4. DOMAĆA ZADAĆA – AK. GOD. 2018/19

Domaća zadaća

U okviru ove zadaće treba izraditi programsku implementaciju genetskog (evolucijskog) algoritma koja će pronalaziti minimum zadane višedimenzijske funkcije. Potrebno je omogućiti da se bez prevođenja programa može definirati:

- veličina populacije,
- prikaz rješenja koji će algoritam koristiti,
- vjerojatnost mutacije (i križanja, ako algoritam koristi taj parametar),
- broj evaluacija funkcije cilja kao kriterij zaustavljanja (ostali su proizvoljni).

Programsko ostvarenje mora podržavati *binarni prikaz rješenja* te *prikaz brojem s pomičnom točkom* (način prikaza je parametar algoritma). Oba prikaza trebaju definirati donju i gornju granicu domene (eksplicitna ograničenja) i broj varijabli. Binarni prikaz dodatno treba definirati željenu preciznost (npr. kao broj decimala ili najveći dopušteni razmak susjednih rješenja). Za svaki od prikaza potrebno je definirati barem po *dva operatora križanja* i barem po *jedan operator mutacije*. Svi operatori moraju podržavati definirana eksplicitna ograničenja.

Ispis algoritma je proizvoljan, no mora omogućiti uvid u ove podatke: broj evaluacija, trenutna najbolja jedinka, vrijednost funkcije cilja najbolje jedinke. Preporuča se ispis svakih *n* evaluacija ili prilikom promjene najbolje jedinke.

Preporučena (ali nipošto i jedina moguća) inačica GA je 3-turnirski eliminacijski odabir:

```
stvori populaciju;
evaluiraj populaciju;   // svaka jedinka ima pridruženu dobrotu
ponavljaj
      odaberi slucajno 3 jedinke;
      izbaci najlosiju od 3 odabrane iz populacije;
      nova_jedinka = krizanje(preostale dvije);
      obavi mutaciju na novoj jedinki uz vjerojatnost p_M;
      evaluiraj novu jedinku;
      dodaj novu jedinku u populaciju;
dok (nije zadovoljen uvjet zaustavljanja);
```

Funkcije cilja

- $f_1(\mathbf{x}) = 100 \cdot \left(x_2 - x_1^2\right)^2 + \left(1 - x_1\right)^2$ (*Rosenbrockova 'banana' funkcija*)

  početna točka: $\mathbf{x}_0 = (-1.9, 2)$, minimum: $\mathbf{x}_{\min} = (1,1)$, $f_{\min} = 0$

- $f_3(\mathbf{x}) = \sum_i \left(x_i - i\right)^2$

  početna točka: $\mathbf{x}_0 = \mathbf{0}$ (nul vektor), minimum: $\mathbf{x}_{\min} = (1, 2, 3, ..., n)$, $f_{\min} = 0$

- $f_6 = 0.5 + \dfrac{\sin^2 \sqrt{\sum x_i^2} - 0.5}{\left(1 + 0.001 \cdot \sum x_i^2\right)^2}$ (*Schaffer's function f6*)

  globalni minimum: $\mathbf{x}_{\min} = \mathbf{0}$ (nul vektor), $f_{\min} = 0$

- $f_7 = \left(\sum x_i^2\right)^{0.25} \cdot \left(1 + \sin^2\left(50\left(\sum x_i^2\right)^{0.1}\right)\right)$ (skoro pa *Schaffer's function f7*)

  globalni minimum: $\mathbf{x}_{\min} = \mathbf{0}$ (nul vektor), $f_{\min} = 0$

Slike funkcija 6 i 7 u jednoj i dvije dimenzije mogu se naći ovdje:
(http://www.fer.unizg.hr/_download/repository/lab_3_slike.pdf).

Laboratorijska vježba

1. Isprobajte vašu implementaciju GA nad svim funkcijama uz granice [-50, 150] za sve varijable *te za oba prikaza rješenja* (binarni, s pomičnom točkom). Za binarni prikaz zadajte preciznost od barem tri decimalna mjesta. Za funkciju f3 odaberite barem 5 varijabli, a za f6 i f7 dvije varijable. Za sve funkcije možete smatrati da je rješenje pronađeno ako je krajnja vrijednost funkcije cilja manja od 10^{-6}. Za neke funkcije algoritam će biti potrebno pokrenuti nekoliko puta. Koje zaključke možete donijeti o uspješnosti GA za pojedinu funkciju? Koje su se funkcije pokazale teškima i zašto?
2. Provedite GA na funkcijama f6 i f7 mijenjajući dimenzionalnost funkcije (1, 3, 6, 10). Kako povećanje dimenzionalnosti funkcije utječe na ponašanje algoritma?
3. Za funkcije f6 i f7 usporedite učinkovitost GA koji koristi binarni prikaz uz preciznost na 4 decimale (tj. 10^{-4}) i GA koji koristi prikaz s pomičnom točkom (ostali parametri neka budu jednaki), za dimenzije 3 i 6. Rad algoritma ograničite zadanim brojem evaluacija (oko 10^5). Inačice algoritma usporedite po uputama u sljedećem odjeljku. Što možete zaključiti o različitim prikazima rješenja za različite funkcije?
4. Za funkciju f6 pokušajte pronaći 'idealne' parametre genetskog algoritma. 'Idealne' parametre potrebno je odrediti barem za veličinu populacije (npr. 30, 50, 100, 200) i vjerojatnost mutacije jedinke (npr. 0.1, 0.3, 0.6, 0.9) a po želji možete i za još neke druge parametre koje je vaš algoritam koristio. Jedan postupak traženja parametara opisan je u nastavku. Koristite medijan kao mjeru usporedbe i prikažite kretanje učinkovitosti za barem jedan parametar uz pomoć *box-plot* prikaza (opisano u nastavku).
5. Ako ste implementirali turnirsku selekciju, probajte nad nekom težom funkcijom (f6 ili f7) izvesti algoritam koristeći različite veličine turnira. Pomaže li veći turnir algoritmu da pronađe bolja rješenja? Ako ste implementirali selekciju *roulette wheel*, isprobajte više vrijednosti omjera odabira najbolje i najlošije jedinke (skaliranje funkcije cilja) te komentirajte dobivene rezultate (također možete isprobati generacijsku i eliminacijsku varijantu).

Usporedba algoritama i prikaz rezultata

Često je pitanje kako usporediti učinkovitost više algoritama ili istog algoritma za različite vrijednosti nekog parametra. Prije svega potrebno je uočiti da samo jedno izvođenje stohastičkog algoritma nije dovoljno; za svaku inačicu algoritam je potrebno izvesti nekoliko puta (barem 10 u ovoj vježbi, 30-50 za statistički potkrijepljenu nirvanu) uz *jednaki broj evaluacija* i iz svakog izvođenja zabilježiti dobrotu najbolje jedinke. Tada se nad tim skupom dobrota mogu izračunati različite mjere (srednja vrijednost, medijan, broj 'pogodaka' i slično), koje se koriste za usporedbu. Prilikom ispitivanja u ovoj vježbi iskoristite **broj pogodaka** (vrijednost funkcije cilja ispod 10^{-6}) kao primarnu te medijan kao sekundarnu mjeru (ako je broj pogodaka jednak). Probajte obrazložiti zašto je medijan dobra mjera (često bolja od srednje vrijednosti).

Konačno, rezultate je korisno i prikazati. Rezultati se često prikazuju u obliku *box-plot* dijagrama [5]. Prednost ovih dijagrama jest u tome što je pomoću njih podatke vrlo lako prikazati njihovim kvartilnim vrijednostima. Nakon obavljenih ispitivanja, rezultate možete prikazati u zasebnom *box-plot* prikazu za parametar koji se optimira. Zgodan i jednostavan *online* alat za izradu *box-plot* prikaza dostupan je na [6].

Kako pronaći dobre parametre

Najčešći postupak optimiranja parametara je jedan po jedan, počevši od parametara veće 'važnosti' (utjecaja na učinkovitost). Primjerice, ako optimiramo veličinu populacije, odabiremo nekoliko vrijednosti za taj parametar i pri tome sve ostale parametre držimo nepromijenjene. Kada smo našli 'idealnu' vrijednost za taj parametar, njega fiksiramo i potom optimiramo idući parametar. Postupak se nastavlja dok to nismo

napravili za sve parametre. Na početku je potrebno definirati početni skup vrijednosti (proizvoljno, po osjećaju).

Pronalaženje parametara moguće je i uporabom optimizacijskog algoritma koji kao funkciju cilja dobiva učinkovitost promatranog algoritma na 'nižoj razini' (GA u ovom slučaju), pa je tu riječ o *meta-optimizaciji*. Algoritam na 'višoj razini' može biti deterministički ili stohastički, pa su moguće kombinacije lokalni-GA, ili GA-GA ako se jedan GA koristi za optimizaciju parametara 'nižeg' GA.

Reference:

[1] http://www.cs.unm.edu/~neal.holts/dga/benchmarkFunction
[2] http://zhanggw.wordpress.com/2010/09/25/optimization-schaffer-f6-function-using-basic-genetic-algorithm-2/
[3] http://www.geatbx.com/docu/fcnindex-01.html
[4] http://www.gamsworld.org/performance/selconglobal
[5] http://en.wikipedia.org/wiki/Box_plot
[6] http://shiny.chemgrid.org/boxplotr/
[7] http://app.rawgraphs.io/