Genetski algoritem na problemu potujočega trgovca

Tina Bertok
Neža Habjan
Gašper Letnar

14. december 2018

1 Uvod

Naša naloga je, da implementiramo genetski algoritem na problem potujočega trgovca. Pri tem bomo uporabljali različna križanja, velikosti populacije in verjetnostmi za mutacijo. Opazovali bomo, kako se spreminja rezultat glede na spreminjanje spremenljivk in pogojev. Prav tako bomo nekaj testnih grafov generirali sami, ter jih nato primerjali s tistimi, najdenimi na internetu.

Genetski algoritem je metahevristika, navdihnjena s strani procesov naravne selekcije in spada v razred razvojnih algoritmov. Uporablja se za generiranje kvalitetnih rešitev v optimizaciji, ki temeljijo na operatorjih kot so mutacija, križanje in selekcija.
V genetskem algoritmu se uporabi množica kandidatov za rešitev, ki jih nato razvijamo do optimalne rešitve. Vsak kandidat ima določene lastnosti, katere lahko spremenimo oziroma lahko mutirajo. Evolucija rešitev se ponavadi začne na naključni izbiri kandidatov, katere potem s pomočjo iteracije razvijamo. Na vsakem iterativnem koraku se potem oceni primernost novih kandidatov za optimizacijski problem. Najboljše kandidate potem uporabimo za naslednji korak iteracije. Na koncu se algoritem zaključi, ko doseže maksimalno število iteracijskih korakov oziroma, ko dobi najboljši približek optimalni rešitvi.

Problem trgovskega potnika oz. *traveling salesman problem (TSP)* je NP-težek problem v kombinatorični optimizaciji pomemben pri operacijskih raziskavah, matematični optimizaciji in teoretičnemu računalništvu. Trgovski potnik mora obiskati določeno množico mest tako, da bo pri tem prehodil čim krajšo pot in se vrniti v izhodišče.

2 Potek in opis dela

Že dobro poznan problem potujočega , smo se odločili predstaviti z grafom v obliki *nxn* matrike cen povezav. Graf smo generirali tako, da smo elemente matrike, ki predstavljajo celoštevilske cene povezav izbrali naključno.
Nato je bilo potrebno definirati *ciljno funkcijo* oz. *fitness function*, ki bo v naštem primeru predstavljala dolžino najcenejše poti v grafu.

Naključno smo ustvarili začetno popoulacijo določene velikosti. Vsak element populacije ali *kromosom* predstavloja neko pot, ki obišče vsa vozlišča grafa.

V nadaljevanju izberemo starše, ki jih damo v paritveni bazen. Za selekcijo imamo dve možnosti, in sicer selekcijo s turnirjem ter proporcionalno selekcijo. Odločili smo se, da bomo bazen gradili turnirsko. To pomeni, da bomo iz populacije naključno izbrali k kromosomov oz. poti. Naša funkcija oz. *selekcija* nam bo vrnila zmagovalca kot pot z najkrajšo dolžino in pripadajočo dolžino. Torej bomo za *n* staršev v bazenu imeli *n* turnirjev. Število kromosomov v turnirju *k* izberemo sami, vendar moramo biti pri tem predvidni. Večji kot je *k*, hitrejša bo konvergenca, kar pa ni nujno dobro. Najboljše starše (tiste, ki so največkrat zmagali) bomo dali v paritveni bazen.

Za ustvarjanje otrok bomo uporabili različna križanja oz. *crossoverje* staršev (poti) iz paritvenega bazena.

Različne variacije križanj:

- urejeno križanje oz. *ordered crossover*
- delno mapirano križanje oz. *partially mapped crossover*
- ciklično križanje oz. *cycle crossover*

Tako bomo dobili otroke, ki so že izboljšani primeri poti. Da pa ohranjamo diverziteto v populaciji, si moramo med temi otroci izbrati določen procent, ki ga bomo mutirali z *SWAP mutacijo* (zamenjali bomo dve vozlišči v poti). Vsako vozlišče z neko verjetnostjo mutiramo, torej zamenjamo položaj mutiranega vozlišča z nekim naključnim vozliščem na poti.

3 Načrt za nadaljnje delo

V nadaljevanju imamo namen naš program preizkusiti na različnih velikostih populacije in primerjati rezultate. Prav tako bomo opazovali, kaj se spremeni, če uporabimo drugačno različico križanja. Postavili si bomo tudi vprašanje, kakšen bo rezultat, če povečamo verjetnost mutacije.