# FIIT STU

Ilkovičova 2, 842 16 Karlova Ves

Zadanie 3

Hľadanie pokladu

Autor: Martin Pirkovský

2020/2021

1. Riešený problém – zadanie

Mojou úlohou v tomto zadaní bolo pozbierať všetky poklady na mape za pomoci evolučného programovania nad virtuálnym strojom. Na mape máme jedného hľadača, ktorý sa môže pohybovať H - hore, D – dole, P – doprava, L – doľava. Hľadač má svoju sadu inštrukcií, ktoré vložíme do virtuálneho stroja. Tie stroj vyhodnotí a vráti nám cestu, ktorou sa hľadač vybral. Na základe nej vieme vyhodnotiť koľko pokladov hľadač pozbieral a ako je jeho cesta efektívna.

Virtuálny stroj má 64 pamäťových buniek o veľkosti 1 byte. Prvé 2 bity nám hovoria o akú inštrukciu sa jedná. „00“ je inštrukcia inkrementácie, „01“ je dekrementácia, „10“ je skoková inštrukcia a „11“ je výpis, respektíve hovorí nám o tom, ktorým smerom sa hľadač posunul. Zvyšných 6 bitov predstavuje adresu, s ktorou je potrebné pracovať. Hľadač končí svoje hľadanie ak našiel všetky poklady, vystúpil z mriežky alebo bolo vykonaných 500 inštrukcií.

1. Opis použitého algoritmu

Ako je v zadaní spomenuté použil som evolučné programovanie, konkrétne lineárne evolučne programovanie, kde je jednotlivec reprezentovaný ako postupnosť inštrukcií. Tieto inštrukcie sú nakopírované do virtuálneho stroja, v ktorom ich vyhodnotím. Inštrukcie musia byť nakopírované, pretože ak by sme v stroji pracovali s pôvodnou postupnosťou, a menili ju priamo jedincovi, tak by to malo za následok, že by sme túto pozmenenú pamäť aj posielali do kríženia a nakoľko by sa stratila pôvodná postupnosť, graf zobrazujúci postupnosť ako sa nám fitness vyvíja s generáciami by nám nekonvergoval. Jedinec je reprezentovaný fitness hodnotou, cestou, ktorú prešiel, sadou inštrukcii, a poľom s pozbieranými pokladmi.

Gén tvorí jedna bunka pamäte inštrukcií jedinca, teda 8 bitové číslo, ktoré som vytvoril za pomoci knižnice numpy. V ňom nám prvé 2 bity hovoria o akú inštrukcia ide (inkrementácia, dekrementácia, skok a výpis), a zvyšných 6 bitov nám hovorí s akou adresou máme pracovať. Chromozómom algoritmu je vlastne jeden jedinec, a teda postupnosť jeho inštrukcií.

Evolučný algoritmus som implementoval nasledovne. Ako je napísané na stránke zadania, tak ako prvé načítam zo súboru rozmer mapy, štartovaciu pozíciu hľadača, počet a rozloženie pokladov. Následne vytvorím počiatočnú populáciu jedincov, pričom jedincovi nastavím prvých 32 buniek na náhodné hodnoty. K počtu koľko buniek je vhodné naplniť náhodnými hodnotami som sa dostal experimentovaním, kedy som pri hodnotách okolo polovice dostával približne najlepšie výsledky a tak som zvolil presnú polovicu 32 buniek.

Ďalej pokračujem poslaním každého jedinca z generácie do virtuálneho stroja, kde sa vykoná jeho sada inštrukcií a tým sa zistí jeho cesta, ktorou išiel. Nakoľko inštrukcie inkrementácie, dekrementácie a skoku sú jasné zo zadania ako fungujú, vysvetlím iba ako som implementoval inštrukciu zápisu, ktorá bola na mne. V nej vezmem adresnú časť bunky a z nej posledné 2 bity, podľa ktorých určím o aký posun sa jedná. Ak sa tam nachádza „00“, jedná sa o posun hore, ak je tam „01“ ide o posun dole, ak „10“ je to posun doprava a ak to je „11“ tak to je posun doľava. Inštrukcie vykonávam pokiaľ ich nevykonám všetky, alebo kým nevykonám 500 inštrukcií.

Po zistení cesty ju vyhodnotím, čím zistím, či jedinec náhodou nevystúpil z mriežky, vtedy ukončím prehľadávanie a odseknem zvyšok vygenerovanej cesty, prechádzaním cesty zároveň zistím koľko pokladov sa podarilo pozbierať hľadačovi. Ak sa mu náhodou podarilo pozbierať všetky poklady tak rovnako ukončím prechádzanie cesty a odseknem koniec cesty nakoľko už v nej nie je potrebné pokračovať. Vďaka informácií o prejdenej ceste a pozbieraných pokladoch môžem jedincovi určiť fitness. Ak je nájdený prvý jedinec so všetkými pokladmi, alebo jedinec pozbieral všetky poklady, pričom jeho cesta je kratšia ako doteraz najlepšia, tak sa program spýta používateľa, či chce pokračovať v hľadaní lepšieho riešenia alebo chce ukončiť prehľadávanie. Ak program vygeneroval požadovaný počet generácií, tak sa program spýta používateľa, či chce pokračovať v generovaní, alebo ukončiť prehľadávanie, pričom vypíše najlepšie riešenie a priemer fitness hodnôt generácií spolu s grafom ako sa vyvíjali.

Následne podľa toho akú metódu selekcie si užívateľ vybral prebehne turnaj alebo ruleta, aby sa vybrali rodičia do kríženia. Turnaj dostane vyberie náhodne 5 jedincov a z nich vyberie najlepšieho, ktorý sa stane prvým rodičom. Turnaj sa zopakuje pre výber druhého rodiča, pričom ak sa vyberie rovnaký jedinec, tak je druhý rodič vyberaný znova až pokiaľ to nie sú dvaja rôzny jedinci. Ruleta podobne ako turnaj vyberie najprv jedného a potom druhého rodiča, pričom ak sa rovnajú, tak opäť hľadám druhého rodiča až pokiaľ to nie sú dvaja rôzny jedinci. Rozdiel je však v tom, že jedinca do kríženia vyberá na základe pravdepodobnosti, kde jedinec s väčšou hodnotou fitness má aj logicky väčšiu pravdepodobnosť, že bude vybratý do kríženia. Kríženie vykonávam nasledovne. Vygenerujem náhodnú dĺžku kopírovanej časti z prvého rodiča a náhodný index od 0 po 63 - dĺžka kopírovanej časti a začnem. Následne do dieťaťa nakopírujem gén z prvého rodiča ak som v intervale kopírovania, ak nie tak kopírujem gény z druhého rodiča. Takto vytvorím novú generáciu, ktorá je tvorená zo zadaného percenta elitarizmom, zadaným percentom úplne nových jedincov, ktorí majú náhodne vygenerované gény a zvyšok je vytvorený krížením. Po vytvorení generácie ešte pošlem jedincov, ktorí boli vytvorený krížením na mutáciu, kde zmutujú ich gény pod istou pravdepodobnosťou.

Mutáciu na géne vykonám ak vygenerované náhodné číslo od 0 po 1 je menšie ako pravdepodobnosť mutácie. Ak sa splní podmienka, tak na miesto daného génu vygenerujem nové náhodné číslo od 0 po 255, vrátane. Takto zmutujem jedincov vytvorených krížením a novo vytvorenú populáciu môžem znova poslať na vykonanie sa do virtuálneho stroja a ďalej opakovať cyklus. Ten pokračuje až pokiaľ sa pri nájdení riešenia, alebo nového lepšieho riešenia nerozhodne užívateľ ukončiť hľadanie, alebo sa prejde na začiatku zadaný počet generácií. Avšak aj po dosiahnutí stropu si vie užívateľ vybrať, či chce pokračovať v ďalšom cykle prehľadávania.

1. Zhodnotenie a porovnanie dosahovaných výsledkov

Ako prvé by som chcel poukázať ako fungujú jednotlivé zložky pri tvorbe generácií,