Autor: hrckov

Napredni algoritmi i strukture podataka

2. laboratorijska vježba
Genetski Algoritam

Zadatak:

Napisati program (poželjno C, C++, C# ili Java) koji će genetskim algoritmom riješiti proizvoljan problem.

- cilj je da student ovlada evolucijskom strategijom kao takvom pa problem sam po sebi može biti i vrlo jednostavan (npr. maksimizacija funkcije kao na predavanjima ili sl.), glavno da se vidi uspješnost usvajanja obrađenog gradiva
- za kolokviranje vježbe važno je pregledno i jasno opisati model problema i ideju rješenja, dakle građu kromosoma, stvaranje polazne populacije, načine križanja i mutacija, kriterij vrednovanja jedinki, stvaranje nove generacije, kontrolu konvergencije algoritma itd.

Uvod:

Genetski algoritam će rješavati problem maksimizacije funkcije. Kao primjer je odabrana funkcija iz predavanja:

$$f(x, y) = 0.5 - \frac{\sin^2\left(\left(x^2 + y^2\right)^{\frac{1}{2}}\right) - 0.5}{\left(1 + 0.001\left(x^2 + y^2\right)\right)^2}$$

Algoritam je lako prilagoditi za minimiziranje funkcija, kao i za druge funkcije.

Veličina generacije, broj generacija, faktor mutacije, vjerojatnosti križanja i mutacije, te interval unutar kojeg se generiraju varijable su varijabilni, te o njima ovisi preciznost rezultata.

Opis problema:

Zadatak se može razlučiti na X manjih, međusobno zavisnih, problema:

1) Struktura kromosoma
2) Inicijalizacija početne generacije
3) Odabir jedinki za križanje
4) Križanje jedinki
5) Mutacija jedinke
6) Zamjena generacija
7) Optimizacija algoritma
8) Korisničko sučelje

1. **Struktura kromosoma**

Kromosom će se sastojati od varijabli X i Y koje predstavljaju točke koje maksimiziramo algoritmom. X i Y će biti binarno reprezentirani. Kako bi implementacija bila jednostavnija i efikasnija, kromosom će sadržavati i vrijednost koju funkcija poprima za zadane točke X i Y.

2. **Inicijalizacija početne generacije**

Inicijalizacija početne generacije biti će slučajna binarna inicijalizacija nad varijablama X i Y. Algoritam zabranjuje stvaranje duplikata kromosoma, pa se neke jedinke odbacuju.

3. **Odabir jedinki za križanje**

Odabir jedinki za križanje izvoditi će se linearnom normalizacijom dobrote od minimuma prema maksimumu. Dobrota jedinke odgovara vrijednosti koju funkcija poprima za zadane točke X i Y.

4. **Križanje jedinki**

Križanje se izvodi jednom prekidnom točkom koja je nasumično odabrana za svaku varijablu X i Y. Svakim križanjem nastaju dvije nove jedinke. Algoritam zabranjuje stvaranje duplikata kromosoma, pa se neke jedinke odbacuju.

5. **Mutacija jedinke**

Mutacija se provodi nad svakim novim kromosomom sa zadanom vjerojatnosti. Mutirani bit uvijek mijenja vrijednost. Sa svakom sljedećom generacijom vjerojatnost mutacije linearno raste ovisno o zadanom faktoru mutacije.

6. **Zamjena generacija**

Prilikom zamjene generacija primjenjuje se strategija elitizma. Najbolja jedinka prošle generacije prenosi se u sljedeću generaciju, a sve ostale jedinke se brišu i zamjenjuju novim jedinkama dobivenim križanjem.

7. **Optimizacija algoritma**

Od optimizacijskih tehnika algoritam koristi elitizam – prijenos najbolje jedinke pojedine generacije u iduću generaciju, linearno povećavanje faktora mutacije sa generacijama, odbacivanje duplikata jedinki. Također, algoritam ima dva načina rada: algoritam radi dok ne završi evoluciju kroz sve generacije ili ako je postavljen vremenski rok, dok ne istekne vrijeme.

8. **Korisničko sučelje**

Kako bi korisnicima olakšali korištenje programa izgrađeno je korisničko sučelje. Sučelje omogućava korisniku postavljanje varijabli algoritma: vrijeme izvođenja, broj generacija, broj jedinki, vjerojatnost mutacije, faktor mutacije, preciznost i interval inicijalne populacije. U grafičkom se sučelju prikazuju jedinke kroz generacije – vrijednost točki X i Y koje maksimiziramo, te dobrota pojedine jedinke. Po završetku izvođenja u sučelju se ispisuje dobiveni rezultat – točke X i Y, te dobrota najbolje jedinke.

Zaključak

Genetski algoritam je implementiran koristeći što više osnovnih metoda uz nekoliko optimizacijskih metoda. Rezultati algoritma ovise o početnoj generaciji, intervalu i preciznosti varijabli. Zbog odbacivanja duplikata program za uči u veoma dugu petlju križanja i traženja ne-duplikata, te se preporuča postaviti ograničenje na trajanje izvođenje iz tih razloga. Kako bi pratili brzinu izvođenja algoritma dodano je praćenje i ispisivanje trajanja izvođenaj algoritma.