Računalna animacija - 3. laboratorijska vježba

Ana Bagić, 0036515780

January 19, 2023

1 Pokretanje

Nakon kloniranja repozitorija lokalno na uređaj, može se pokrenuti projekt. Zbog korištenja Maven-a i biblioteke OpenGL nisam uspjela pronaći jednostavno rješenje koje bi pokrenulo projekt iz komandne linije zbog više jar datoteka. Međutim, importanjem projekta *lab03* u neko razvojno okruženje (npr. IntelliJ koji sam koristila), jednostavno je pokrenuti aplikaciju. Potrebno je postaviti konfiguraciju, kako se vidi na slici 1 (bitno je da je verzija Jave 11).

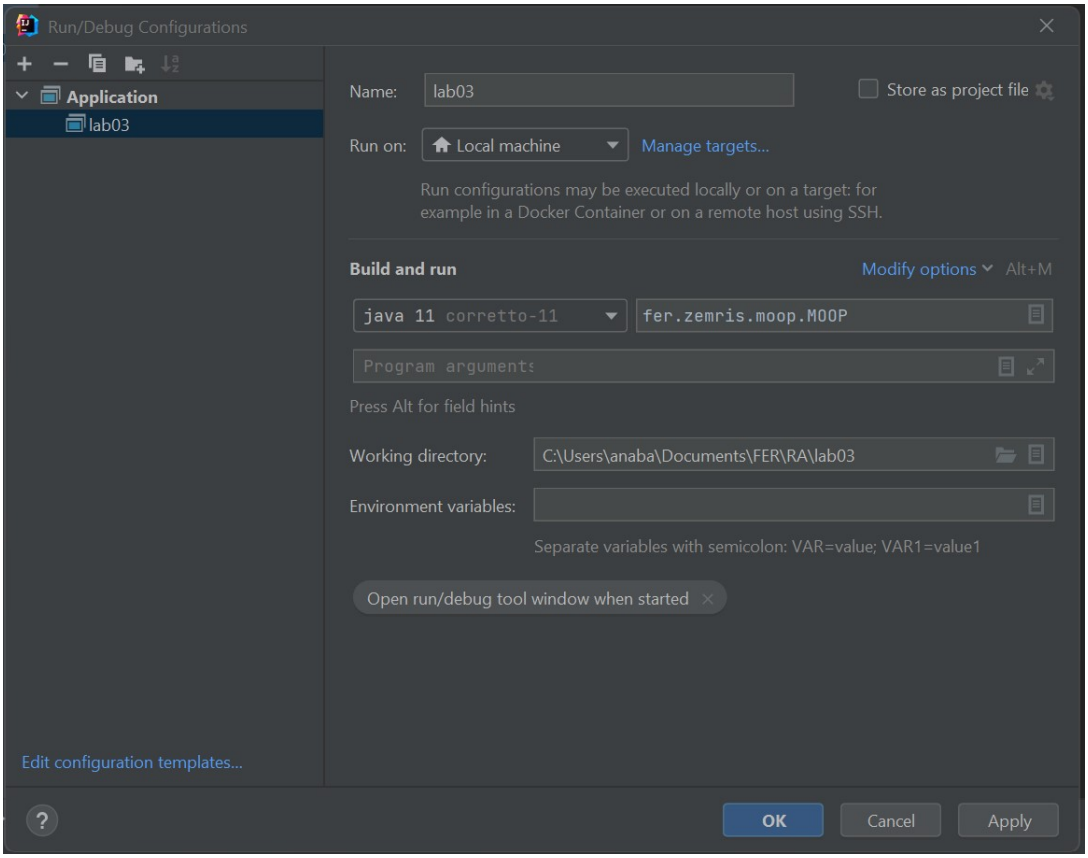

Figure 1: Konfiguracija za pokretanje

2 Opis rada

U ovom radu implementirala sam grafički prikaz algoritma višekriterijske optimizacije NSGA-II u Javi koristeći OpenGL (JOGL) biblioteku. Točnije, radi se o vizualizaciji samo dvokriterijske optimizacije zbog ograničenja s prikazom više dimenzija.

2.1 NSGA-II algoritam

NSGA-II (*Non-dominated Sorting Genetic Algorithm II*) algoritam je jedan od najpoznatijih algoritama trenutno korištenih za višekriterijsku optimizaciju. Nastao je kao poboljšanje algoritma NSGA, tako da uvodi elitizam, koristi nedominirano sortiranje i mjeru *crowding distance*. Glavna ideja algoritma je sortirati populaciju rješenja u različite fronte, po tome koliko su rješenja "dobra", odnosno po kriteriju nedominacije.

Svojstvo nedominacije znači da neko rješenje barem po nekom kriteriju bolje od ostalih koja nisu u toj fronti, odnosno da nije dominirano ostalim rješenjima. Po tom svojstvu rješenja se slažu u fronte, gdje je prva fronta ona koja sadrži rješenja s najvećom dobrotom, druga ona malo lošija i tako dalje. Pseudokod algoritma može se vidjeti na slici 2 preuzete iz prezentacije o višekriterijskoj optimizaciji iz kolegija *Optimiranje evolucijskim računanjem*.

NSGA-II

```
Inicijaliziraj populaciju P
dok(kriterij zaustavljanja nije zadovoljen)
      Q = stvori novu populaciju provođenjem operatora nad P
      R= Q ∪ P
      F = Nedominirano sortiraj R
      C - nova populacija
      i = 0
      dok (|C|+|Fi|<|P|)
            izračunaj crowding distance za Fi
            C = C ∪ Fi
            i++
      Izračunaj crowding distance za Fi
      Sortiraj Fi po crowded distanceu
      C = C ∪ Fi[1:N-|C|]
      P = C
```

Figure 2: Pseudokod algoritma NSGA-II

Crowding distance, odnosno "udaljenost gužve" govori nam koliko su blizu susjedna rješenja iz te fronte. Korištenjem te mjere, daje se prednost rješenjima koja sadrže manje ostalih rješenja iz iste fronte u blizini. Time se održava raznolikost rješenja.

2.2 Vizualizacija

U ovom primjeru koristila sam sljedeći problem. Minimiziraj funkcije:

$$f(x, y) = x \tag{1}$$

$$g(x, y) = \frac{1 + y}{x} \tag{2}$$

uz ograničenja:

$$0.1 <= x <= 1 \tag{3}$$

$$0 <= y <= 5 \tag{4}$$

Odmah prilikom pokretanja aplikacije, prozor izgleda kao na slici 3. Algoritam se može pokrenuti pritiskom na tipku P (ta tipka također služi za pauziranje izvođenja algoritma, kako bi se vidjela populacija u određenom trenutku). Također, odgoda od dvije sekunde postavljena je u svaku iteraciju algoritma, kako bi se u stvarnom vremenu mogao promatrati napredak. X os prikazuje iznos funkcije *f(x, y)*, a y os funkcije *g(x, y)*.

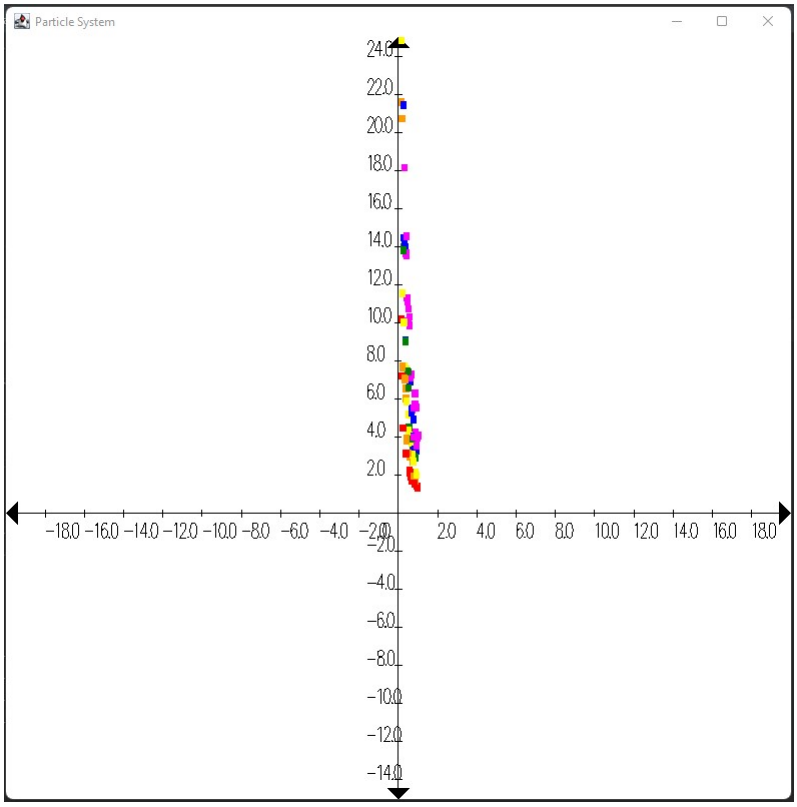

Figure 3: Nakon pokretanja algoritma

Kao što se vidi, koordinatni sustav trenutno nije dobro pozicioniran, te su rješenja zbog malog iznosa na x osi, koncentrirana na jedan dio prozora. Taj problem riješila sam mogućnošću translacije i skaliranja koordinatnih osi. Tipkama *strelica gore* i *strelica dolje* pomiče se pogled na koordinatni sustav po y osi, a tipkama *strelica lijevo* i *strelica desno* po x osi. Zatim, tipkama W i S povećava se odnosno smanjuje skala y osi, a tipkama A i D skala x osi. Nakon dobre translacije i skaliranja koordinatnog sustava, te nakon par iteracija algoritma, prikaz izgleda kao što prikazuje slika 4.

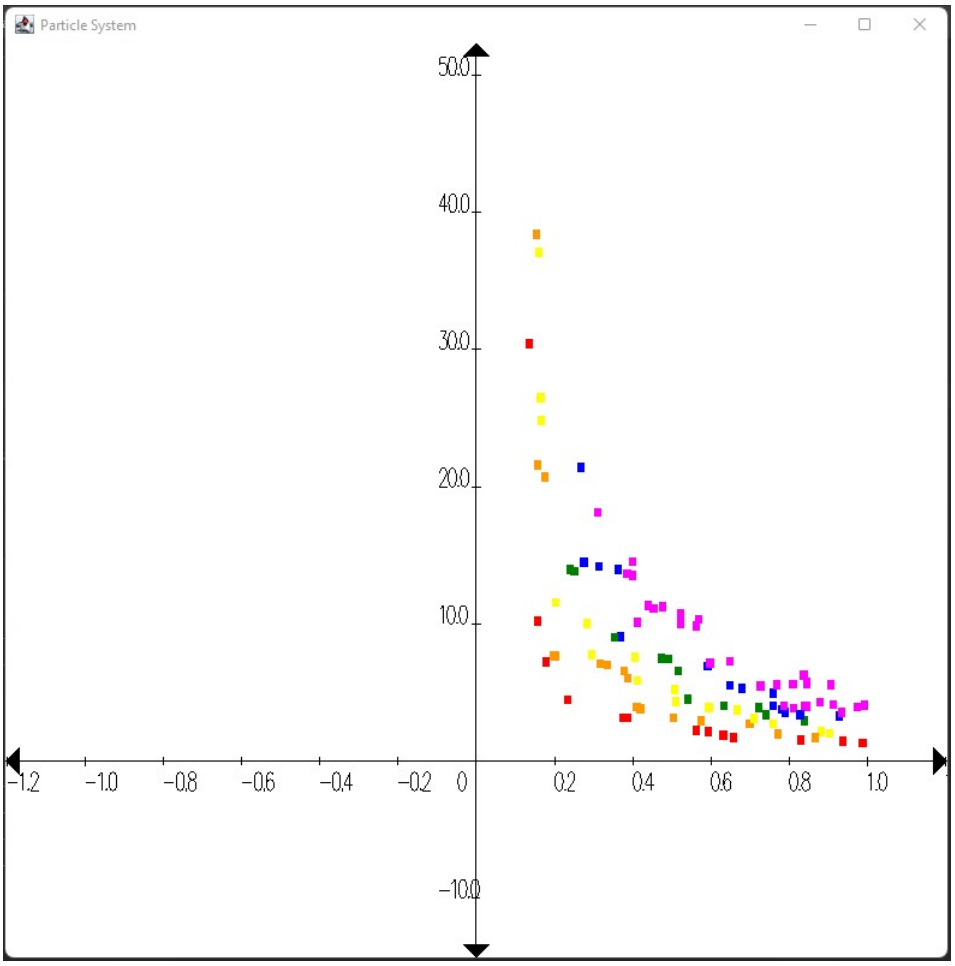

Figure 4: Nakon nekoliko iteracija algoritma

Može se vidjeti kako su različite fronte prikazane različitim bojama. Najbolja fronta je crvene boje. Zatim fronte idu redoslijedom: narančasta, žuta, zelena, plava, te su sve fronte nakon toga ljubičaste. Par desetaka iteracija kasnije, rješenja se stabiliziraju što se može vidjeti na slici 5. Sva rješenja su u najboljoj fronti (crvena su), što znači da za svako rješenje ne postoji nijedno drugo koje ga dominira po svim kriterijima.

Programsko rješenje implementirano je u programskom jeziku Java verzije 11 i koristeći biblioteku OpenGL (odnosno JOGL - Java OpenGL).

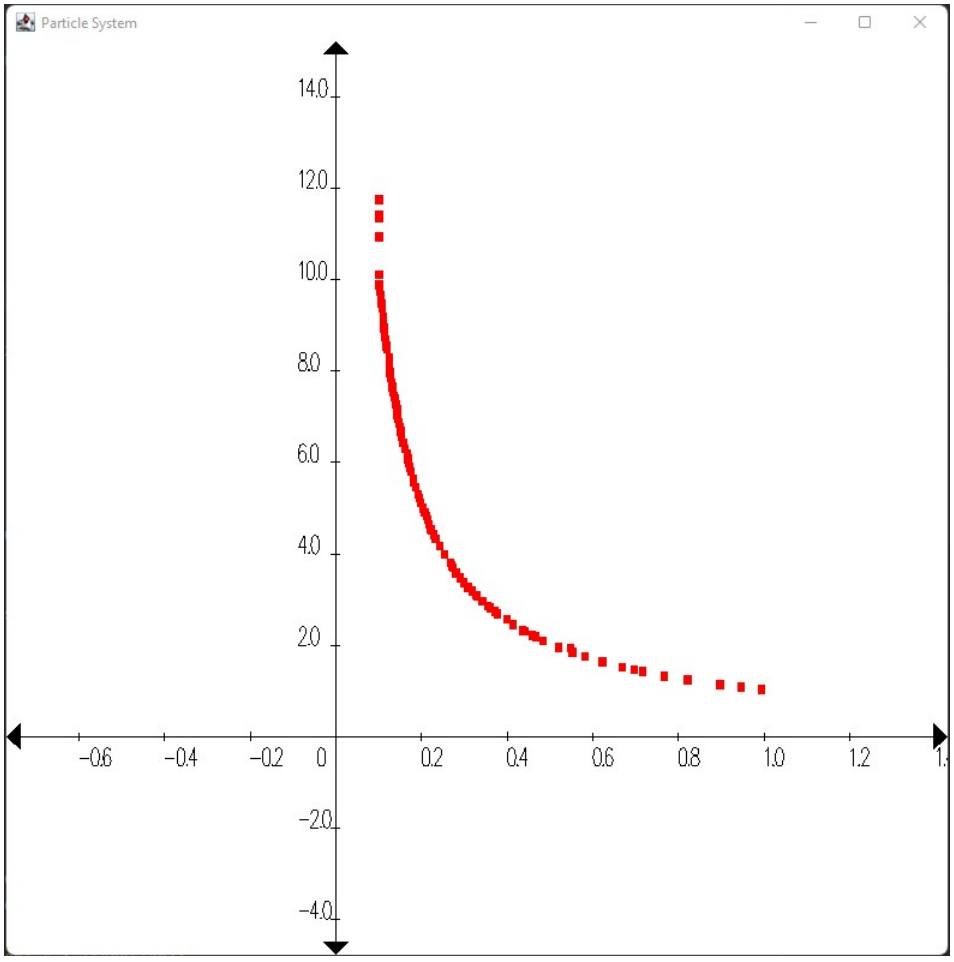

Figure 5: Stabilizacija rješenja