

# Rešavanje slagalice primenom genetskog algoritma

Projekat u okviru kursa Računarska inteligencija  
Matematički fakultet  
Univerzitet u Beogradu

Lazar Čeliković  
[celikoviclazar@hotmail.com](mailto:celikoviclazar@hotmail.com)  
Miloš Milaković  
[mi17152@alas.matf.bg.ac.rs](mailto:mi17152@alas.matf.bg.ac.rs)

Avgust 2021

# Sadržaj

<b>1 Opis problema</b>	<b>3</b>
<b>2 Implementacija</b>	<b>3</b>
2.1 Obrada ulaznih podataka . . . . .	3
2.2 Genetski algoritam . . . . .	3
2.2.1 Uopšteno o algoritmu . . . . .	3
2.2.2 Implementacija jedinke . . . . .	3
2.2.3 Selekcija, ukrštanje, elitizam . . . . .	4
2.2.4 Parametri genetskog algoritma . . . . .	5
2.3 Izlaz programa . . . . .	5
<b>Rezultati</b>	<b>5</b>
<b>3 Rezultati</b>	<b>5</b>
<b>4 Zaključak</b>	<b>7</b>
<b>Literatura</b>	<b>8</b>

# 1 Opis problema

Problem koji se u ovom radu rešava jeste problem slaganja slagalice. Na ulazu se dobija niz izmešanih delova i cilj je složiti sve delove na svoje mesto. U ovom radu je dat postupak implementacije automatizacije ovog procesa pomoću genetskog algoritma.

## 2 Implementacija

### 2.1 Obrada ulaznih podataka

Na ulazu u algoritam dobijamo fotografiju od koje želimo da napravimo slagalicu. Takođe, među parametrima zadajemo i dimenzije jednog dela. Uz pomoć tog parametra sećemo sliku i na taj način formiramo niz delova slagalice. Taj niz nam postaje ulaz u genetski algoritam. Svaki deo je predstavljen kao matrica piksela. Dakle, za svaki piksel pamtimo njegovu RGB vrednost i te informacije koristimo za kalkulaciju fitnesa što će biti objašnjeno kasnije u radu.

### 2.2 Genetski algoritam

#### 2.2.1 Uopšteno o algoritmu

Osnovna ideja genetskog algoritma meže se opisati u nekoliko koraka:

1. Definisanje svih potrebnih parametara problema i genetskog algoritma
2. Formiranje inicijalne populacije
3. Dekodiranje hromozoma — ovaj korak se javlja samo kod binarnih GA
4. Određivanje cena hromozomima (fitnes funkcijom)
5. Selekcija hromozoma koji će postati ulaz za fazu ukrštanja
6. Ukrštanje — iz populacije se izdvajaju dva hromozoma (roditelji) i njihov genetski materijal se ukršta. Kao proizvod tog ukrštanja nastaje jedan ili više potomaka koji bi trebalo da imaju bolji genetski materijal (u terminima fitnes funkcije) od roditelja
7. Mutacije, pri kojima se menja genetski sadržaj hromozoma
8. Ispitivanje konvergencije, da bi se utvrdilo da li ima osnova da se tok algoritma prekine. Ukoliko nije ispunjen uslov konvergencije, vratiti se na korak 3 odnosno 4.

#### 2.2.2 Implementacija jedinke

Svaka jedinka, odnosno hromozom predstavljen je kao objekat koji sadrži niz elemenata slagalice. Dakle, hromozom predstavlja jedno potencijalno rešenje slagalice. Inicijalna populacija onda predstavlja niz potencijalnih rešenja. Ona se kreira tako što se napravi niz delova od ulazne fotografije i onda se od tog niza pravi potreban broj jedinki nasumičnim mešanjem tih delova.

Takođe, svaka jedinka ima polje koje predstavlja vrednost fitnes funkcije. Ta vrednost označava koliko je ta konkretna jedinka dobra, odnosno koliko je bliska polaznoj fotografiji u našem slučaju. Ideja za računanje ove vrednosti korišćena u našoj implementaciji jeste da se svaki deo poredi sa svojim susedima, odnosno da im se porede ivice. Intuicija iza ovoga jeste da ako delovi treba da stoje jedan pored drugoga onda bi RGB vrednosti trebalo da im budu slične. Dakle, fitnes funkcija se računa oduzimanjem RGB vrednosti za dve ivice i sabiranjem.

$$f(x) = \sum(((w_1 - w_2)/255)^2)$$

U prethodnoj formuli  $w_1$  i  $w_2$  predstavljaju ivice nekog dela slagalice. Deljenje sa 255 koristimo da bismo doveli vrednosti RBG za piksele na interval  $[0, 1]$ . Dakle, što je vrednost fitnes funkcije manja veća je verovatnoća da je ta jedinka naše rešenje.

### 2.2.3 Selekcija, ukrštanje, elitizam

Prilikom implementacije podržana je i turnirska i ruletska selekcija. Parametre koje koristimo prilikom vršenja selekcije prosledjujemo algoritmu na početku.

Ukrštanje je posebno zanimljivo u ovom slučaju. Ideja je da se uzmu dve jedinke i da se od njih ukrštanjem pravi jedan potomak. Izazov ovde jeste u tome što moramo da budeo vrlo pažljivi koji delove ubacujemo u jezgro jer može vrlo lako da se desi da ubacimo isti deo iz oba roditelja i onda narušavamo inicijalnu strukturu jedinke. Proces ukrštanja jedinki sastoji se iz tri faze. Sada ćemo ih ukratko opisati.

Prva faza se sastoji od sledećeg. Naime, krećemo od proizvoljnog dela jednog roditelja i tražimo sve susede tog dela koji su isti kod oba roditelja. Intuicija iza ovog postupka jeste u tome da ako se delovi kod oba roditelja nalaze jedni pored drugih veća je verovatnoća da je to korektan raspored. Rezultat ove faze biće jezgro koje predstavlja matricu delova. Vrlo je verovatno da će neka polja u jezgru ostati nepopunjena. To se rešava u narednim fazama.

Druga faza podrazumeva traženje takozvanog best buddy dela. Naime, kada smo dobili jezgro koje čine delovi koji su isto raspoređeni kod oba roditelja prelazimo na traženje onih delova koji se najbolje uklapaju sa već složenim delovima. Kada pronađemo deo za koji smatramo da je najbolji za tu poziciju ubacujemo ga na istu. I ovde kao i u prethodnoj fazi moramo da vodimo računa o granicama jezgra koje konstantno raste. Znamo da je slagalica dimenzija  $n \times m$  i samim tim moramo da pazimo da rezultat ukrštanja takođe bude tih dimenzija. Na primer, ako želimo da ubacimo deo desno od najdešnjeg dela u jezgru, moramo da proverimo da li ta pozicija upada u granice jezgra. Ako ne, pokušavamo da transliramo sve delove u levo za jednu poziciju ako je to moguće. Ukoliko je to moguće, ubacujemo izabrani deo na najdešnju poziciju i nastavljamo dalje. Ukoliko transliranje nije bilo moguće jednostavno vraćamo taj deo među neraspoređene i nastavljamo sa algoritmom.

Treća faza podrazumeva uzimanje preostalih delova i nasumično raspoređivanje na preostala slobodna mesta u jezgru.

Nakon ovoga dobijamo novu jedinku koja takođe ima niz delova slagalice. Sam postupak nam garantuje da će dimenzije novonastale jedinke odgovarati domenizijama roditelja kao i to da će se svaki deo naći u novonastaloj jedinci tačno jednom.

Elitizam je realizovan pre procesa selekcije. Automatski je uzeto n jedinki i one su ubaćene u narednu generaciju. Nakon toga se proces selekcije primenjuje na preostale jedinke. Takođe, među parametrima algoritma se može zadati broj jedinki za elitizam kao i opcija da ne želimo da imamo elitizam uopšte.

#### 2.2.4 Parametri genetskog algoritma

Parametri za ovaj algoritam se prosleđuju prilikom pokretanja komande iz terminala. U odeljku koji se bude bavio analizom rezultata će biti prikazano kako se algoritam ponaša za različite vrednosti parametara. Neki od parametara koji se mogu zadavati su broj generacija, broj jedinki za elitizam, veličina populacije, tip selekcije i veličina dela slagalice.

### 2.3 Izlaz programa

Kada se algoritam završi kao rezultat dobijamo fotografiju koja se dobija spajanjem svih delova slagalice. Pored fotografije, program nam ispisuje još neke korisne statistike kao što je vreme koje je bilo potrebno da se dođe do rešenja, da li se algoritam zaustavio ranije jer je iskonvergirao... Ove statistike će biti korišćenje u odeljku koji se bavi prikazivanjem i analizom rezultata.

## 3 Rezultati

Sledi par primera rešavanja slagalice i prikaz izgleda slagalice u određenim generacijama.



(a) Početni izgled slagalice

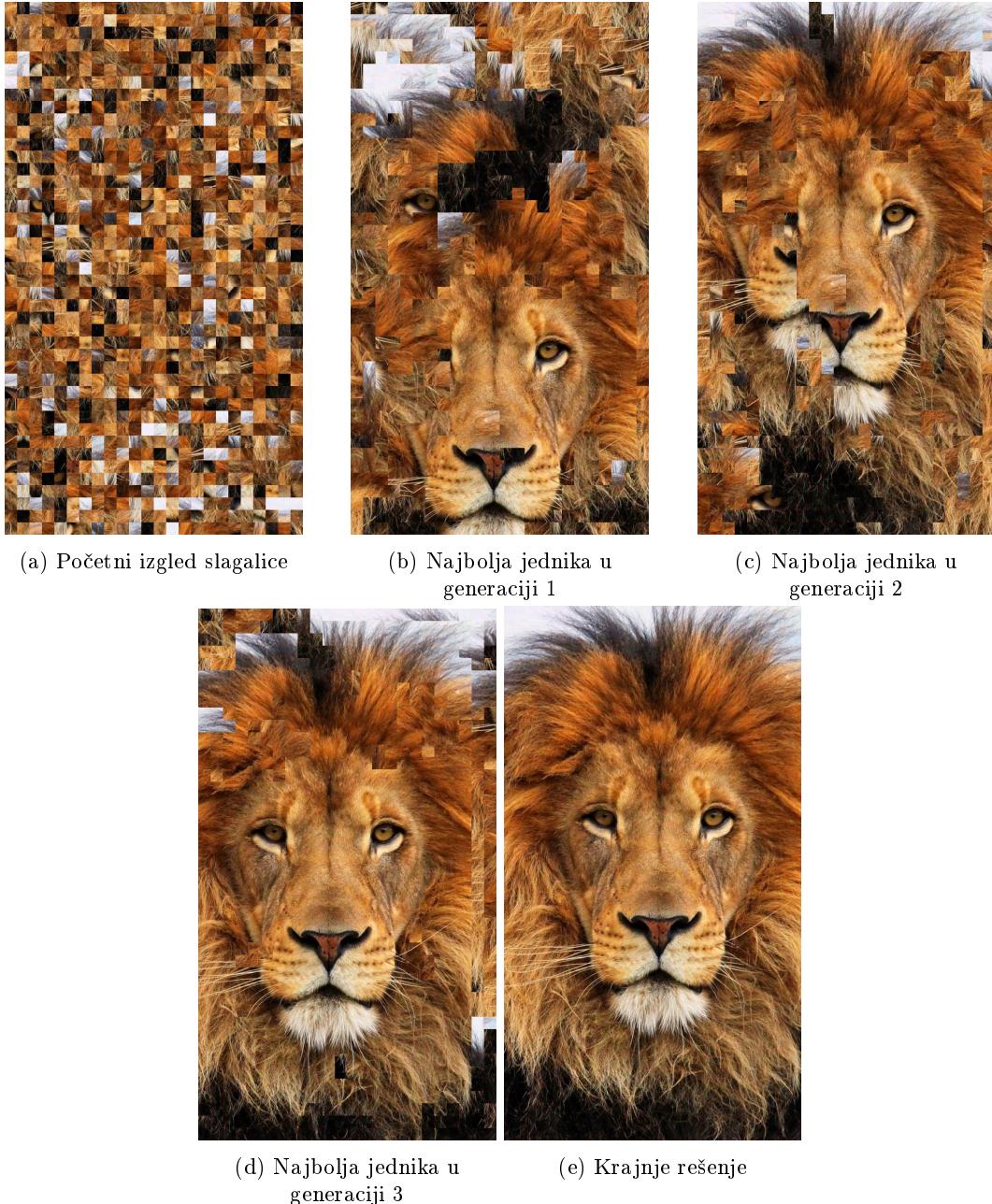


(b) Najbolja jednika u generaciji 1



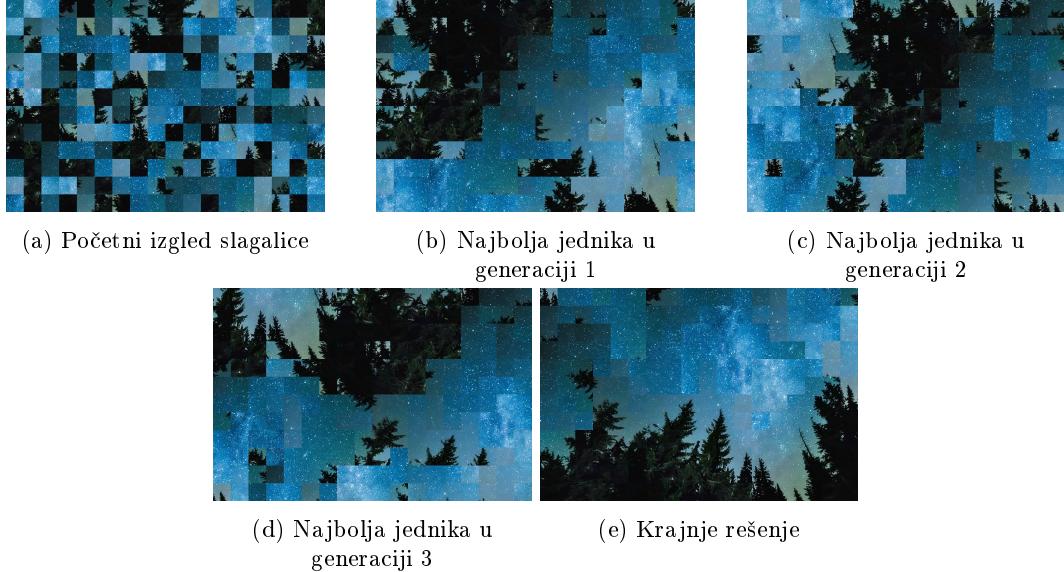
(c) Krajnje rešenje se dostiže u trećoj generaciji

Slika 1: Primer rešavanja slagalice lava. Veličina dela je  $48 \times 48$  piksela a broj delova slagalice je 405. Potrebno vreme da reši je 66.153 sekunde.



Slika 2: Primer rešavanja slagalice lava. Veličina dela je  $30 \times 30$  piksela a broj delova slagalice je 1032. Potrebno vreme da reši je 230.02 sekunde.

Naredni primer je primer slagalice u kojoj prilikom rešavanja dolazi do problema jer su svuda boje slične i ne može da se napravi jasna razlika između delova.



Slika 3: *Primer rešavanja slagalice noćnog neba. Veličina dela je 48x48 piksela a broj delova slagalice je 216. Potrebno vreme da reši je 70.481 sekunde.*

U zavisnosti od broja delova slagalice algoritmu će biti potrebno više ili manje vremena da reši slagalicu. Takođe povećavanjem broja generacija i jedinki u generaciji može da se dobije bolje rešenje ali će isto tako i vreme izvršavanja programa da raste. Mi smo testirali ove primere sa 20 generacija i 600 jedinki u jednoj generaciji.

## 4 Zaključak

Na osnovu rezultata iz prethodnog odeljka možemo zaključiti da naša implementacija ovog algoritma daje zadovoljavajuće rezultate. Za veliki broj ulaznih fotografija se dobija perfektno složena slika, međutim postoje i slučajevi kada rešenje i nije tako dobro. Primer takvog slučaja jeste kada imamo deo slike koji je veoma monoton, odnosno jednobojan. Algoritam tu vidi da su RGB vrednosti slične i uklapa te elemente. Problem je što u većini slučajeva ispermutuje te elemente odnosno ne može da odredi tačne pozicije tih elemenata jer su svi prilično slični. Ovo jeste veoma lep prostor za modifikaciju i dalje unapredivanje ovog algoritma.

## Literatura

- [1] A Genetic Algorithm-Based Solver for Very Large Jigsaw Puzzles
- [2] A Genetic Algorithm-Based Solver for Small-Scale Jigsaw Puzzles
- [3] A Genetic Algorithm-Based Solver for Jigsaw Puzzles