i većom preciznošću povećava se broj bitova
- Već za jednostavne problem algoritam može zapeti u suboptimalnim rješenjima
- Dobra stvar -> rješenja su uvijek u zadanim intervalima

Realni prikaz

- Najčešće korišten za kontinuirane probleme
- Općenito potrebno definirati gornju i donju granicu za varijable $x_{min}\ i\ x_{max}$
- Nakon križanja i mutacije potrebno je paziti da rješenje nije izašlo iz dozvoljenog intervala

2.14	**-15.2**	**17.3**	**3.33**	**-27.6**	**17.5**	**19.2**	**32.2**	**-34.7**

Permutacijski prikaz

- Niz jedinstvenih brojeva od 1 do n
- Koristan za probleme u kojima se mora odrediti niz određenih stvari
- Npr. niz izvršavanja poslova ili obilazak čvorova u grafu

3	2	5	1	8	9	7	6	6

Cjelobrojni prikaz

- Rjeđe korišten
- Potrebno definirati gornju i donju granicu za varijable x_{min} i x_{max}
- Za razliku od permutacijskog vrijednosti se smiju ponavljati i ne moraju sve cjelobrojne vrijednosti biti prisutne u rješenju

2	5	7	3	7	7	9	3	4

Ostali prikazi

- Matrični
- Stablo (Genetsko programiranje)
- Permutacijski s delimiterima

Miješani prikaz

- Za složene probleme jedan prikaz možda nije dovoljan za predstaviti čitavo rješenje
- Npr. za predstavljanje rješenja potrebna je kombinacija permutacijskog i cjelobrojnog prikaza
- Može se koristiti i prikaz koji ne kodira cjelovito rješenje
- Primjerice, permutacijom se određuje samo redoslijed elemenata, a njihova dodjela u neke pretince se određuje u evaluaciji rješenja nekom heuristikom

Prikazi rješenja

- Fenotip – rješenje nekog problema
- Genotip – kako je rješenje prikazano u jedinci
- Jedan fenotip može se prikazati različitim genotipovima
- Za neke probleme fenotip i genotip mogu biti identični
- Često je teško razlučiti što je fenotip a što genotip

Funkcija dobrote

- Numerička vrijednost koja predstavlja kvalitetu rješenja
- Maksimizira se
- Funkcija kazne – analogno dobroti, ali ona se minimizira
- Kao funkciju dobrote/kazne često možemo koristiti vrijednost kriterija kojeg optimiramo
- Prilagodbe:
  - Relativna funkcija dobrote: $g_i = f_i - f_{min}$
  - *Windowing*: $F_i = a + (b - a) * \frac{f_i - f_w}{f_b - f_w}$, dobrotu preslikavamo u interval $[a, b]$

Selekcija

- Odabir boljih jedinki za reprodukciju i lošijih za eliminaciju
- Zašto ne bi uvijek križali najbolje i odbacivali najgore?
- Selekcijski pritisak
  - Može se definirati na razne načine
  - Općenito, omjer vjerojatnosti preživljavanja boljih i lošijih jedinki
  - Veliki selekcijski pritisak -> prebrza konvergencija i zapinjanje u lokalnim optimumima
  - Niski selekcijski pritisak -> spora konvergencija, pretraga je više nasumična

Proporcionalna selekcija

- *Roulette wheel selection*
- Vjerojatnost odabira jedinke proporcionalna je s njenom dobrotom
- Vjerojatnost odabira jedinke i $p_i = \frac{f_i}{\sum_{i=1}^{n} f_j}$
- Prednost: vjerojatnost odabira ovisi o dobroti jedinke
- Relativna proporcionalna selekcija – skaliramo sve dobrote

$$p_i = \frac{f_i - f_{min}}{\sum_{i=1}^{n}(f_j - f_{min})}$$

- Uvođenje selekcijskog pritiska:

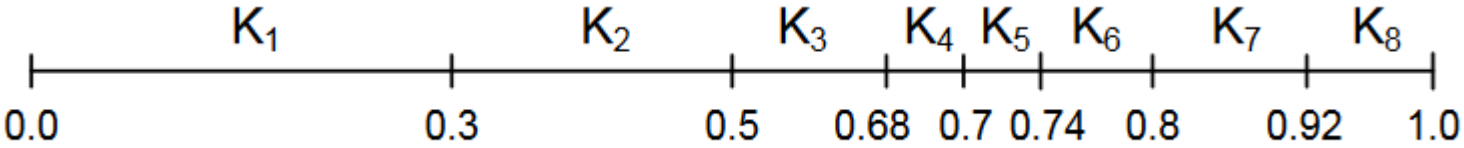
$$F_i = 1 + (SP - 1) * \frac{f_i - f_{min}}{f_{max} - f_{min}}$$

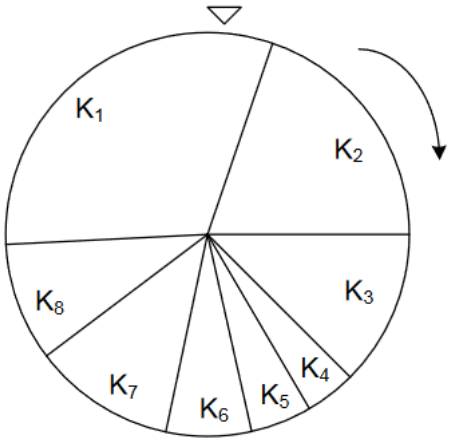
Proporcionalna selekcija

- Nedostaci:
  - Problem skale – vjerojatnosti ovise o veličinama dobrote
  - Prevladavanje najbolje jedinke
  - Dobrota mora biti pozitivna
  - Velike populacije uzrokuju često slične vjerojatnosti
  - Kod promjena u populaciji moramo ponovo računati iznose vjerojatnosti

Proporcionalna selekcija

Jedinka	1	2	3	4	5	6	7	8
Dobrota jedinke	6	4	3.6	0.4	0.8	1.2	2.4	1.6
Vjerojatnost odabira	0.3	0.2	0.18	0.02	0.04	0.06	0.12	0.08

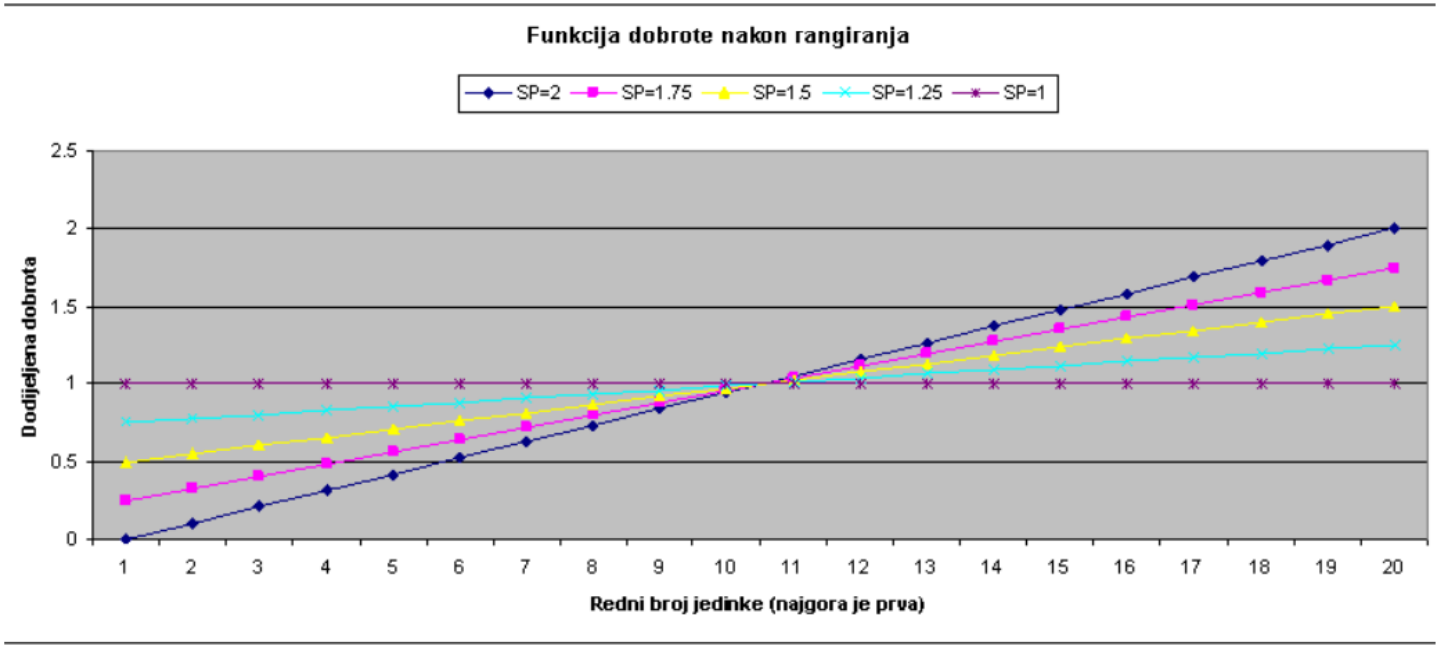




Selekcija linearnim rangiranjem

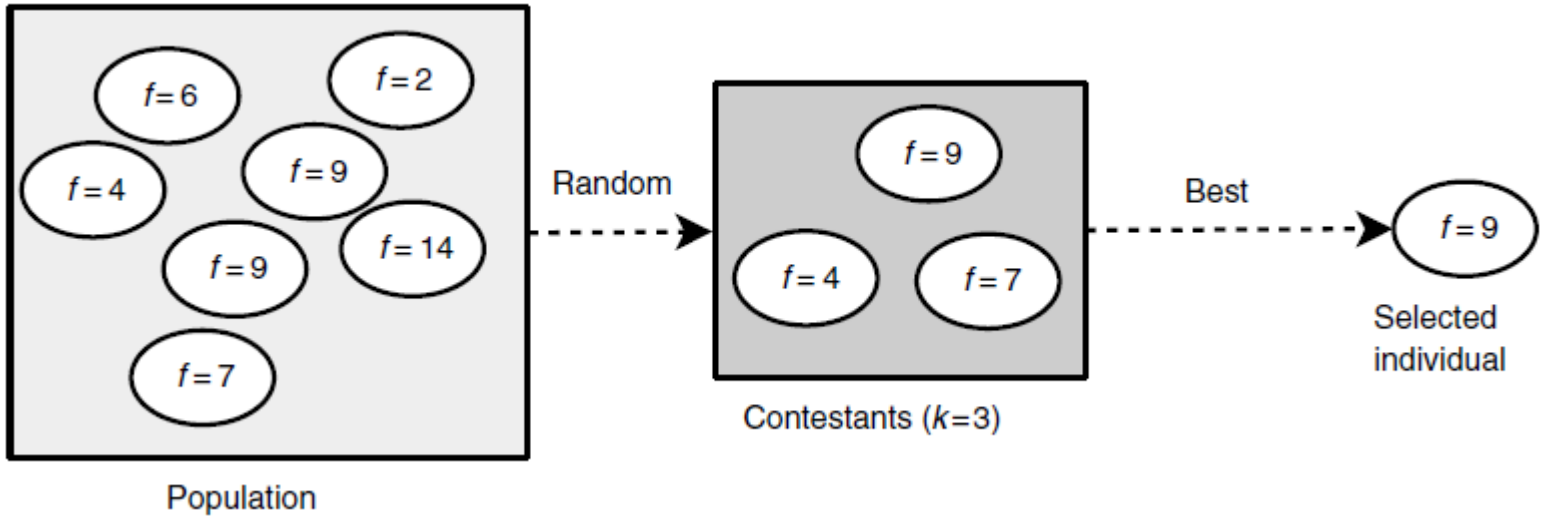
- Sortiramo sve jedinke po njihovoj dobroti
- Vjerojatnost je proporcionalna rangu
  - Rang 1 najlošije rješenje
  - Rang *n* najbolje rješenje
- $p_i = \frac{2-SP+2(SP-1)*\frac{i-1}{n-1}}{n}, SP \in [1,2]$
- SP definira selekcijski pritisak

Selekcija linearnim rangiranjem



Turnirska selekcija

- *k-tournament selection*
- Odabiremo nasumično *k* jedinki
- Od odabranih k jedinki odabiremo najbolju (ili najgoru)



Turnirska selekcija

- Parametar *k* određuje selekcijski pritisak u turnirskoj selekciji
- Mali parametar k -> SP je nizak, veća vjerojatnost odabira lošijih jedinki za preživaljanje
- Veliki parametar k -> SP je visok, manja vjerojatnost da lošije jedinke budu odabrane
- Šta bi se dogodilo kada bi parametar *k* bio jednak veličini populacije?

Jednostavna 3-turnirska selekcija

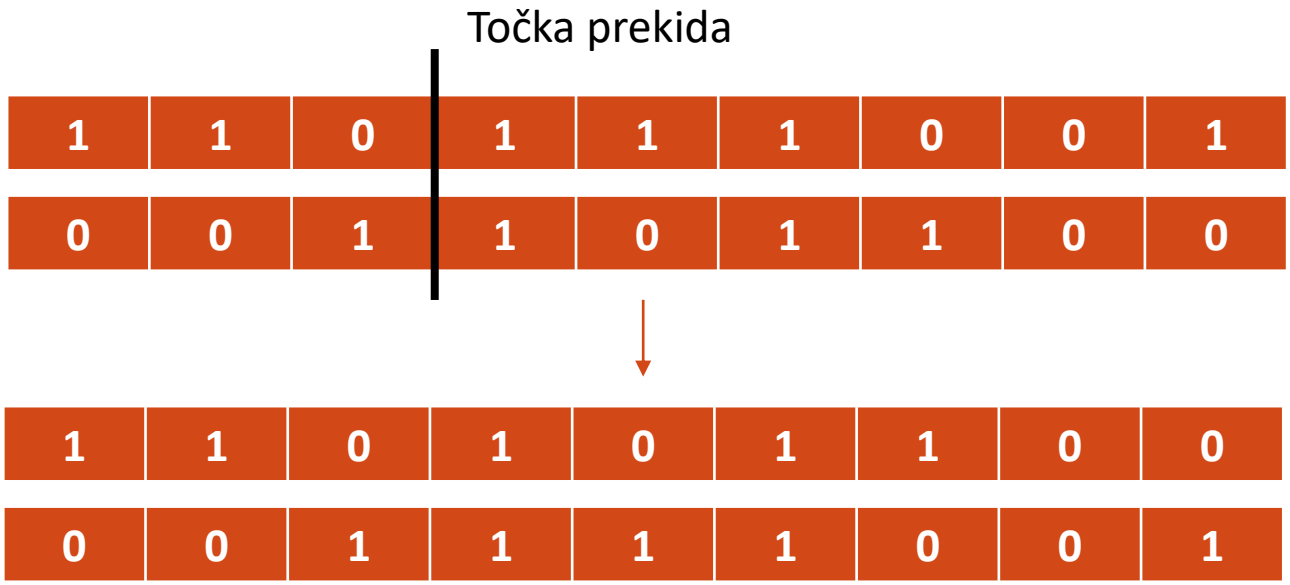
- Kod turnirske selekcije moramo više puta provoditi selekciju za odabir jedinki
- Ideja: provesti jednu selekciju za odabir roditelja i djeteta
- Odabrati 3 jedinke nasumično
- Dvije najbolje odabrati za roditelja
- Najlošija jedinka se eliminira i zamijeni novonastalom jedinkom

Križanje

- Kombiniranjem svojstva dviju ili više jedinki dobiti potencijalno bolje jedinke
- Svojstva od roditelja bi se trebala dobro preslikati u dijete
- Eksploatacija
- Ovise o prikazu

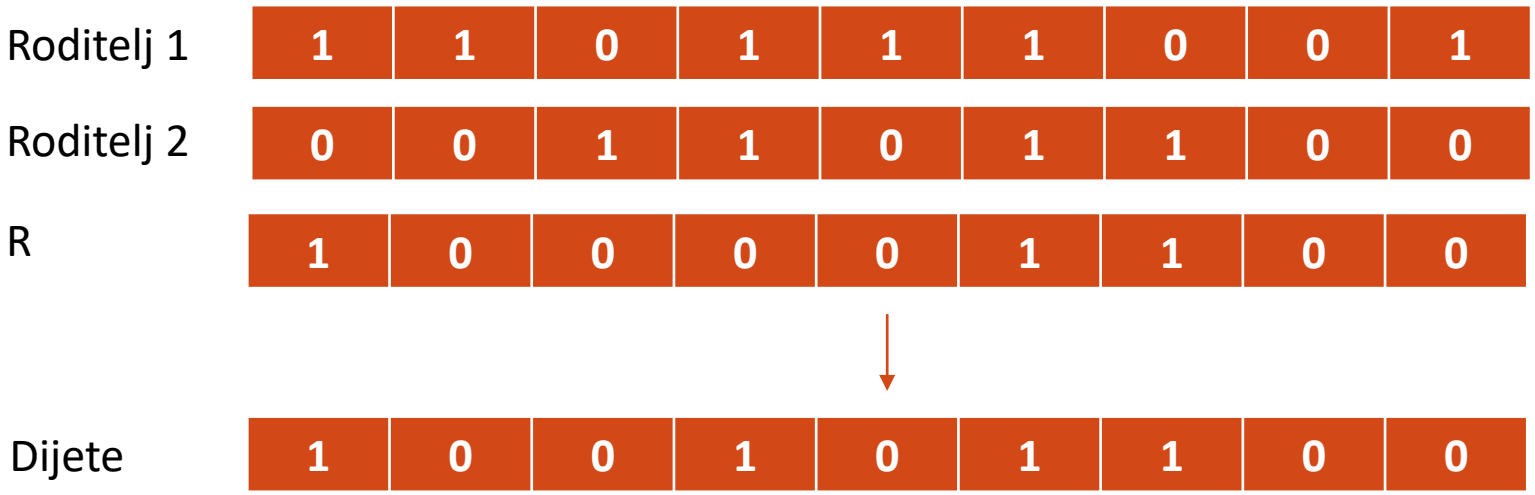
Križanje – binarni prikaz

- Križanje jednom ili više točki prekida
- Odabiremo nasumično jednu poziciju u jedinkama



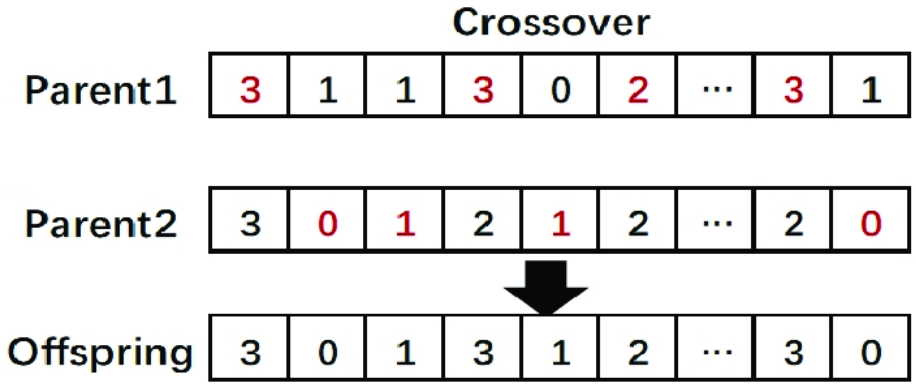
Uniformno križanje

- Generiramo nasumični vektor R
- Ako je bit u oba roditelja jednak, prepisuje se u dijete, ako je različit na temelju R se odlučuje koja vrijednost se uzima.



Križanje – realni prikaz

- Diskretno križanje – svaki gen nasumično odabiremo između oba roditelja



- Jednostavno križanje – isto kao i križanje s jednom točkom kod binarnog prikaza

Križanje – realni prikaz

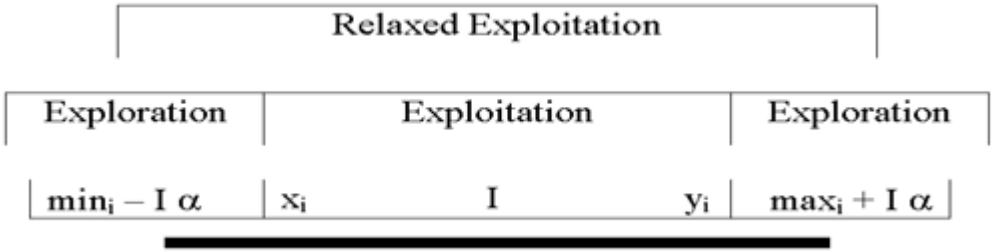
- Aritmetičko križanje

$$h_i^1 = \lambda c_i^1 + (1-\lambda) c_i^2$$
$$h_i^2 = \lambda c_i^2 + (1-\lambda) c_i^1$$

- Što se dobije za $\lambda = 0.5$?
- BLX-α

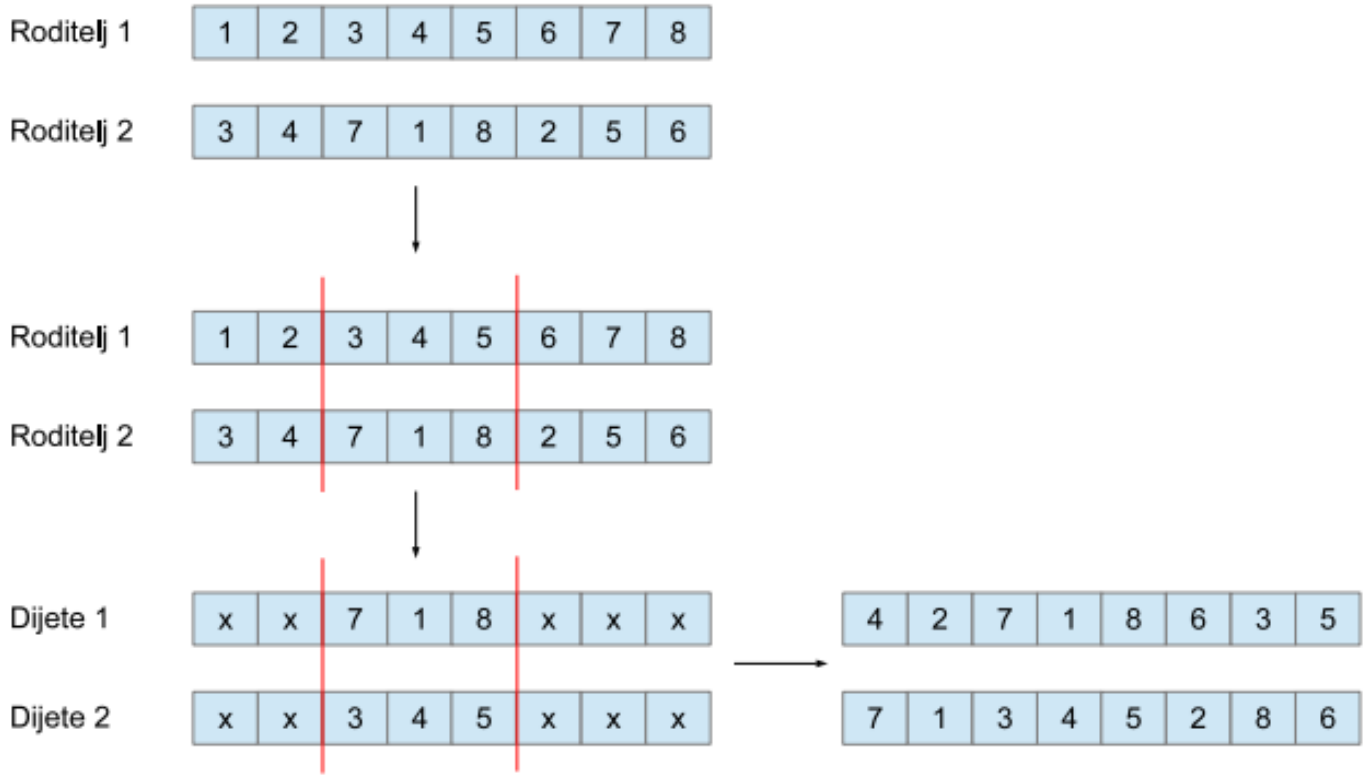
$$c_{i,min} = \min\left(c_i^1, c_i^2\right), c_{i,max} = \max\left(c_i^1, c_i^2\right)$$

$$I_i = c_{i,max} - c_{i,min}$$
$$\mathrm{h_i} \in [c_{i,min} - I_i * \alpha, c_{i,max} + I_i * \alpha]$$



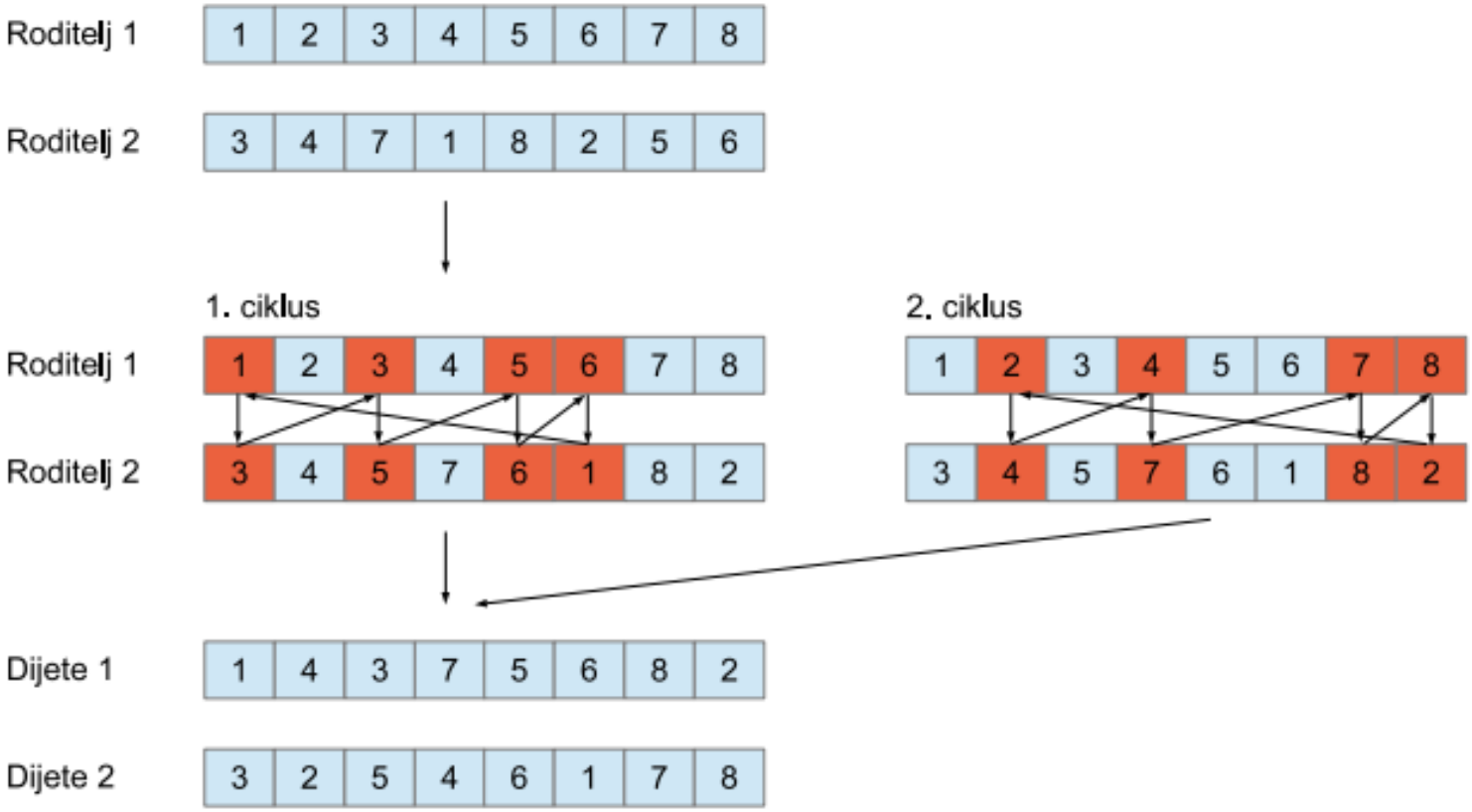
Križanje – permutacijski prikaz

- Djelomično preslikano križanje (PMX)



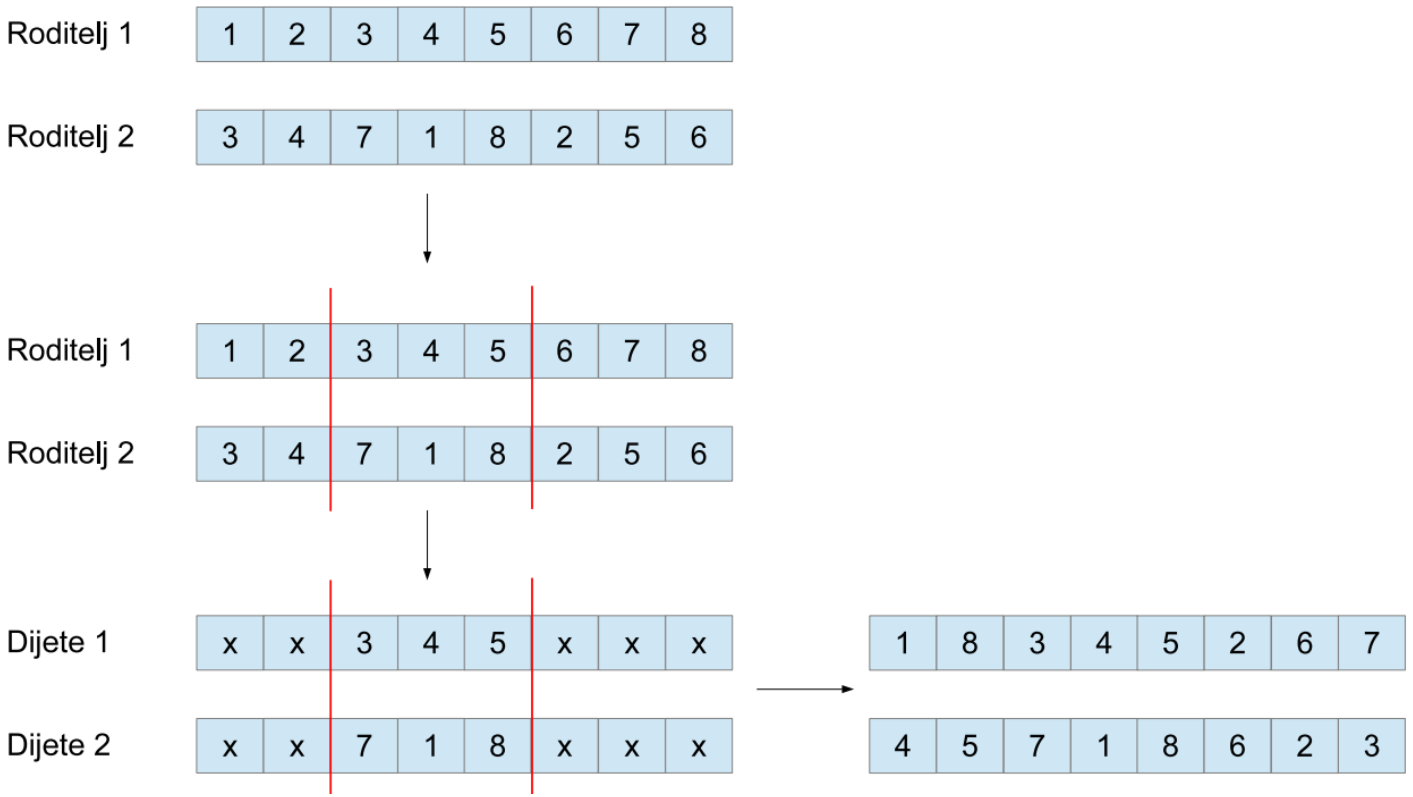
Križanje – permutacijski prikaz

- Križanje ciklusa



Križanje – permutacijski prikaz

- Križanje poretka



Mutacija

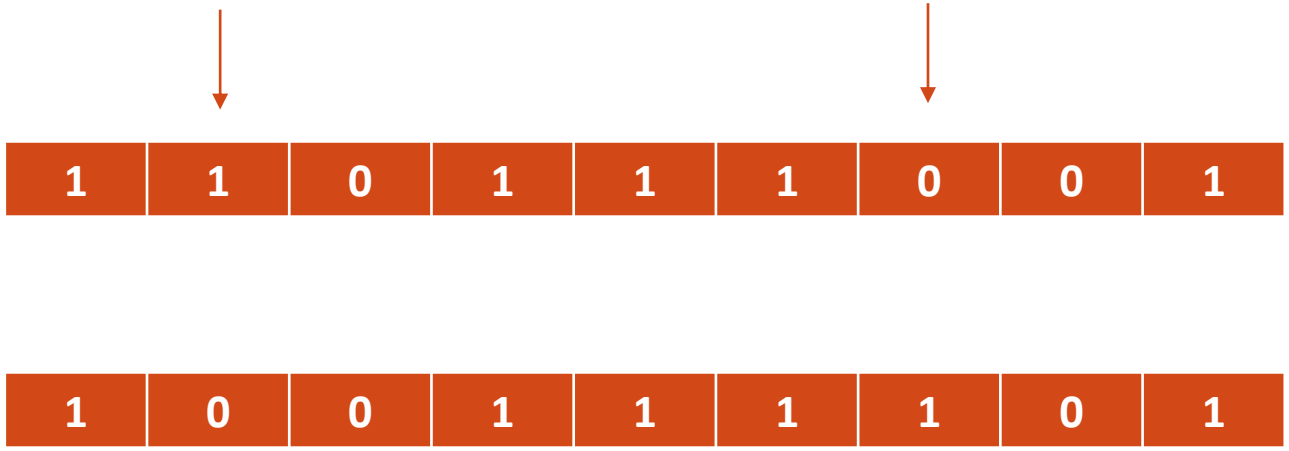
- Uvesti nasumične promjene u rješenja
- Ciljevi:
  - Izbjeći lokalne optimume
  - Vratiti izgubljene gene u populaciji
  - Dobiti možda bolja rješenja
- Ne provodimo ju uvijek, već s određenom vjerojatnošću mutacije
  - Mala vjerojatnost mutacije -> lako zapnemo u lokalnim optimumima
  - Velika vjerojatnost mutacije -> nasumična pretraga

Željena svojstva mutacije

- Korištenjem operatora mutacije trebali bi moći dobiti bilo koje rješenje
- Rješenja bi trebala biti valjana (nije uvijek moguće za probleme s ograničenjima)
- Mutacijom bi trebali dobiti rješenja koja su blizu trenutnog rješenja

Mutacija – binarni prikaz

- Zamjena bitova
- Svaki bit mutiramo s određenom vjerojatnošću
- Čemu su jednake vjerojatnosti?
  - Svaki bit ima istu vjerojatnost
  - Bitovi imaju različite vjerojatnosti

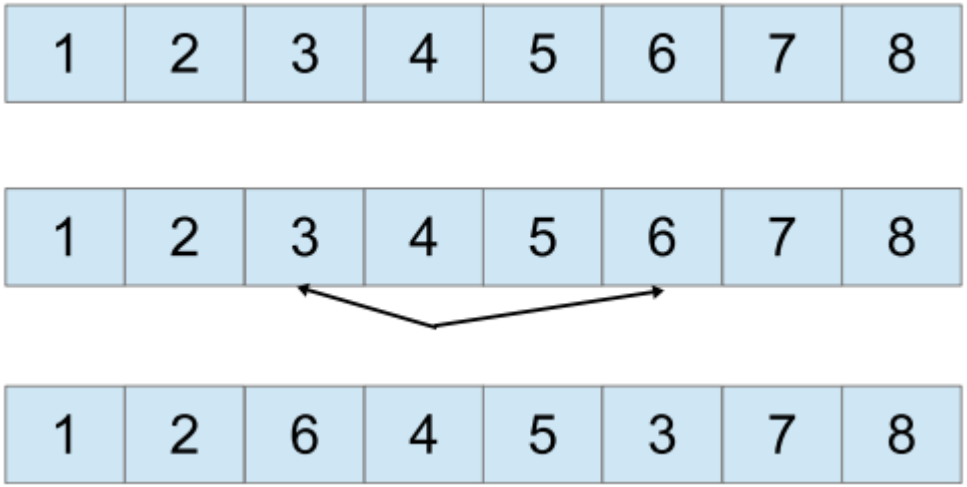


Mutacija – realni prikaz

- Uniformna mutacija – odabiremo vrijednost uniformno iz zadanog intervala
  - Zamjena cijele vrijednosti: $x = r, r \in [x_{min}, x_{max}]$
  - Dodavanje neke uniformne vrijednosti: $x = x * (b - a) + a$ generiramo rješenje iz intervala $[a, b]$
- Gaussolika mutacija – dodajemo neki Gaussov šum na jednu ili više varijabli
  - $x = x + r, r \in \mathcal{N}(0, s)$, s predstavlja standardnu devijaciju
  - Koliki je raspon rješenja?
- Kako odrediti željene parametre?

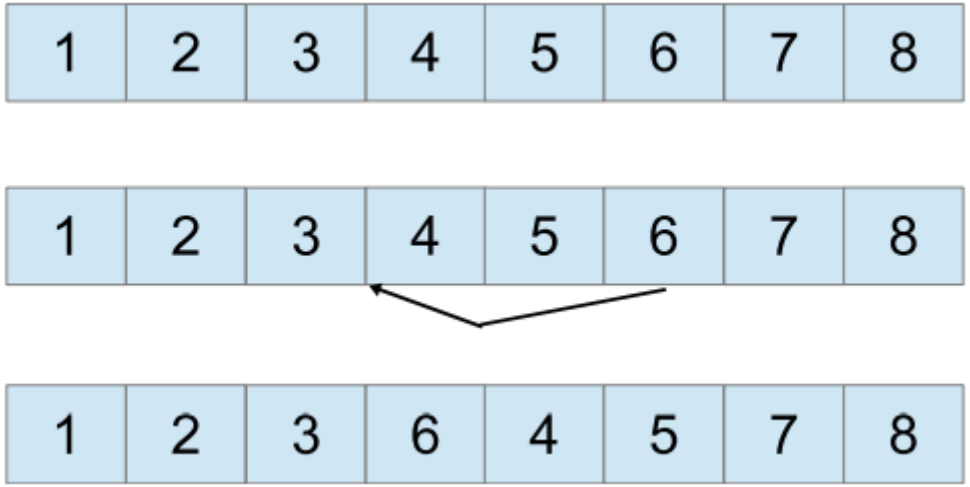
Mutacija – permutacijski prikaz

- Mutacija zamjenom



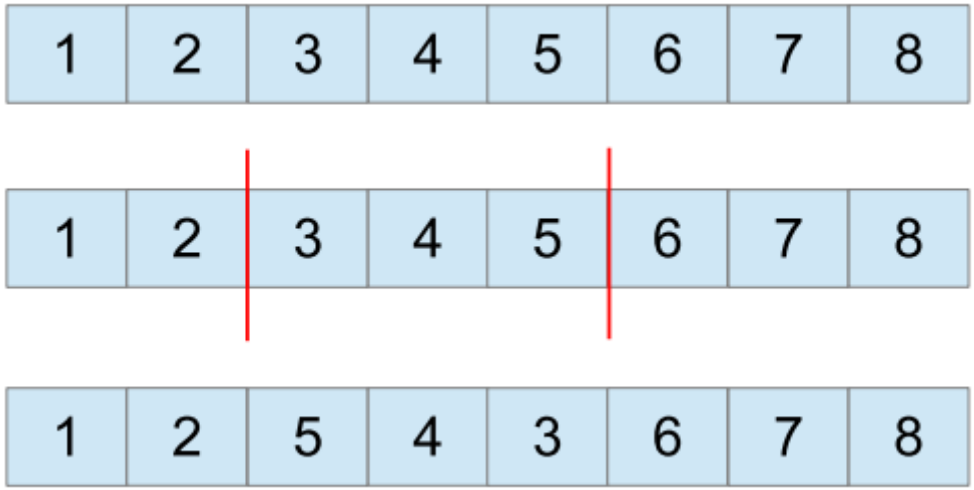
Mutacija – permutacijski prikaz

- Mutacija umetanjem



Mutacija – permutacijski prikaz

- Mutacija inverzijom



Genetski operatori

- Koje genetske operatore odabrati?
- Možemo koristiti jedan ili više operatora
- Ako koristimo više operatora:
  - Svi mogu imati jednaku vjerojatnost primjene
  - Vrijednost se može adaptivno mijenjati ovisno o uspješnosti pojedinih operatora

Kriteriji zaustavljanja

- Broj iteracija/generacija
- Broj evaluacija funkcije cilja
- Stagnacija
- Vremensko ograničenje
- Postignuto dovoljno dobro rješenje

Parametri GA

- Veličina populacije
- Vjerojatnost mutacije
- Kriterij zaustavljanja
- Parametri specifični za prikaze/križanja/mutacije i slično
- Koje vrijednosti odabrati?
- Parametri uvelike ovise jedni o drugima:
  - Vjerojatnost mutacije i veličina populacije
  - Veličina populacije i broj iteracija/evaluacija

GA i ograničenja

- Ne postoji jednostavno rješenje
- Ograničiti GA da radi samo s ispravnim rješenjima
  - Potrebno prilagoditi sve dijelove GA
  - Ne garantira dobar uspjeh, prostor može biti dosta razlomljen pa ga nije moguće raditi dobro pretražiti
- Uvesti kaznu u funkciju dobrote koja usmjerava pretragu u bolja područja
  - $F = f(x) - \lambda P(x)$
  - Kakav mora biti odnos između dodane kazne i funkcije dobrote
  - Iznos bi trebao biti adaptivan
  - Nemamo garanciju da ćemo dobiti ispravno rješenje

Memetički algoritam

- Cilj: poboljšati efikasnost genetskih algoritama
- Ugraditi operatore lokalne pretrage u GA
- U određenim fazama genetskog algoritma primijeniti postupke lokalne pretrage na neka rješenja u populaciji
- Lokalnu pretragu možemo upotrijebiti nakon
  - Mutacije nekog rješenja
  - Određenog broja iteracija nad nasumičnim/najboljim rješenjima u populaciji

Memetički algoritam

Inicijaliziraj populaciju P

Evaluiraj populaciju

Ponavljaj

- Odaberi roditelje i križaj ih
- Mutiraj dijete
- Evaluiraj dijete
- **Poboljšaj dijete primjenom lokalne pretrage**
- Ubaci dijete u populaciju

Dok kriterij zaustavljanja nije zadodvoljen

Memetički algoritam

Inicijaliziraj populaciju P

Evaluiraj populaciju

Ponavljaj

- Odaberi roditelje i križaj ih
- Mutiraj dijete
- Evaluiraj dijete
- Ubaci dijete u populaciju
- **Ako zadovoljen kriterij primjene lokalne pretrage**
  - **Provedi lokalnu pretragu nad odabranim jedinkama**

Dok kriterij zaustavljanja nije zadovoljen

Zaključak

- Jedna od najstarijih i najpopularnijih metaheuristika
- Osnovna varijanta radi dobro za veliki broj problema
- Za bolje rezultate potrebno je dobro prilagoditi algoritam problemu
- Često se hibridizira s drugim metodama: simulirano kaljenje, lokalne pretrage i slično
- Primjenjiv na širok spektar problema