# Domaći 1 iz Veštačke inteligencije

Nemanja Saveski 2019/0056

15. april 2023.

**Zadatak:** Potrebno je isprobati 5 različitih algoritama lokalne pretrage:

1. nasumična pretraga
2. gramziva pretraga
3. simulirano kaljenje
4. pretraga po snopu
5. genetski algoritam

nad magičnim kvadratom radi optimizacije rešenja. Magični kvadrat je matrica NxN popunjena brojevima od brojevima od 1 do $N^2$, gde je svaki broj pojavljuje tačno jednom i zbir elemenata svake kolone, vrste kao i po obe dijagonale iznosi $N(N^2+1)/2$.

**Izveštaj:**

**1**) Rešenja su predstavljena kao matrice NxN tako što je prvo napravljen niz od $N^2$ članova, zatim slučajno izmešan i na kraju resizeovan u matricu NxN. Primer jednog generisanog rešenja (matrica veličine 3x3):

$$\begin{bmatrix} 4 & 2 & 7 \\ 1 & 3 & 9 \\ 5 & 8 & 6 \end{bmatrix}$$

Treba napomenuti da je vremenska složenost algoritma koji računa ciljnu funkciju trenutnog rešenja jednaka $O(N^2)$.

**2**) Generisanje suseda se vrši tako što se u trenutnom rešenju (tj. trenutnoj matrici), rotira po jedna podmatrica veličine 2x2 za 90 stepeni u smeru kazaljke na satu (jedna rotacija generise jednog suseda), a na primeru generisanje prvog suseda izgleda ovako:

$$\begin{bmatrix} 4 & 2 & 7 \\ 1 & 3 & 9 \\ 5 & 8 & 6 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 4 & 7 \\ 3 & 2 & 9 \\ 5 & 8 & 6 \end{bmatrix}$$

Takođe, moguće je rotirati na isti način i elemente iz prve i poslednje vrste, kao i prve i poslednje kolone. Primer za kolone:

$$\begin{bmatrix} 4 & 2 & 7 \\ 1 & 3 & 9 \\ 5 & 8 & 6 \end{bmatrix} \rightarrow \begin{bmatrix} 7 & 2 & 9 \\ 4 & 3 & 1 \\ 5 & 8 & 6 \end{bmatrix}$$

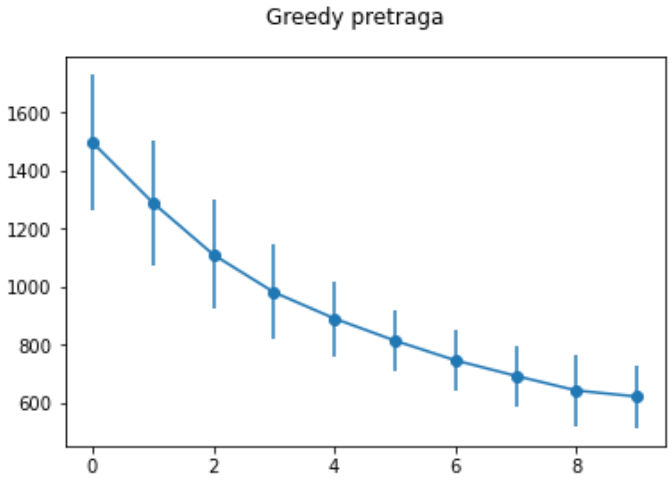
Pošto imamo da je red kvadrata jednak N, onda imamo da svaki put iz trenutnog rešenja možemo generisati $N^2 - 1$ suseda, jer toliko je tačaka gde se susreću 4 polja. Ovaj algoritam ima vremensku složenost $O(N^2)$.

**3**) Tabela u kojoj se upoređuju prosek i standardne devijacije između algoritama testirane kroz Monte Karlo simulacije, kao i vremenska složenost svakog algortima:

| lok. pret. | $O(\cdot)$ | mean | std |
|---|---|---|---|
| random | $O(N^2m)$ | 750.16 | 64.23 |
| greedy | $O(N^4m)$ | 652.24 | 156.83 |
| sim. kalj. | $O(N^2m)$ | 690.93 | 154.72 |
| po snopu | $O(lN^4m)$ | 608.87 | 182.61 |
| genetski | $O(nN^2m)$ | 700.65 | 120.78 |

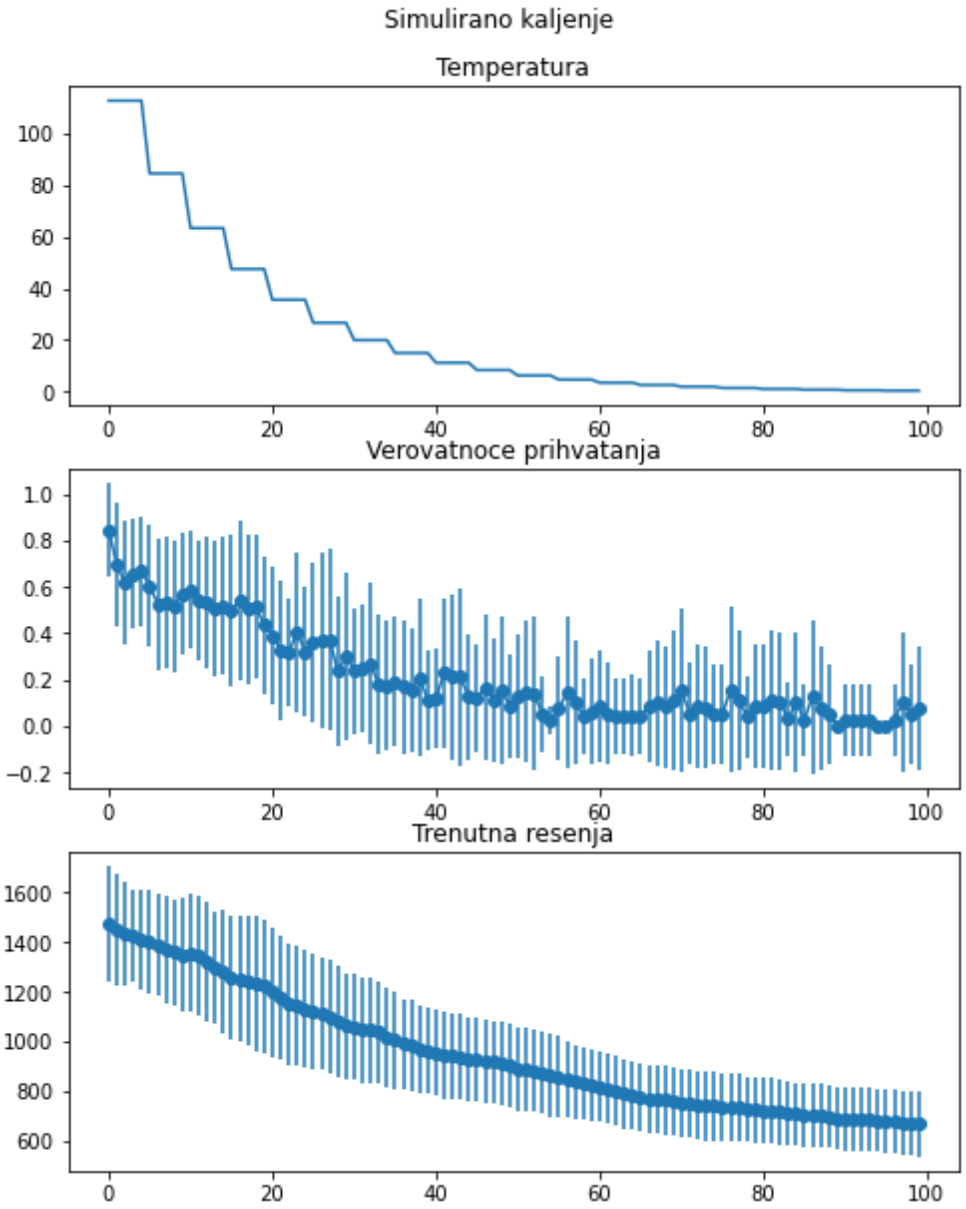
Kod vremenske složenosti, $N$ predstavlja red kvadrata, $m$ predstavlja broj iteracija/generacija u algoritmu, $l$ kod pretrage po snopu predstavlja veličinu snopa, a $n$ kod genetskog algoritma predstavlja veličinu populacije. Zanimljivo je da najgore prosečno rešenje, ali sa najmanjom devijacijom ima nasumična pretraga, dok najbolje prosečno rešenje, ali sa najvećom devijacijom ima pretraga po snopu. Treba napomenuti da je u ovim Monte Karlo simulacijama uzeto $N = 10$, a da je za 100 iteracija svakog algoritam bilo potrebno oko 40 miliona iteracija (tako je nameštено da se poklopi). Takođe, u svim sledećim tačkama izveštaja su korišćene iste vrednosti za $l$, $m$, $N$ i $n$ kao u simulacijama čiji su rezultati dati u tabeli. U slučaju manjeg $N$, mogli bismo povećati broj iteracija kod gramzive i pretrage po snopu kao i veličinu snopa (odnosno, kod gramzive pretrage, broj različitih početnih tačaka), kako bi nam i procene bile konzistentnije - jedino ove dve pretrage imaju problem sa složenosti, jer imaju taj član $O(N^4)$. Zbog ovoga, nema smisla pokretati ova dva algoritma za veće $N$, sam trenutni broj iteracija i veličine snopa (odnosno, kod gramzive pretrage, broj različitih početnih tačaka) je dovoljno mali.

**4**) Za jedan paket realizacije gramzive pretrage dobijeni su sledeći rezultati:



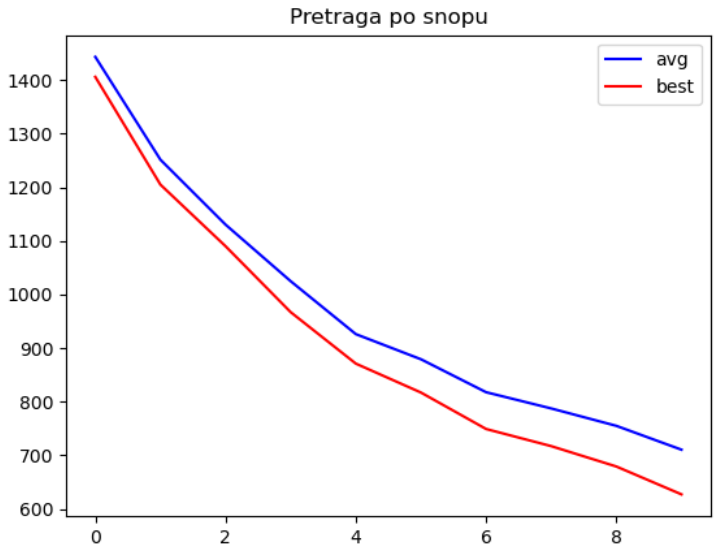

Algoritam je pokrenut $n = 4$ puta (iz 4 različite početne tačke), gde je broj iteracija $m = 10$. Glavni razlog za ovako male vrednosti $n$ i $m$ je sama vremenska složenost algoritma - $O(N^4m)$. Kao što možemo primetiti, prosek, kao i standardna devijacija se smanjuju.

**5)** Kod simuliranog kaljenja, temperatura se menja tako što se u svakoj petoj iteraciji, sama temperatura množi sa faktorom iz skupa $[0, 1]$ (konkretno, u ovim simulacijama 0.75). Startna temperatura je nameštena da bude 150, da bi se red veličine poklopio sa razlikama generisanih i trenutnih najboljih rešenja:

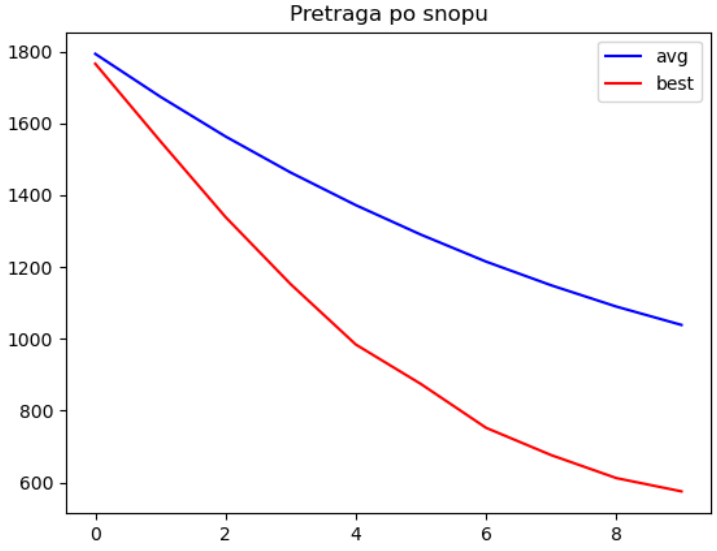



Algoritam je pokrenut $n = 40$ puta, a broj iteracija je bio $m = 100$ (sada zbog vremenske složenosti $O(N^2 m)$ možemo prebaciti resurse na veći broj iteracija i pokretanja). I kod verovatnoća prihavanja kao i kod trenutnih rešenja vidimo da se i prosečna verovatnoća i standardna devijacija smanjuju.

**6**) i **7**) Prosečne vrednosti i minimalne vrednosti ciljnih funkcija trenutne populacije u algoritmu pretraga po snopu su prikazane ovde:




Snop (populacija) je bio veličine $l = 4$, a broj iteracija $m = 10$. Srednja vrednost ciljne funkcije cele populacije u svakoj iteraciji se računala pre nego što je selektovano $l$ rešenja sa najmanjom ciljnom funkcijom. U slučaju da hoćemo da prikažemo srednju vrednost selektovanih $l$ rešenja dobijamo veću razliku između trenutnih najboljih i srednjih vrednosti populacije:



U slučaju genetskog algoritma imamo tri faze:

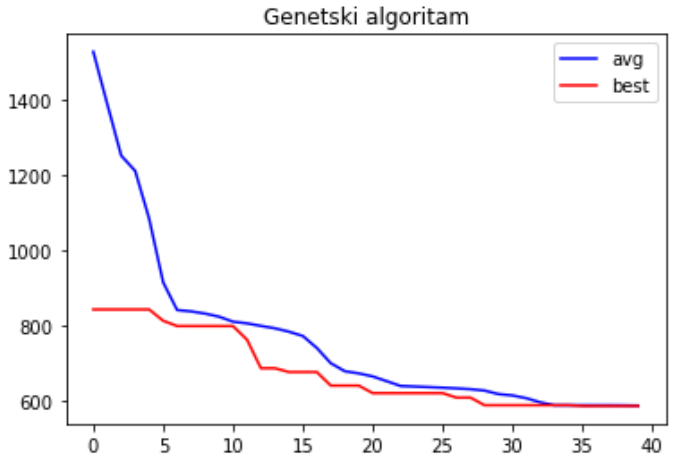
**Selekcija:** Prvo se na početku svake iteracije, u populaciji izabere 10 *elitnih* rešenja. Zatim se biraju roditelji za ukrštanje tako što se iskoristi *Roulette wheel selection*, tj. biramo roditelje na osnovu vrednosti ciljne funkcije - što je manja vrednost ciljne funkcije to je veća verovatnoća da će se to rešenje uključiti u ukrštanje kao roditelj. Treba napomenuti da se ovde može desiti da isti roditelj bude izabran i više puta, dok se neka rešenja uopšte ne biraju. Krajnji deo selekcije podrazumeva deo gde se svi naslednici i roditelji pomešaju i od njih se bira $n - 10$ najboljih iz tog združenog skupa, gde je $n$ veličina početne populacije, a zatim se dodaje 10 elitnih rešenja koja smo na početku iteracije izabrali.

**Ukrštanje:** Ovde nastaje problem ako želimo da "pocepamo" matricu na dva dela a zatim spojimo delove i dobijemo naslednike, jer se može dogoditi da nam se isti broj (element) pojavi dva puta u nekom nasledniku, ili da se ne pojavi uopšte. Ovaj problem se rešava tako što prvo "ispeglamo" roditelje u listu (ima $N^2$ elemenata), a zatim istu listu podelimo na pola, gde prvi deo liste prvog roditelja ide u prvog naslednika, a prvi deo liste drugog roditelja ide u drugog naslednika. Dalje se druga polovina drugog roditelja smešta u prvo dete tako što se do sada nepostojeći elementi automatski ubacuju, a oni koji već postoje preskaču. Konačno, na kraju se nasumično popunjavaju mesta elementima koji fale u listi naslednika. Analogno se proizvodi i drugi naslednik, samo od druge polovine prvog roditelja.Ovaj tip ukrštanja se naziva *Partially mapped crossover*. Na ovom primeru je pokazano kako se dobijaju novi naslednici na matrici 4x4:

$$\begin{bmatrix} 1 & 8 & 7 & 12 \\ 15 & 6 & 3 & 9 \\ 16 & 11 & 4 & 10 \\ 14 & 2 & 5 & 13 \end{bmatrix} + \begin{bmatrix} 10 & 8 & 5 & 6 \\ 15 & 9 & 13 & 11 \\ 3 & 12 & 2 & 14 \\ 4 & 16 & 1 & 7 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 8 & 7 & 12 \\ 15 & 6 & 3 & 9 \\ 5 & 10 & 2 & 14 \\ 4 & 16 & 13 & 11 \end{bmatrix}, \begin{bmatrix} 10 & 8 & 5 & 6 \\ 15 & 9 & 13 & 11 \\ 16 & 1 & 4 & 12 \\ 14 & 2 & 7 & 3 \end{bmatrix}$$

**Mutacija**: Izabere se element na poziciji $i, j$ u matrici i zameni mesta sa elementom na poziciji $j, i$.

Dobijaju se sledeći rezultati za genetski algoritam:



Broj iteracija ovog algoritma je bio $m = 40$, gde je broj jedinki bio 100, a u svakoj iteraciji je izdvajano 10 elitnih (trenutno najboljih) jedinki.