

# Non-dominated Sorting Genetic Algorithm (NSGA-II)

Seminarski rad u okviru kursa

Računarska inteligencija

Matematički fakultet

Staša Dorđević

8. januar 2025.

## Sažetak

Sažetak ako treba?

## 1 Uvod u genetske algoritme

Genetski algoritmi predstavljaju grupu optimizacionih metoda koje se zasnivaju na principima prirodne selekcije i evolucije. Inspirisani su biološkim procesima kao što su selekcija, ukrštanje (kroz reprodukciju), mutacija i nasleđivanje, koji omogućavaju preživljavanje i adaptaciju organizama u prirodi. Slično tome, osnovni koraci u implementaciji genetskih algoritama uključuju selekciju, ukrštanje i mutaciju. Ovi koraci se ponavljaju kroz više generacija kako bi se iz populacije rešenja razvila najbolja moguća rešenja za dati problem.

- **Selekcija** je proces odabira jedinki za ukrštanje na osnovu njihove prilagođenosti. Postoje dve osnovne varijante selekcije:
  1. **Turnirska selekcija** - Odabir k slučajnih jedinki iz populacija i "turnirsko takmičenje" gde pobeđuje najprilagođenija od izabranih jedinki
  2. **Ruletska selekcija** - U ovoj metodi, verovatnoća selekcije jedinke zavisi od njene uspešnosti u odnosu na ostale jedinke u populaciji. Prilagođenije jedinke imaju veću verovatnoću da budu izabrane, slično kao u ruletu.
- **Ukrštenje** omogućava kombinovanje gena odabranih jedinki, stvarajući nove potomke koji mogu naslediti najbolje karakteristike svojih "roditelja". Postoji nekoliko osnovnih varijanti ukrštanja:
  1. **Jednopoloziciono ukrštanje** - Genetski materijal od roditelja se deli na osnovu jedne slučajno odabrane tačke preseka, a potomci nasleđuju deo od oba roditelja prema toj tački.
  2. **Višepoloziciono ukrštanje** - Ova metoda koristi više tačaka preseka na genomima roditelja, što omogućava veću raznovrsnost u potomcima.
  3. **Uniformno ukrštanje** - U ovoj varijanti, gene sa oba roditelja se nasumično kombinuju kako bi se stvorio potomak, bez fiksnih tačaka preseka.

- **Mutacija** se koristi da bi se unela nasumična promena u genetski kod jedinke, što omogućava istraživanje novih mogućnosti i sprečava algoritam da se „zaglavi“ u lokalnim ekstremumima.

**Elitizam** je metoda koja garantuje da će najbolje jedinke iz trenutne generacije biti prenete u sledeću generaciju bez promena. Elitizam se koristi kako bi se sprečilo da se najbolja rešenja izgube tokom evolucije.

Kroz ove procese, genetski algoritmi omogućavaju efikasno istraživanje prostora rešenja i postepeno poboljšanje kvaliteta rešenja tokom vremena.

## 2 Opis algoritma NSGA-II

NSGA je popularan genetski algoritam zasnovan na nedominaciji za višeciljnu optimizaciju. Njegova modifikovana verzija, NSGA-II, koja rešava neke probleme zbog kojih je kritikovana osnovna verzija algoritma, često se koristi kao efikasnije rešenje u primenama višeciljne optimizacije. Višeciljna optimizacija podrazumeva istovremenu optimizaciju dva ili više međusobno suprotstavljenih ciljeva. Cilj je naći skup rešenja koji je najbolji kompromis između ciljeva. Ta rešenja formiraju tzv. **Pareto front**, u kojem nijedno rešenje nije bolje od drugog, osim ako se jedan cilj ne poboljša na račun pogoršanja drugog.

U NSGA-II algoritmu, termini *non-dominated* i *dominated* se koriste da opišu odnos između rešenja na osnovu njihovih performansi u odnosu na više ciljeva optimizacije.

Rešenje se smatra **nedominiranim** (engl. *non-dominated*) u odnosu na drugo rešenje ako nijedno od njih nije bolje u svim ciljevima. Drugim rečima, rešenje  $A$  je *nedominirano* u odnosu na rešenje  $B$  ako:

- $A$  nije lošije u svim ciljevima od  $B$ ,
- i  $B$  nije lošije u svim ciljevima od  $A$ .

Rešenje se smatra **dominiranim** (engl. *dominated*) u odnosu na drugo rešenje ako postoji rešenje koje je bolje u svim ciljevima. Drugim rečima, rešenje  $A$  je *dominirano* u odnosu na rešenje  $B$  ako:

- $B$  je bolje ili jednako u svim ciljevima od  $A$ ,
- i u barem jednom cilju  $B$  je bolje od  $A$ .

Kratak opis algoritma: Prvo se populacija inicijalizuje na standardan način, u skladu sa problemom koji rešavamo. Nakon toga, jedinke u njoj se sortiraju po frontovima prema principu nedominacije. Prvi front je potpuno nedominirani skup u trenutnoj populaciji, tj. skup svih rešenja od kojih ne postoji bolje rešenje u svim ciljevima. Drugi front sadrži jedinke koje su dominirane samo od strane jedinki iz prvog fronta, i tako dalje. Svakoј јединки se dodeljuje rang na osnovu fronta kojem pripadaju - one iz prvog fronta dobijaju rang 1, iz drugog 2, i tako dalje. Pored ranga, svaka jedinka ima i novi parametar - *distanca gužve* (engl. *crowding distance*). To je mera koja se koristi za održavanje raznolikosti između rešenja unutar jednog pareto fronta. Predstavlja meru bliskosti jedinke njenim susedima. Veća prosečna distanca gužve rezultira boljom raznovrsnošću u populaciji. Favorizuje manje naseljene regione. Nakon sortiranja, unutar svakog fronta, računa se distanca gužve za jedinke u tom frontu. Primarni kriterijum za selekciju je

rang. Ako dve jedinke imaju isti rang, preferira se ona sa većom distancom gužve. Ovaj pristup osigurava da algoritam održava i intenzifikaciju (kroz rang) i diverzifikaciju (kroz distancu gužve). Roditelji se biraju iz populacije koristeći turnirsku selekciju. Odabrana populacija generiše potomke pomoću operacija ukrštanja i mutacije, koje će biti detaljnije opisane u narednom poglavlju. Populacija, zajedno sa trenutnom populacijom i trenutnim potomcima, ponovo se sortira prema principu nedominacije, i samo se najboljih N jedinki selektuje, gde je N veličina populacije. Selekcija se zasniva na rangui i distanci gužve u poslednjem pareto frontu.

## 3 Opis mog rešenja

Moje rešenje - implementacija opis

### 3.1 Grupisanje u pareto frontove - sortiranje

Ovaj algoritam koristi metod nedominiranog sortiranja da bi organizovao populaciju u Pareto frontove. U prvom koraku se inicijalizuje broj dominacija i lista dominiranih jedinki za svaku jedinku. Zatim, kroz dvostruki for petlju, algoritam poredi svaku jedinku sa svim ostalim u populaciji i ažurira broj dominacija i listu dominiranih jedinki. Nakon što su svi odnosi dominacije utvrđeni, jedinke se grupišu u Pareto frontove prema njihovoj dominaciji. Nakon što je jedinka pridružena određenom Pareto frontu, dodeljuje joj se rang na osnovu njega, koji će nam kasnije koristiti u selekciji.

```

1: Input: populacija
2: Output: pareto frontovi
3: pareto frontovi ← prazna lista
4: broj dominacija ← prazna mapa
5: dominirane jedinke ← prazna mapa
6: n ← dužina populacije
7: for svaku jedinku u populaciji do
8:   broj dominacija[jedinka] ← 0
9:   dominirane jedinke[jedinka] ← prazna lista
10: end for
11: for i = 0 to n-1 do
12:   for j = 0 to n-1 do
13:     if i ≠ j then
14:       if dominira(populacija[i], populacija[j]) then
15:         dodaj populacija[j] u dominirane jedinke[populacija[i]]
16:       else if dominira(populacija[j], populacija[i]) then
17:         broj dominacija[populacija[i]] ← broj dominacija[populacija[i]] + 1
18:       end if
19:     end if
20:   end for
21: end for
22: trenutni front ← sve jedinke sa brojem dominacija 0
23: trenutni indeks ← 1
24: while trenutni front nije prazan do
25:   dodaj trenutni front u pareto frontovi

```

```

26: sledeći front  $\leftarrow$  prazna lista
27: for svaku jedinku u trenutnom frontu do
28:   rang jedinke  $\leftarrow$  trenutni indeks
29:   for svaku dominiranu jedinku do
30:     broj dominacija[dominirana]  $\leftarrow$  broj dominacija[dominirana] - 1
31:     if broj dominacija[dominirana] == 0 then
32:       dodaj dominiranu u sledeći front
33:     end if
34:   end for
35: end for
36: trenutni front  $\leftarrow$  sledeći front
37: trenutni indeks  $\leftarrow$  trenutni indeks + 1
38: end while
39: return pareto frontovi

```

## 3.2 Određivanje distance gužve

Ova funkcija izračunava distancu gužve za svaku jedinku u Pareto frontu, koristeći sve ciljeve optimizacije. Za svaku jedinku u Pareto frontu, vrednost distance gužve se inicijalizuje na 0. Zatim se za svaki cilj, Pareto front sortira prema vrednostima cilja, a prvoj i poslednjoj jedinki dodeljuje se beskonačna vrednost distance gužve. Distanca za svaku jedinku i svaki cilj se izračunava kao odnos razlike između fitnesa susednih jedinki i razlike između minimalne i maksimalne vrednosti fitnesa (unutar trenutnog Pareto fronta) za dati cilj. Izračunata distanca se dodaje na ukupnu distancu gužve. Ovaj proces se ponavlja za svaki cilj.

```

1: Input: pareto front, broj ciljeva
2: Output: ažurirani pareto front sa izračunatim distancama gužve
3: n  $\leftarrow$  dužina pareto fronta
4: for svaka jedinka u pareto frontu do
5:   jedinka.distanca gužve  $\leftarrow$  0
6: end for
7: for i = 0 to broj ciljeva - 1 do
8:   sortiraj pareto front prema fitnesu[i]
9:   pareto front[first].distanca gužve  $\leftarrow \infty$ 
10:  pareto front[last].distanca gužve  $\leftarrow \infty$ 
11:  f_min  $\leftarrow$  pareto front[first].fitness[i]
12:  f_max  $\leftarrow$  pareto front[last].fitness[i]
13:  for k = 1 to n-2 do
14:    distanca  $\leftarrow$  (pareto front[k+1].fitness[i] - pareto front[k-1].fitness[i]) / (f_max - f_min)
15:    pareto front[k].distanca gužve  $\leftarrow$  pareto front[k].distanca gužve + distanca
16:  end for
17: end for
18: return pareto front

```

### **3.3 Selekcija**

### **3.4 Ukrštanje**

### **3.5 Mutacija**

### **3.6 Algoritam**

### **3.7 Test primeri**

## **4 Eksperimentalni rezultati**

Moji rezultati - grafici

## **5 Poređenje mojih rezultata i onih iz literature**

Poređenje rezultata - vizuelno i tekstualno

## **6 Zaključak**

Kritički osvrt na sve što je urađeno i eventualni pravci daljeg unapređivanja

## **Literatura**

- [1] A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II at:  
<https://ieeexplore.ieee.org/abstract/document/996017>
- [2] A fast elitist multiobjective genetic algorithm at:  
[https://www.academia.edu/download/53297141/NSGA\\_II.pdf](https://www.academia.edu/download/53297141/NSGA_II.pdf) - ovde algoritam  
za polinomijalnu mutaciju i sbx - najvise odavde u implementaciji
- [3] Materijali sa kursa Računarska inteligencija
- [4] Analyzing the Simulated Binary Crossover Operator in Multi-Objective Evolutionary Algorithms  
<https://www.duo.uio.no/handle/10852/111478>