Vještačka inteligencija

Predavanje 10: Genetički algoritmi

„Postaviti pravo pitanje je pola znanja.“
~Roger Bacon

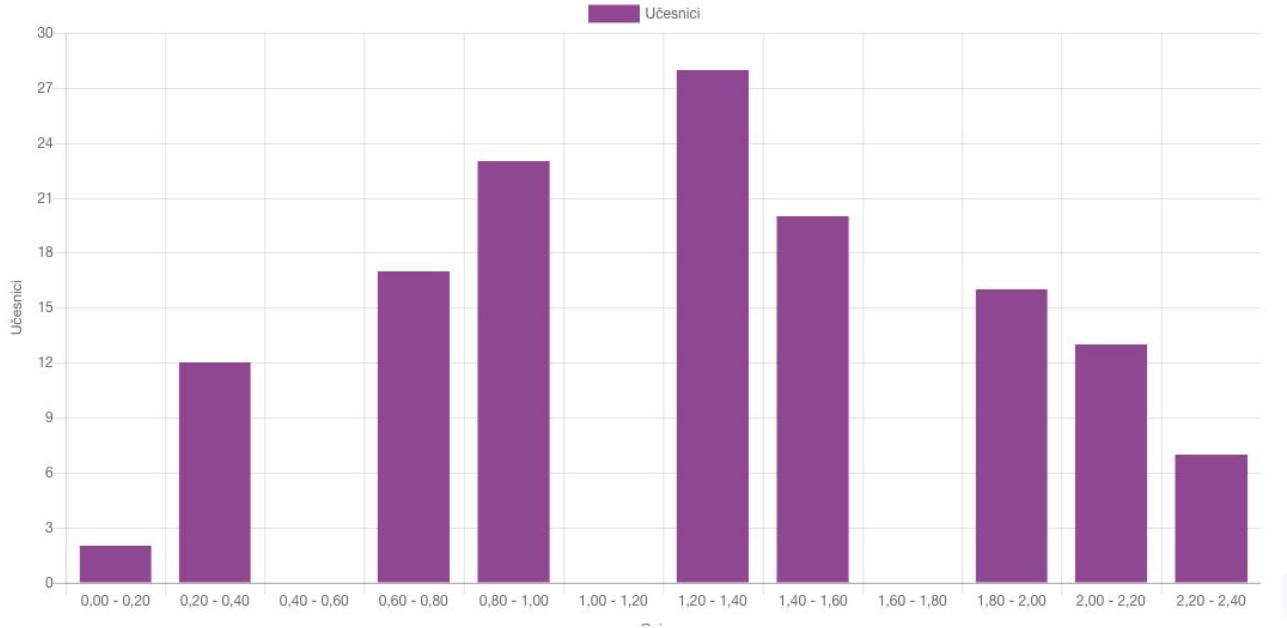
Odgovorna nastavnica: Vanr. prof. dr Amila Akagić

Univerzitet u Sarajevu

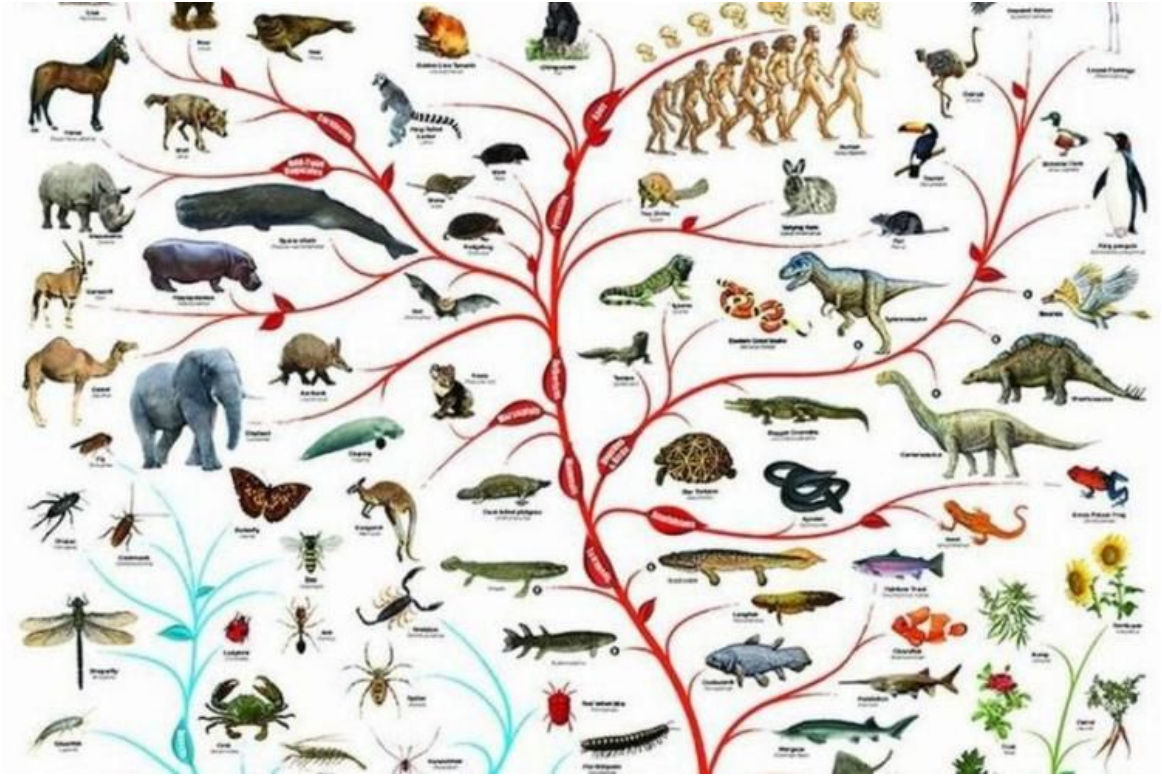
Uvodne informacije

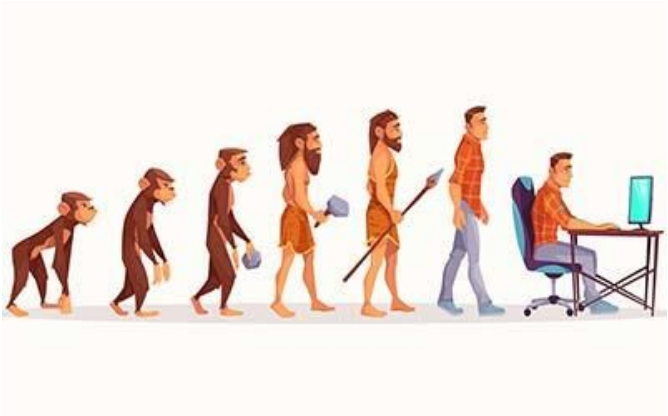
Najave

- Kviz broj 7 održat će se 20.05.2021. u 14:02 preko c2.
- Sa predavanjima nastavljamo od 14:10+.
- Ovo je XII sedmica nastave.
- U XV sedmici je **II provjera znanja**.

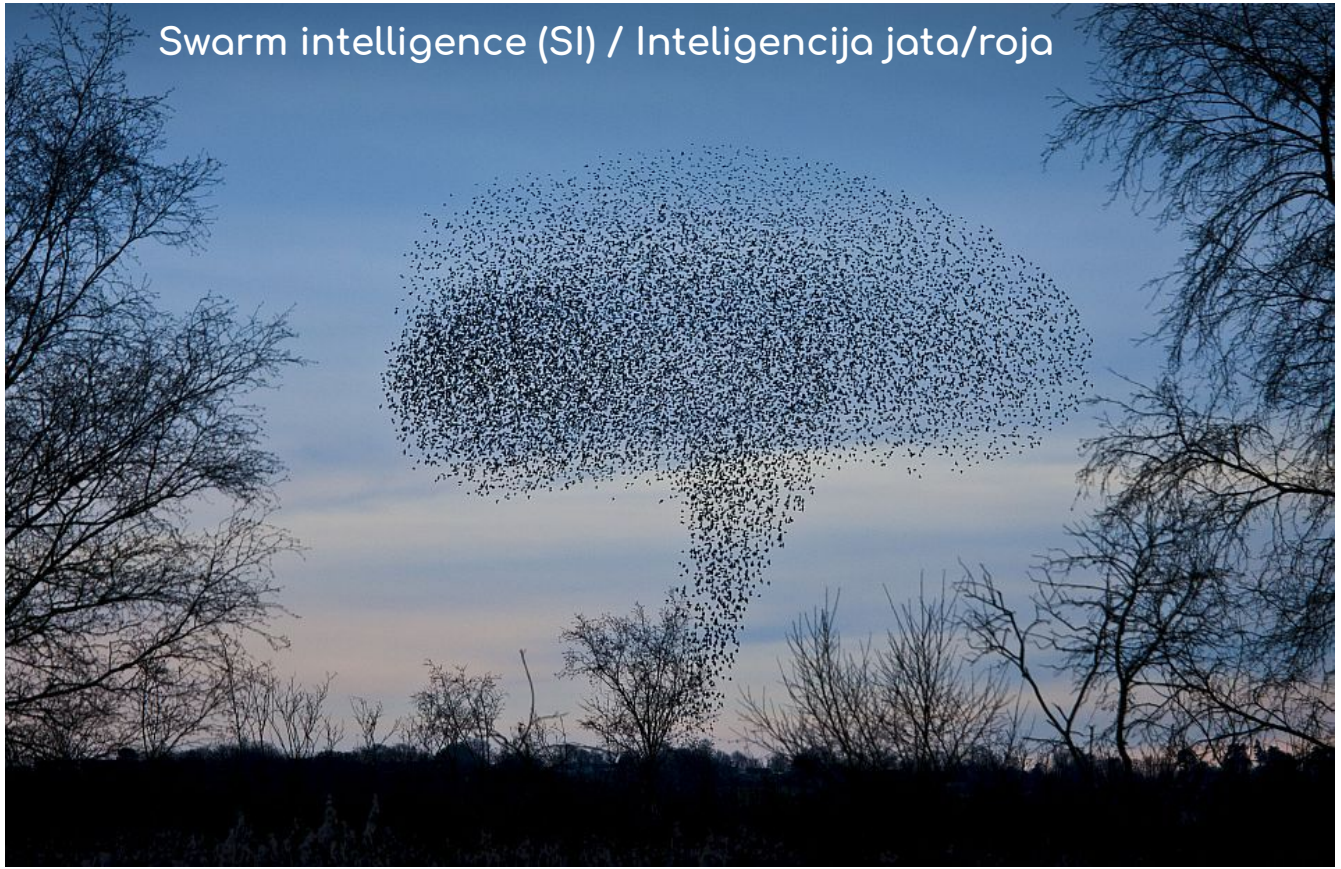


Algoritmi evolucijskog računarstva



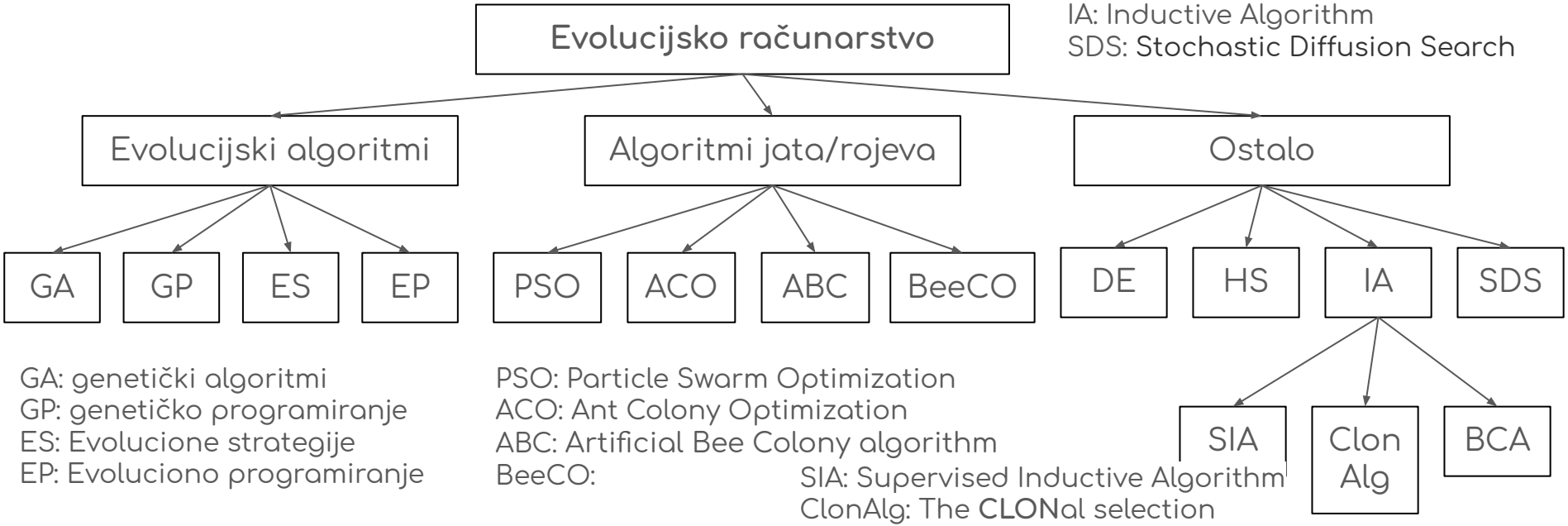


Algoritmi evolucijskog računarstva



Evolucijsko računarstvo

- **Evolucijsko računarstvo**: grana vještačke inteligencije koja se, najvećim dijelom, bavi rješavanjem optimizacijskih problema.
- Začeci: kasne 50-te godine



DE: Differential Evolution
HS: Harmony Search
IA: Inductive Algorithm
SDS: Stochastic Diffusion Search

GA: genetički algoritmi
GP: genetičko programiranje
ES: Evolucione strategije
EP: Evoluciono programiranje

PSO: Particle Swarm Optimization
ACO: Ant Colony Optimization
ABC: Artificial Bee Colony algorithm
BeeCO:

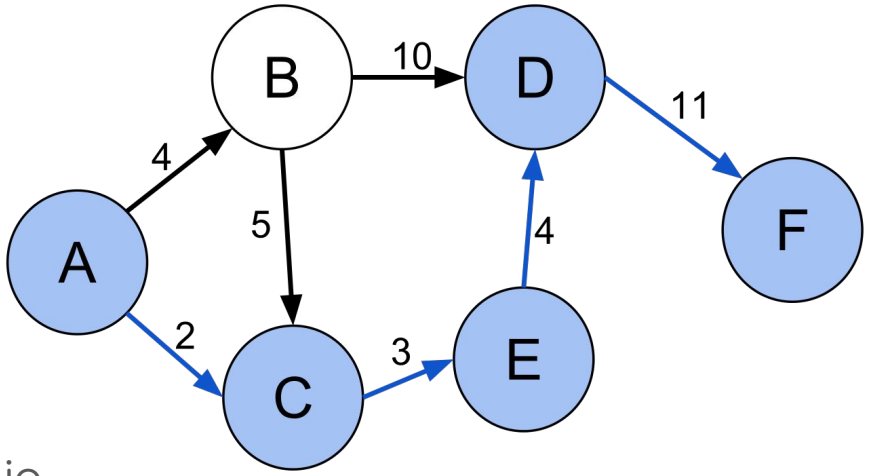
SIA: Supervised Inductive Algorithm
ClonAlg: The **CLON**al selection **ALG**orithm
BCA: The B-Cell Algorithm

Optimizacija

- Optimizacija: postupak pronalaženja najboljeg rješenja problema.
- Rješenje će pri tome imati definisanu funkciju sposobnosti (eng. *fitness-function*) koju će optimizacijski postupak maksimizirati, ili funkciju gubitka (cijene; eng. *cost-function*) koju će optimizacijski postupak minimizirati.
- Problemi koje pokušavamo optimizirati:
  - **kombinatorički** - definisani nad diskretnom domenom koja može ali čak i ne mora imati uređaj između elemenata (poredite npr. domenu cijelih brojeva i činjenicu da je 1 manje od 5 s domenom tropskog voća i činjenicom da ne možemo reći je li banana manja, veća ili jednaka ananasu);
    - Pretraživanje u širinu
    - Pretraživanje u dubinu
    - A*
  - **kontinualni** - definirani nad kontinualnim domenama poput realnih brojeva, kompleksnih brojeva i slično, i gdje već ta činjenica povlači beskonačan prostor pretraživanja koji grubom silom nikada ne možemo pretražiti.

Optimizacija

- Pretraživanje prostora stanja:
  - Nađu put od A do F.
  - Rješenje je najkraći put.
- **Problem pretraživanja prostora stanja** svodi se na problem pronalaska puta od početnog stanja s_p do ciljnog stanja s_c koristeći dozvoljene poteze.
- **Problem zadovoljavanja ograničenja** (eng. Constraint Satisfaction Problem): vrsta problema pretraživanja prostora stanja kod koje put od početnog do konačnog stanja nije bitan.
  - Rješenje je isključivo konačno stanje.



Optimizacija

- Problem zadovoljavanja ograničenja (eng. Constraint Satisfaction Problem): vrsta problema pretraživanja prostora stanja kod koje put od početnog do konačnog stanja nije bitan.
  - Rješenje je isključivo konačno stanje.
- Čvrsta ograničenja: nužno moraju biti ispunjena da bi rješenje bilo prihvatljivo.
  - Primjer rasporeda ispita: dva predmeta sa jedne studijske godine ne mogu biti u istom danu.
- Meka ograničenja: rješenja koja definišu poželjne (ili nepoželjne) osobine rješenja.
  - Primjer rasporeda ispita: razmak između uzastopnih ispita dva dana. Bolje je ono rješenje u kojem 390 studenata od 400 ima zadovoljen uslov, od slučaja u kojem samo 300 studenata ima zadovoljen uslov.

Heuristika

- Približni algoritmi, heurističke metode, heuristički algoritmi ili jednostavno heuristike.
- Algoritmi koji pronalaze rješenja koja su zadovoljavajuće dobra, ali ne nude nikakve garancije da će uspjeti pronaći optimalno rješenje, te koji imaju relativno nisku računsku složenost (tipično govorimo o polinomijalnoj složenosti).
- Dijelimo ih na:
  - konstrukcijske algoritme: rješenje problema grade dio po dio sve dok ne naprave cijelo rješenje.
    - Primjer: problem trgovackog putnika, započinje tako da nasumice odabere početni grad, a potom uvijek bira sljedeći najbliži grad.
  - algoritme koji koriste lokalnu pretragu: rješavanje započinje od nekog početnog rješenja koje potom pokušavamo inkrementalno poboljšati.

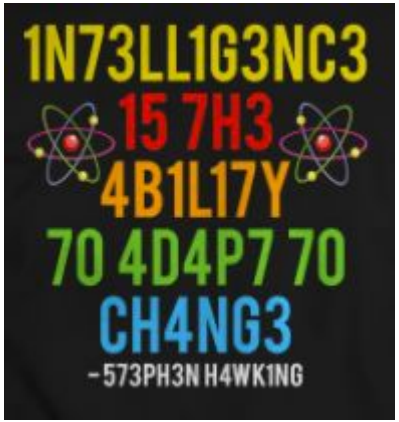
Metaheuristika

- Skup algoritamskih koncepata koje koristimo za definisanje heurističkih metoda primjenjivih na širok skup problema.
- Metaheuristika je heuristika opće namjene čiji je zadatak usmjeravanje problemski specifičnih heuristika prema području u prostoru rješenja u kojem se nalaze dobra rješenja.
  - Primjeri: simulirano kaljenje (Monte Carlo kaljenje, stohastičko hlađenje), tabu pretraživanje, algoritmi evolucijskog računanja i slicni.
- Možemo ih podijeliti na:
  - algoritmi koji rade nad jednim rješenjem (gdje spadaju algoritam simuliranog kaljenja te algoritam tabu pretrage) i
  - na populacijske algoritme koji rade sa skupovima rješenja (gdje spadaju algoritmi evolucijskog računanja).

Genetički algoritmi

Evoluciono računarstvo

- Inteligencija se definiše kao sposobnost sistema da se adaptira u okruženju koje se konstantno mijenja.
- Ako se zna da je ljudska vrsta proizvod **evolucije**, onda se postavlja pitanje da li se simuliranjem evolucije može kreirati inteligentni sistem?
- Oblast nauke koja se bavi simulacijom evolucije naziva se **evoluciono računarstvo**.
  - Optimizacijski algoritmi: iterativno poboljšavanje rješenja dok se ne nađe optimalno rješenje.



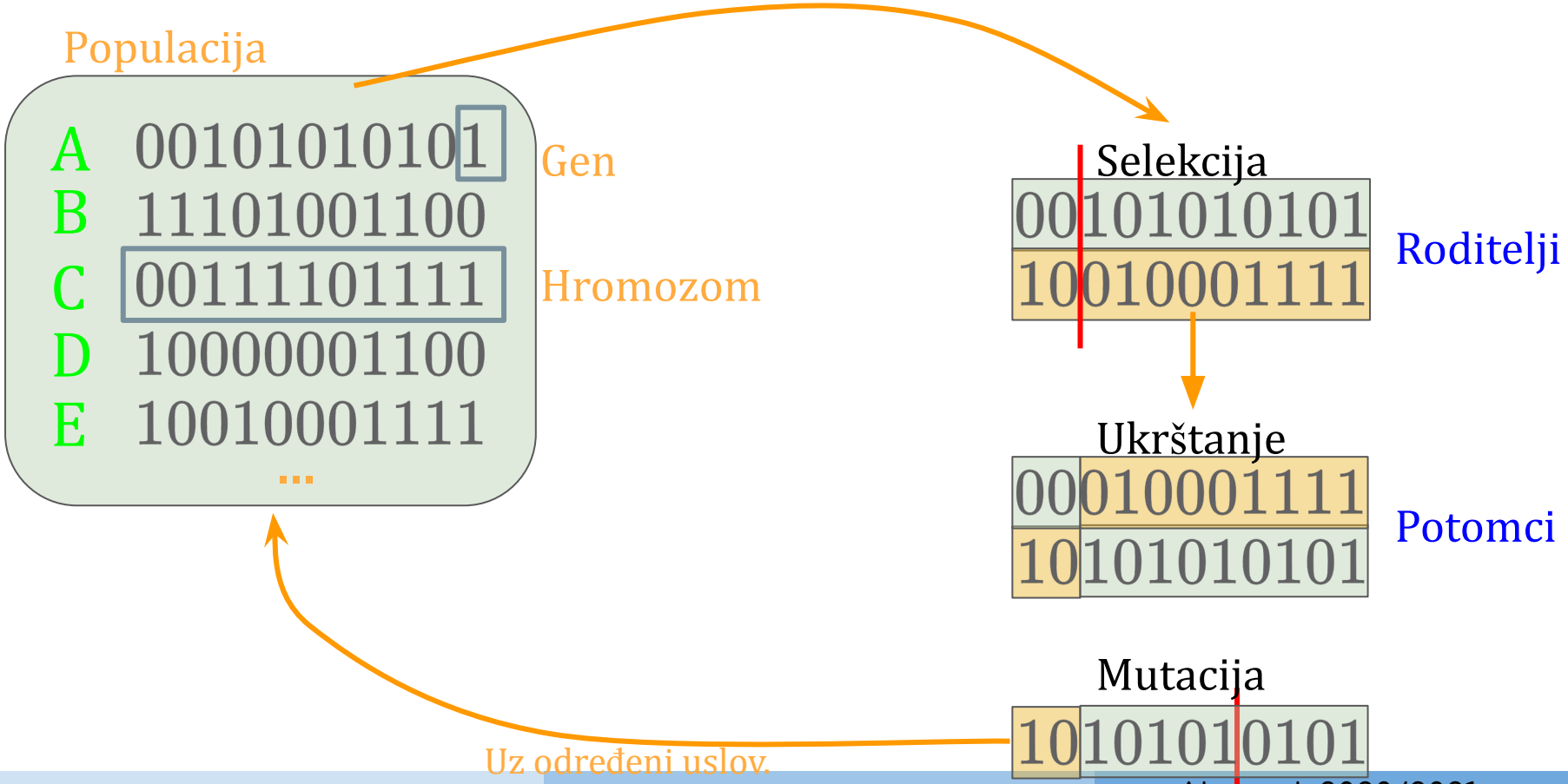
Evoluciono računarstvo

- **Evoluciono računarstvo** je jedan oblik mašinskog učenja koji podrazumijeva korištenje modela prirodne selekcije i genetike.
- Postoje različite tehnike u praksi, a neke od njih su genetički algoritmi, evolucione strategije i genetičko programiranje.

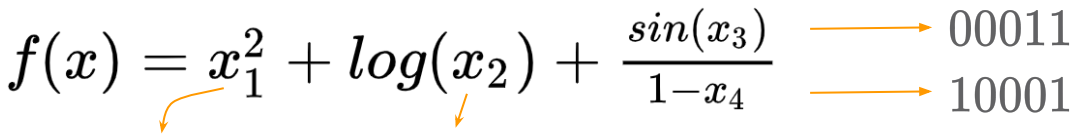
Problem: $f(x) = x_1^2 + log(x_2) + \frac{sin(x_3)}{1-x_4}$

Cilj: naći maksimalnu vrijednost funkcije f(x) ukoliko su dati neki uslovi.

Evoluciono računarstvo



Evoluciono računarstvo



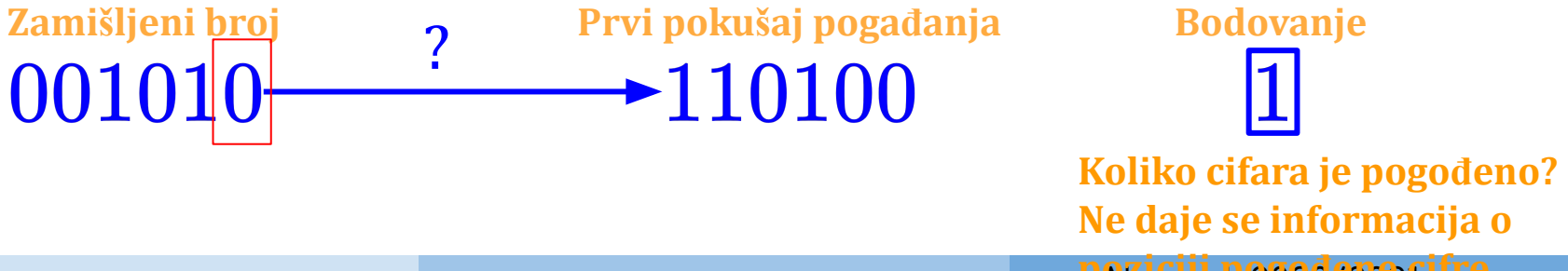
Traži se hromozom koji daje najbolje rješenje za dati problem (maksimizacija funkcije f(x)).

Genetički algoritmi

- **Genetički algoritmi** se definišu kao grupe računskih procedura koje konceptualno prate korake inspirisane biološkim procesima evolucije.
- Rezultati algoritma evoluiraju kroz nekoliko generacija sve dok se ne dođe do optimalnog ili blizu optimalnog rješenja. Poznati su i kao **evolucioni algoritmi**.
- Imaju veoma slične karakteristike kao najsposobniji biološki organizmi, poput osobina **samoorganizacije** i **adaptacije**.
- Metod ima mogućnost da uči na način da generiše sve bolje i bolje potomke, čija se sposobnost mjeri **funkcijom sposobnosti** ili **fitness funkcijom**.
- Genetički algoritmi korišteni su u velikom broju aplikacija kao što je usmjeravanje ili rutiranje vozila, predviđanje stečaja i pretraživanje Web-a.
- Koncept je prvi put predstavio John Holland 1975 godine.

Primjer 1: Igra pogađanja vektora brojeva

- Cilj igre je da se **pogodi broj koji je zamislio protivnik** na način da postoji određeni određena povratna sprega od strane protivnika, tj. postoji informacija o tome koliko je pogodak blizu rezultata.
- Povratna informacija se koristi kao znanje iz kojeg se generiše novi pokušaj.
- Za demonstraciju ove igre napravit ćemo neka ograničenja:
  - Broj koji se pogađa je **niz od šest cifara** ((u genetičkom algoritmu ovaj niz brojeva je sačinjen od hromozoma).
  - Svaka cifra može biti ili 0 ili 1.
- Igra je završena kada najbrži igrač pogodi rješenje.
- Cilj nije samo pogoditi broj nego pogoditi broj u što manjem broju pokušaja (što brže).

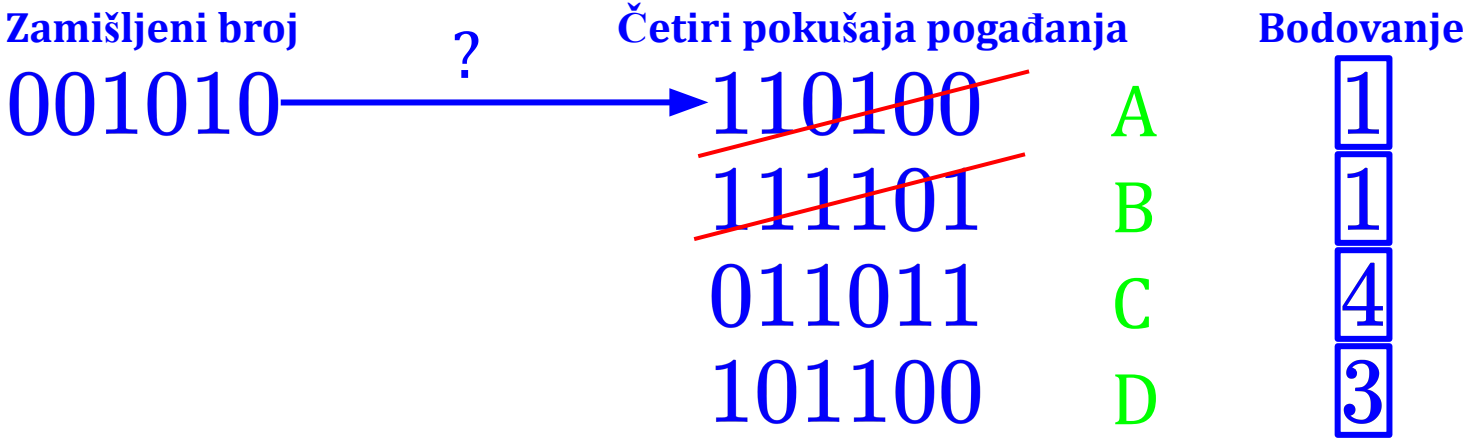


Primjer 1: Igra pogađanja vektora brojeva

- Koliko mogućih kombinacija ima? $2^6 = 64$
- Najjednostavnija metoda je metoda nasumičnog pokušaja i greške (*trial and error*).
  - Potrebno je u prosjeku 32 puta da se pogodi broj.
- Da li postoji bolji, tj. brži način za pogađanje broja?
  - Da, ako se ispravno interpretira povratna informacija od protivnika (mjera valjanosti ili fitness funkcija).

Primjer 1: Igra pogađanja vektora brojeva

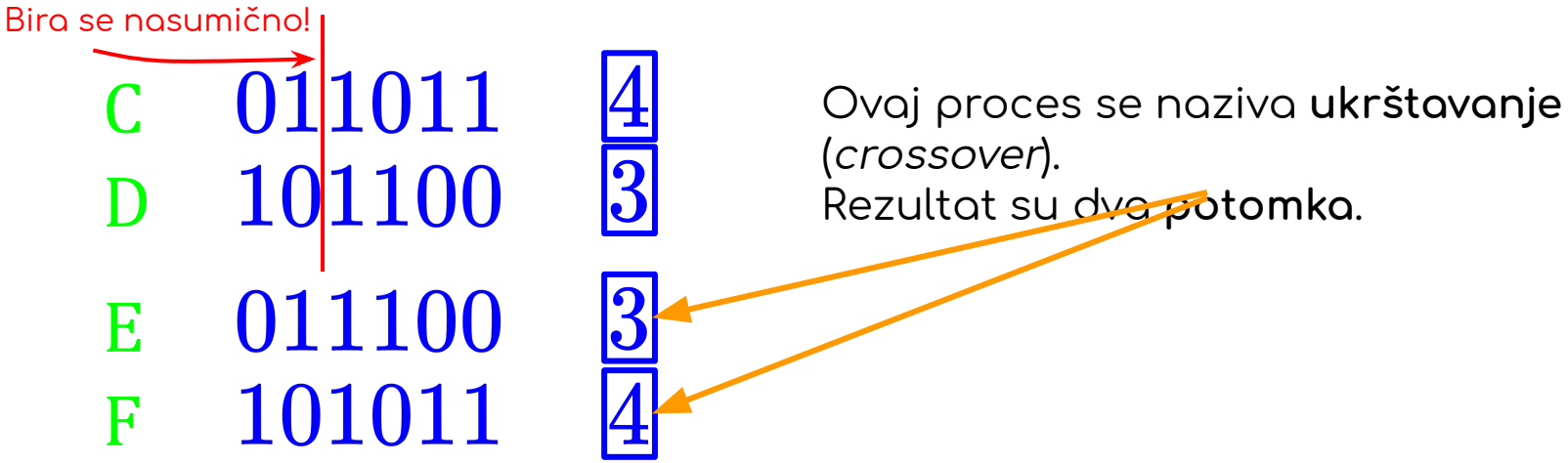
Korak 1: ponuditi četiri nasumično generisana odgovora.



Korak 2: ukloniti kombinacije koje imaju najmanje bodovanje. Dvije kombinacije koje imaju najveće bodovanje proglasiti roditeljima.

Primjer 1: Igra pogađanja vektora brojeva

Korak 3: Upariti dvije kombinacije iz koraka 2 na način da se nizovi rascjepe na dva dijela na proizvoljnom mjestu.

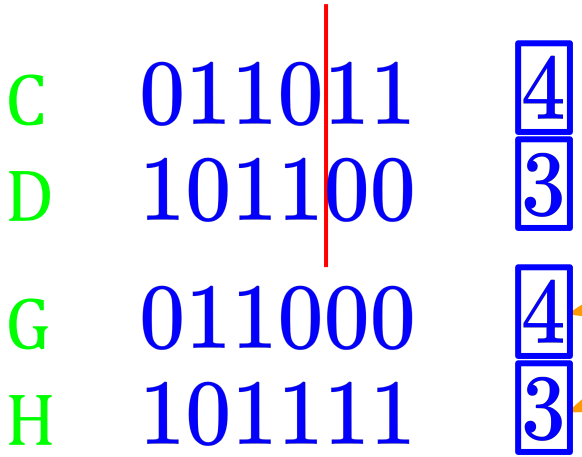


Nažalost, nije došlo u poboljšanju bodovanja u odnosu na roditelje.
Potrebno je pokušati rascjepiti početne kombinacije roditelja.

Primjer 1: Igra pogađanja vektora brojeva

Korak 4: Kopirati kombinacije C i D.

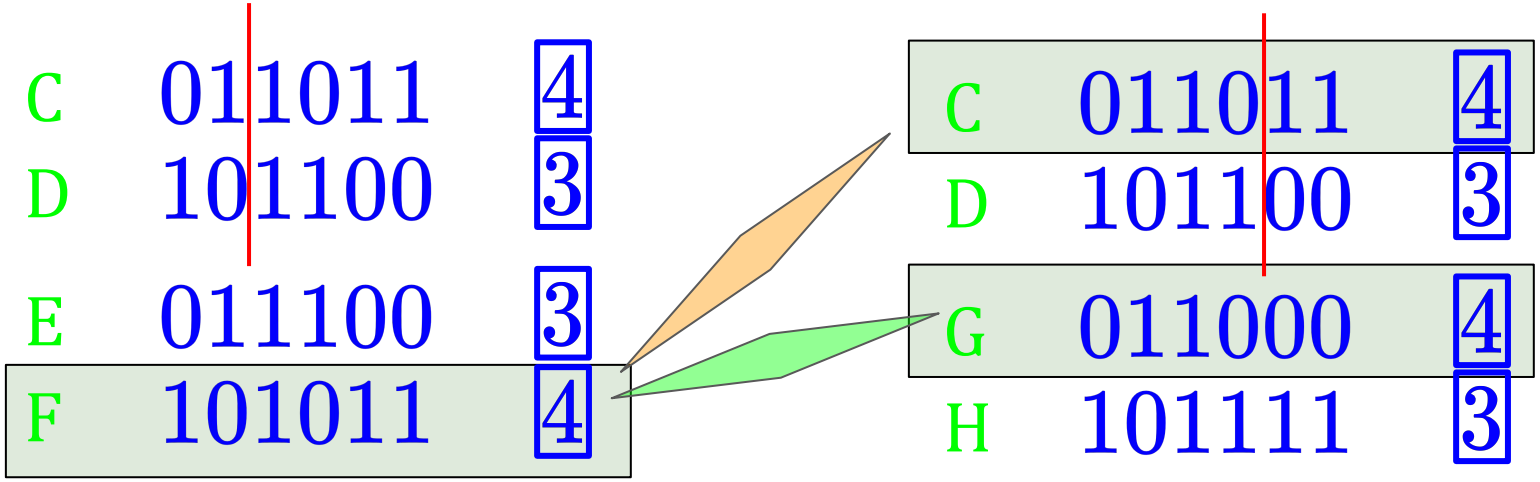
Korak 5: Upariti C i D na način da se kombinacije rascjepe na različitom mjestu.



Ovaj proces se naziva **ukrštavanje** (*crossover*).
Rezultat su dva **potomka**.

Dalje je potrebno ponoviti Korak 2: izabrati najbolje parove iz prethodnih rješenja za parenje. Postoji nekoliko kombinacija.

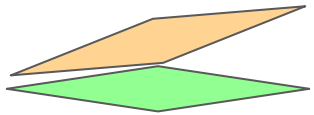
Primjer 1: Igra pogađanja vektora brojeva



Mogući parovi: 1) C i G, 2) F i G.

Primjer 1: Igra pogađanja vektora brojeva

F	101011	4
G	011000	4
I	111000	3
J	001011	5



F	101011	4
G	011000	4
K	101000	4
L	011011	4

Mogući parovi: 1) J i K, 2) J i L.

J	001011	5
K	101000	4
M	001010	6

Definicija i proces

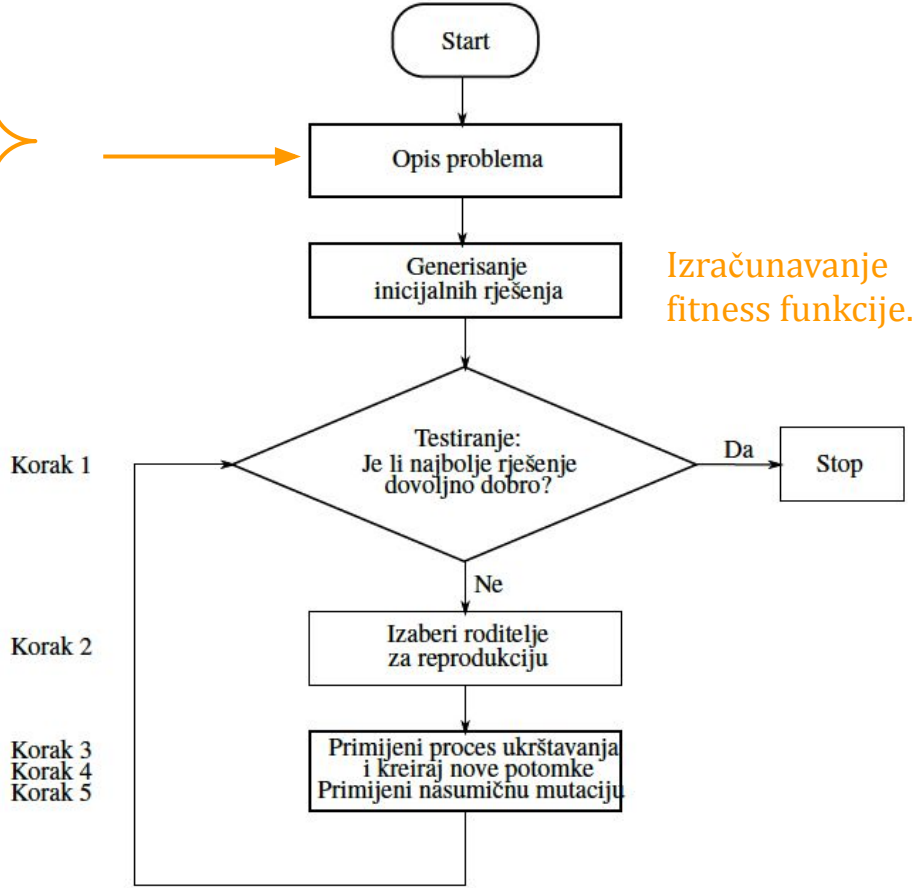
- Genetički algoritam je iterativna procedura koja moguća rješenja predstavlja kao nizove gena ili hromozoma, a zatim daje mjeru održivosti ili sposobnosti za preživljavanje koja se dobije putem **fitness funkcije**.
- Fitness funkcija je mjera cilja koja se želi postići. Obično se koristi funkcija minimuma ili maksimuma.
- Jedno moguće rješenje u jednoj iteraciji naziva se **generacija**.
- Iz jedne generacije roditelja i potomaka biraju se samo oni najsposobniji i oni postaju roditelji sljedećih potomaka.

Genetičke operacije

- U postupku generisanja sljedećih potomaka koriste se posebne genetičke operacije:
  - **Reprodukcija** (eng. *reproduction*): Genetički algoritam kroz proces reprodukcije generiše nove generacije poboljšanih rješenja na način da bira roditelje sa najvećim bodovanjem ili dodjeljivanjem veće vjerovatnoće za preživljavanje roditelja.
  - **Ukrštavanje** (eng. *crossover*): Ukrštavanje podrazumijeva izbor nasumičnog rascjepa u nizu i zamjenu segmenata dva niza. Mogu se zamijeniti segmenti lijevo ili desno od rascjepa, sve dok je jedan segment iz jednog broja a drugi iz drugog.
  - **Mutacija** (eng. *mutation*): Mutacija je proizvoljna promjena u situaciji. U nekim situacijama se koristi kako algoritam ne bi došao u stanje iz kojeg ne može nastaviti dalje. Procedura podrazumijeva zamjenu cifara 1 u 0 ili 0 u 1.

Proces genetičkog algoritma

- Napraviti procjenu da li se problem može riješiti genetičkim algoritmom.
- Ako da, moguće rješenje predstaviti kao niz 0 i 1.
- Formirati fitness funkciju tako da se ona može efikasno izračunati.
- Poslije generisanja inicijalnih rješenja, izračunavaju se **fitness funkcije svakog rješenja (fitness populacije)**, te se zatim računa suma svih fitness funkcija.
- Zatim se izračunava vjerovatnoća svakog rješenja u svrhu izbora u narednom koraku.



Parametri genetičkog algoritma

- Broj inicijalnih rješenja za generisanje.
- Broj potomaka za generisanje.
- Broj roditelja i potomaka koji će se zadržati za sljedeću generaciju.
- Vjerovatnoća mutacije (veoma niska).
- Distribucija vjerovatnoće pojave tačaka ukrštanja.

Primjer 2: Problem ranca

- Problem ranca je jednostavan organizacijski problem koji se može riješiti direktnom primjenom analitičkih metoda.
- U praksi se vrlo često koristi za rješavanje nekih kritičkih zadataka poput:
  - Šta ponijeti u šatlu na nekoj svemirskoj misiji?
  - Šta spakovati na robota koji se šalje na Mars?
- Koristi se kada je potrebno optimizirati broj stavki koje želimo ponijeti u rancu po nekom kriteriju.
  - Kriterij može biti ukupna težina (kg) stavki u odnosu na njihovu vrijednost/cijenu ili drugu vrijednost u cilju postizanja određenog cilja.

Stavka	1	2	3	4	5	6	7
Beneficija	5	8	3	2	7	9	4
Težina	7	8	4	10	4	6	4

- Primjer jednog rješenja: [1010100], pri čemu su izabrane stavke 1, 3 i 5, tj. Ukupna težina ranca je 5 + 3 + 7 = 15kg.
- Da li se primijenom genetičkog algoritma može doći do boljeg rješenja?

Primjer 2: Problem ranca

- Za generisanje rješenja koristimo sljedeće parametre:
  - Veličina populacije: 20
  - Broj generacija: 20
  - Vjerovatnoća ukrštanja: 0.7 (0.0 do 1.0)
  - Vjerovatnoća mutacije: 0.4 (0.0 do 1.0)
  - Kapacitet ranca: 7
  - Maksimalna težina ranca: 22

- Primjer rješenja: [0 1 0 0 1 1 1], pri čemu su izabrane stavke 2, 5, 6 i 7, tj. ukupna težina ranca je 22kg.
- Ukupna beneficija računa se kao suma beneficija pojedinačnih (izabranih) stavki, tj. iznosi 28.

Stavka	1	2	3	4	5	6	7
Beneficija	5	8	3	2	7	9	4
Težina	7	8	4	10	4	6	4

Jedno od mogućih rješenja: https://github.com/jmyrberg/mknapsack

Primjer 2: Problem ranca

- Za generisanje rješenja koristimo sljedeće parametre:
  - Veličina populacije: 20
  - Broj generacija: 20
  - Vjerovatnoća ukrštanja: 0.7 (0.0 do 1.0)
  - Vjerovatnoća mutacije: 0.4 (0.0 do 1.0)
  - Kapacitet ranca: 7
  - Maksimalna težina ranca: 22

Kako izabrati veličinu populacije?
- **Izabrati veći broj**: problem sa korištenjem velikog broja resursa.
- **Izabrati manji broj**: nema puno opcija za ukrštavanje.

- Primjer rješenja: [0 1 0 0 1 1 1], pri čemu su izabrane stavke 2, 5, 6 i 7, tj. ukupna težina ranca je 22kg.
- Ukupna beneficija računa se kao suma beneficija pojedinačnih (izabranih) stavki, tj. iznosi 28.

Stavka	1	2	3	4	5	6	7
Beneficija	5	8	3	2	7	9	4
Težina	7	8	4	10	4	6	4

Jedno od mogućih rješenja: https://github.com/jmyrberg/mknapsack

Primjer 2: Problem ranca

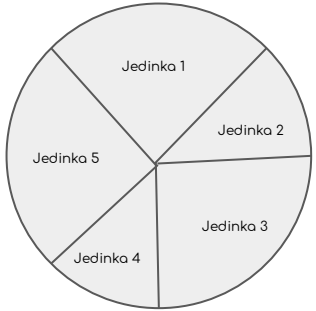
- Za generisanje rješenja koristimo sljedeće parametre:
  - Veličina populacije: 20
  - Broj generacija: 20
  - Vjerovatnoća ukrštanja: 0.7 (0.0 do 1.0)
  - Vjerovatnoća mutacije: 0.4 (0.0 do 1.0)
  - Kapacitet ranca: 7
  - Maksimalna težina ranca: 22

Kako izabrati roditelje?
- **Kada su vrijednosti fitness funkcije veoma različite:**

Selekcija na osnovu ranga: svi hromozomi se rangiraju na osnovu vrijednosti fitness funkcije. Izaberu se oni koji imaju najveću vrijednost ranga.

- Primjer rješenja: [0 1 0 0 1 1 1], pri čemu su izabrane stavke 2, 5, 6 i 7, tj. ukupna težina ranca je 22kg.
- Ukupna beneficija računa se kao suma beneficija pojedinačnih (izabranih) stavki, tj. iznosi 28.

Stavka	1	2	3	4	5	6	7
Beneficija	5	8	3	2	7	9	4
Težina	7	8	4	10	4	6	4



Jedno od mogućih rješenja: https://github.com/jmyrberg/mknapsack

Primjer 2: Problem ranca

- Za generisanje rješenja koristimo sljedeće parametre:
  - Veličina populacije: 20
  - Broj generacija: 20
  - Vjerovatnoća ukrštanja: 0.7 (0.0 do 1.0)
  - Vjerovatnoća mutacije: 0.4 (0.0 do 1.0)
  - Kapacitet ranca: 7
  - Maksimalna težina ranca: 22

Kako inicijalizirati rješenja?
- **Postoji nekoliko opcija:**
  - **Nasumično.**
  - Može se koristiti **domensko znanje** kako bi se "začela" nova populacija.
  - Uniformna kombinacija mogućih vrijednosti.

- Primjer rješenja: [0 1 0 0 1 1 1], pri čemu su izabrane stavke 2, 5, 6 i 7, tj. ukupna težina ranca je 22kg.
- Ukupna beneficija računa se kao suma beneficija pojedinačnih (izabranih) stavki, tj. iznosi 28.

Stavka	1	2	3	4	5	6	7
Beneficija	5	8	3	2	7	9	4
Težina	7	8	4	10	4	6	4

Jedno od mogućih rješenja: https://github.com/jmyrberg/mknapsack

Primjer 2: Problem ranca

- Za generisanje rješenja koristimo sljedeće parametre:
  - Veličina populacije: 20
  - Broj generacija: 20
  - Vjerovatnoća ukrštanja: 0.7 (0.0 do 1.0)
  - Vjerovatnoća mutacije: 0.4 (0.0 do 1.0)
  - Kapacitet ranca: 7
  - Maksimalna težina ranca: 22

Kada zaustaviti evoluciju?

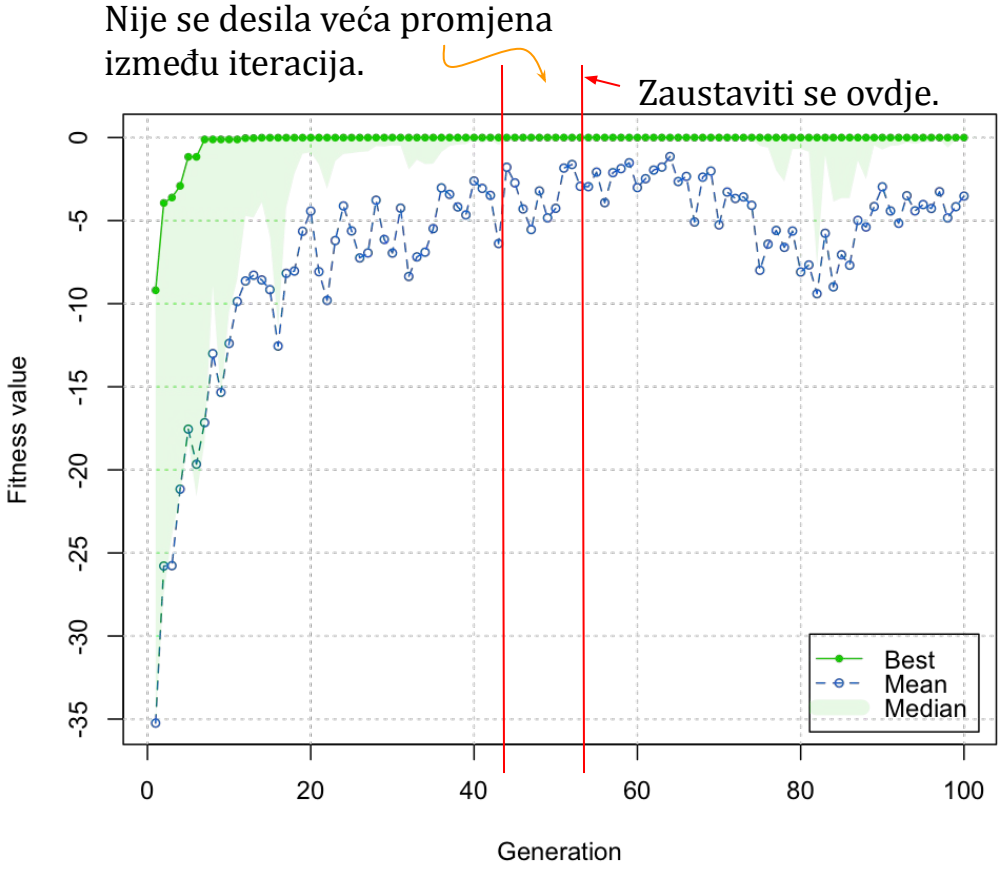
- **Postoji nekoliko opcija:**
  - Koristiti maksimalan broj operacija.
  - Minimalan opseg diverziteta (različitosti među jedinkama).
  - Ciljati neku vrijednost ukupnog fitnessa populacije.
  - Zaustavi se onda kada nema značajne promjene u fitness funkciji.

- Primjer rješenja: [0 1 0 0 1 1 1], pri čemu su izabrane stavke 2, 5, 6 i 7, tj. ukupna težina ranca je 22kg.
- Ukupna beneficija računa se kao suma beneficija pojedinačnih (izabranih) stavki, tj. iznosi 28.

Stavka	1	2	3	4	5	6	7
Beneficija	5	8	3	2	7	9	4
Težina	7	8	4	10	4	6	4

Jedno od mogućih rješenja: https://github.com/jmyrberg/mknapsack

Kada zaustaviti GA?



Zašto koristiti genetičke algoritme?

- **Prednosti:**
  - Jednostavni za kodiranje.
  - Daju mnogo rješenja: moguće je izbjeći lokalne ekstreme.
  - Pogodni za paralelizaciju!
- **Mane:**
  - Veoma su spori! Evolucija se ne može desiti brzo. Potrebno je vrijeme.
  - Izbor fitness funkcije možda nije trivijalan proces. Primjer funkcija: sigmod, *radial basis function* (RBF), ...