4. domaća zadaća – genetski algoritam

Zadatak 1.

Napišite razred `hr.fer.zemris.optjava.dz4.part1.GeneticAlgorithm` koji rješava četvrti zadatak (traženje koeficijenata *a* do *f* iz druge domaće zadaće). Algoritam treba biti *generacijski elitistički*. Neka kao reprezentaciju rješenja koristi polje decimalnih brojeva. Operator križanja izvedite kao BLX-α a operator mutacije prema pseudokodu 3.2 (vidi poglavlje 3.1.2 u knjizi). Algoritam kao parametre komandne linije treba primiti:

- veličinu populacije,
- ciljanu minimalnu vrijednost pogreške uz koju će algoritam prestati s radom,
- maksimalni broj generacija koje algoritam smije napraviti (može stati i prije ako pronađe dovoljno dobro rješenje – vidi prethodni uvjet),
- vrstu selekcije koja se koristi te
- parametar *sigma* koji je potreban za operator mutacije.

Program treba podržati sljedeće vrste selekcije:
1) `rouletteWheel` te
2) `tournament:n`, gdje je *n* broj veći ili jednaki 2.

U slučaju da se traži proporcionalna selekcija (*rouletteWheel*) i s obzirom da se radi minimizacija funkcije, u populaciji pronađite dobrotu najgore jedinke (f_{worst}); potom *i*-toj jedinki dodijelite "efektivnu" dobrotu prema formuli f_i-f_{worst}, gdje je f_i prethodno direktno utvrđena dobrota te jedinke (primjerice kao `-error(i)`) . Ovo će osigurati da su sve dobrote s kojima radite pozitivne (što je nužno za proporcionalnu selekciju) te da je postupak otporan na *problem skale* (vidi knjigu).

U slučaju da korisnik traži turnirsku selekciju, parsirajte zadani string koji mora biti oblika `tournament:n` i izvadite *n*. Parametar *n* određuje veličinu turnira koji se radi za izbor svakog roditelja. Primjerice, ako je *n*=4, iz populacije ćete izvući 4 jedinke i najbolju uzeti za roditelja. Trebate li 2 roditelja, postupak ćete ponoviti dva puta.

Program na zaslon treba ispisati nakon svake generacije najbolju jedinku i njezinu dobrotu.

Pokušajte utvrditi kada algoritam radi bolje:

- za veće ili manje populacije,
- za proporcionalnu ili turnirske selekcije,
- u slučaju turnirskih selekcija, za veći ili manji *n* te
- za veći ili manji *sigma*?

Dajte jedan skup parametara uz koje algoritam radi dobro.

Zadatak 2.

Napišite razred `hr.fer.zemris.optjava.dz4.part2.BoxFilling` koji koristi *steady-state* genetski algoritam i rješava zadatak A.3 iz knjige (zadatci se nalaze na kraju knjige). Kao argumente, program mora primiti sljedeće:

- stazu do datoteke s definicijom problema,
- veličinu populacije,
- *n* (pri čemu n≥2),
- *m* (pri čemu m≥2),
- *p* (pri čemu je p=true ili p=false),
- maksimalni broj iteracija koji algoritam smije napraviti te
- prihvatljiva duljina spremnika uz koju algoritam smije stati i prije dosezanja maksimalnog broja iteracija.

Ako Vaš algoritam ima još koji parametar (primjerice, koji kontrolira mutaciju), dodajte ga također kao argument komandne linije.

U repozitorij sam stavio nekoliko datoteka s primjerima. Ime svake datoteke je sljedećeg oblika: `problem-h-s-n.dat`, gdje je *h* visina spremnika (u svim primjerima koje sam dao *h*=20) a *s* je minimalna duljina spremnika u koji se sve sigurno može zapakirati (dao sam primjere sa *s*=10, *s*=30 te *s*=50); ova informacija može Vam poslužiti da provjerite koliko dobro algoritam radi. **Sam algoritam ni na koji način ne smije dobiti i/ili koristiti ovu informaciju.** *n* je redni broj primjera.

Kod *steady-state* algoritma jedna iteracija odgovara ujedno i jednoj generaciji. Algoritam treba koristiti *n*-turnir za odabir roditelja (ako treba dva roditelja, radi dva takva *n*-turnira), roditelje treba križati pa dijete mutirati. Kako bi odlučio na koje će mjesto umetnuti nastalo dijete, algoritam treba napraviti *m*-turnir pri čemu je pobjednik najgora od *m* izvučenih jedinki. Ako je parametar *p* postavljen na `false`, dijete bezuvjetno zamjenjuje jedinku pobjednika *m*-turnira. Ako je parametar *p* postavljen na `true`, dijete će pobjednika *m*-turnira zamijeniti samo ako ima bolju dobrotu od pobjednika; u suprotnom, pobjednik ostaje u populaciji a dijete se eliminira.

Program na kraju rada treba na zaslon ispisati najbolje pronađeno rješenje. Tijekom rada, program treba ispisati dobrotu rješenje svaki puta kada pronađe novo najbolje rješenje (u tim međukoracima ne treba ispisivati i samo rješenje).

Napomene:

Rok za predaju Eclipse projekta je sljedeći četvrtak do termina predavanja (14:00).