Zenová záhrada

Zadanie č. 3a

Úloha

Zenová záhradka je plocha vysypaná hrubším pieskom (drobnými kamienkami). Obsahuje však aj nepohyblivé väčšie objekty, ako napríklad kamene, sochy, konštrukcie, samorasty. Mních má upraviť piesok v záhradke pomocou hrablí tak, že vzniknú pásy ako na nasledujúcom obrázku.

Pásy môžu ísť len vodorovne alebo zvislo, nikdy nie šikmo. Začína vždy na okraji záhradky a ťahá rovný pás až po druhý okraj alebo po prekážku. Na okraji – mimo záhradky môže chodiť ako chce. Ak však príde k prekážke – kameňu alebo už pohrabanému piesku – musí sa otočiť, ak má kam. Ak má voľné smery vľavo aj vpravo, je jeho vec, kam sa otočí. Ak má voľný len jeden smer, otočí sa tam. Ak sa nemá kam otočiť, je koniec hry. Úspešná hra je taká, v ktorej mních dokáže za daných pravidiel pohrabať celú záhradu, prípade maximálny možný počet políčok. Výstupom je pokrytie danej záhrady prechodmi mnícha.

Úlohu je možné rozšíriť tak, že mních navyše zbiera popadané lístie. Listy musí zbierať v poradí: najprv žlté, potom pomarančové a nakoniec červené. Príklad vidno na obrázku nižšie. Listy, ktoré zatiaľ nemôže zbierať, predstavujú pevnú prekážku. Je potrebné primerane upraviť fitness funkciu. Za takto rozšírenú úlohu je možné získať navyše jeden bonusový bod.

Uvedenú úlohu riešte pomocou evolučného algoritmu. Maximálny počet génov nesmie presiahnuť polovicu obvodu záhrady plus počet kameňov, v našom príklade podľa prvého obrázku 12+10+6=28. Fitnes je určená počtom pohrabaných políčok. Výstupom je matica, znázorňujúca cesty mnícha. Je potrebné, aby program zvládal aspoň záhradku podľa prvého obrázku, ale vstupom môže byť v princípe ľubovoľná mapa.

*Dokumentácia musí obsahovať konkrétne použitý algoritmus (nie len náčrt algoritmu, ako v zadaní), podrobný opis vlastností použitých génov, opis ako sa pohybuje a rozhoduje mních, spôsob tvorby novej generácie a možnosti nastavenia parametrov. Dôležitou časťou dokumentácie je zhodnotenie vlastností vytvoreného systému a porovnanie dosahovaných výsledkov pre viacero nastavení parametrov. Vývoj fitnes je vhodné zobraziť grafom (stredná hodnota, maximálna). Dokumentácia by mala tiež obsahovať opis vylepšovania, dolaďovania riešenia.*

Inšpiráciu k mutáciám a kríženia som čerpal z knihy P. Návrat: Umelá inteligencia, z cvičení a prednášok premetu Umelá inteligencia.

Reprezentácia záhradky

Trieda Garden predstavuje záhradku, konkrétne dvojrozmerné pole gardenMap naplnené hodnotami 0 alebo -1 pre pôdu resp. kameň. Ďalej obsahuje premenné x a y , ktoré slúžia na vytvorenie záhradky daných rozmerov v týchto osiach. Funkciou je createGarden naplní pole gardenMap hodnotami 0 a -1. Funkcia insertStone slúži na naplnenie mapy kameňmi a je volaná vo funkcii createGarden.

Reprezentácia jedinca

Jedinca predstavuje trieda Monk, a obsahuje pole genes ako reprezentáciu chomozómu čiže poľa génov daného mnícha. Pomocou funkcie initializeGenes() sa prvotne inicialzujú náhodné gény jedinca, ktoré sú z daného intervalu od 0 po veľkosť obvodu záhradky-1. Každý jedinec ďalej obsahuje hodnoty ako sila(strength), fitness alebo blocked a counter, ktoré sú potrebné pre vyhodnotenie sily alebo správnosti jedinca. Ďalšie funkcie mnícha sú v nasledujúcej časti dokumentácie.

Algoritmus hrabania záhradky

Inštancia mnícha vytvorí inštanciu záhradky vo funkcii rakeGarden() a volaním g.createGarden() naplní jej polia. Na základe mníchovho génu sa vypočíta vstupná súradnica a určí sa z ktorej strany záhradky vošiel a ktorým smerom ju má hrabať. Gény mnícha sú teda použité iba ako počiatočné súradnice vstupu do záhrady. Mních sa teda pohybuje daným smerom vypočítaným z jeho génu a pred každým posunom na ďalšie políčko sa zisťuje či sa na danom políčku, kde by sa mal posunúť nenachádza prekážka a to vo forme kameňa alebo už pohrabanej záhradky. To sa vykoná volaním funkcie blockAhead(), ktorá vracia do podmienky if() true ak sa nachádza blok na ďalšom políčku a false ak nie. Ak vráti true, mních sa potrebuje otočiť a na výber má dva smery. Ak ide zhora alebo zdola, môže ísť vpravo alebo vľavo, tento prípad je zabezpečený funkciou changeDirection(). Do tejto funkcie ako parametre vojdú súradnice miesta, kde práve stojí mních a zistí sa či sa môže posunúť do daných strán čiže vpravo alebo vľavo. Najprv sa mních pozrie doprava (na políčko s väčšou x-ovou súradnicou) a potom doľava (na políčko s menšou x-ovou súradnicou) a kde sa nachádza nepohrabané políčko tam bude nasledovať. Ak sa nemôže pohnúť ani jedným smerom nastaví premennej blocked hodnotu true. Po vrátení blocked ako true sa skončí hľadanie cesty a vyhodnotí sa fitness daného mnícha. Podobne to funguje aj keď mních ide sprava alebo zľava (funkcia changeDirectionUpDown()) a na výber má cestu dole (na políčko s väčšou y-ovou súradnicou) a hore (na políčko s menšou y-ovou súradnicou). Následne je jeho smer určený funkciami up(), down(), right() a left() ktoré svojimi názvami indikujú ktorým smerom sa bude mních uberať. Ich funkcionalita je rovnaká ako pri vstupe (vyššie opísanom).   
Na záver sa vypočíta fitness mnícha, čiže počet políčok, ktoré mních pohrabal, a vypíše sa cesta záhradkou (tá sa kvôli prehľadnosti po novom vypíše iba pri poslednom teda úspešnom prechode záhradkou).

Algoritmus pre vylepšovanie jedincov

Na vylepšovanie jedincov slúži trieda Evolution, v ktorej sa na začiatku inicializuje prvá generácia jedincov pomocou funkcie initialize(). Pre každého mnícha z populácie vygeneruje náhodné gény a následne mních pohrabe záhradku – proces opisovaný v predošlej kapitole. Na základe výslednej hodnoty fitness sú mnísi zoradený a je im vypočítaná ich sila pomocou funkcie findStrength(). Pri výpočte sily som sa inšpiroval knihou P. Návrata: Umelá inteligencia, s.350. Metóda nájde silu pre všetkých jedincov populácie usporiadaných podľa ich fitness hodnoty.   
Následne je vytvorená nová generácia mníchov na základe tejto prvotnej. Tento proces, ktorý je kľúčový pre program sa odohráva vo funkcii newPopulation(). Robí to podobne ako funkcia initialize(), ale namiesto vygenerovania náhodných génov pre jedincov, volá na základe hodnoty v EVOLUTIONTYPE funkcie evolve(1,2,3...), ktoré vylepšujú gény jedincov napríklad kríženiami alebo mutáciami.

Funkcie evolve() vznikali postupne, ako som sa snažil vylepšiť a analyzovať jednotlivé prístupy k mutáciám a kríženiam. Na výber jedinca som použil dve spôsoby a to ruletu a turnaj.

Na výber pomocou turnaja (funkcia tournament()) som vybral dve náhodné čísla z rozsahu 0..numberOfMonks(počet mníchov v populácii) – 1 a vybral som toho z nich, ktorý ml vyššiu hodnotu sily (tým pádom aj hodnotu fitness, keďže sila sa vyvíja úmerne s fitness).  
 Pri výbere ruletou som najprv použil výber na základe sily, ktorú som vypočítal na základe knihy P. Návrata: Umelá inteligencia, spočítal som sily všetkých jedincov a vynásobil náhodným číslom z intervalu (0,1>. Následne od tejto hodnoty odčítaval sily jedincov populácie od najslabšieho, čiže s najnižšou hodnotou strength. Jedinec, pri ktorom sa hodnota „kvázi súčtu“ zníži po odčítaní pod nulu sa stane vyhľadaným rodičom pre kríženie resp. mutáciu.   
Tento spôsob mi pri testovaní neprinášal však požadované hodnoty a v porovnaní s turnajom bola omnoho slabšia, a preto som sa rozhodol vytvoriť novú funkciu, ktorá by nehodnotila jedincov podľa sily, ale podľa samotnej fitness hodnoty jedincov. Túto funkciu som nazval myRoulette() a vykonáva sa v podstate rovnakým spôsobom ako predošlá, avšak s rozdielom že sa spočítavajú a následne odčítavajú hodnoty fitness jedincov. Táto funkcia, ako bude spomenuté v kapitole Testovanie, prinášala markantne lepšie výsledky pri kríženiach a mutáciách.

Kríženie chromozómov mníchov sa vykonáva vo funkciách crossoverTournament() a crossoverRoulette() a jednoduché rozoznať podľa názvu aký spôsob výberu rodičov sa kde využíva. Prvá zo spomínaných funkcií zobrazuje spôsob kríženia, pri ktorom sa pomocou turnaja vyberú dvaja rodičia a vypočíta sa limit pre kríženie – náhodné číslo z intervalu (0,1> sa vynásobí počtom génov jedinca. Na základe tohto limitu sa vyberú gény pre nového jedinca od prvého rodiča(od začiatku chromozómu po daný limit) a od druhého rodiča (od limitu po koniec chromozómu). Inšpirácie ku kríženiu: Návrat P.: Umelá inteligencia s.348 a samozrejme cvičenia ku predmetu UI. Druhá zo spomínaných funkcii funguje na rovnakom princípe avšak rodičia sú vyberaný pomocou funkcii roulette() a myRoulette() – ktorá z nich sa využije je určená boolean hodnotou myRoulette (true – pouťitá funkcia myRoulette).

Mutáciu chromozómov som riešil pomocou komplementu k danému číslu, ktoré reprezentuje gén v chromozóme a to tak že od maximálnej veľkosti čísla v géne odčítal hodnotu génu rodiča spravil som absolútnu hodnotu tohto výsledného čísla. To sa deje vo funkciách mutationComplementTournament() a mutationComplementRoulette().

Pri tvorení nových generácii vo funkciách evolve() som použil okrem predošlých spôsobov aj elitizmus, ktorý je znázornený funkciou elitism() (len kvôli lepšej prehľadnosti, dalo sa to spraviť jednoduchšie), kde vyjadrujem, že jedinci, ktorý majú najvyššiu hodnotu fitness majú právo zotrvať aj v nasledujúcej generácii (Návrat P.: Umelá inteligencia s.349). Tento spôsob som využil pri desatine všetkých jedincov a pri skoro každej z evolve() funkcii. Taktiež som využil formu mutácie, kde som vymenil celý chromozóm jedinca, ktorý mal najnižšiu hodnotu fitness a nahradil ju náhodnými génami aby som takto zamedzil uviaznutiu pri vysokých fitness hodnotách a priniesol nejaké nové potenciálne lepšie gény na kríženie.

Evolve1: elitizmus, kríženie a výber ruletou(fitness), kríženie a výber turnajom, mutácia a turnaj, mutácia a ruleta(fitness), náhodná generácia 110

Evolve2: elitizmus, kríženie ruleta(sila), mutácia ruleta(sila), náhodná generácia 103, 120

Evolve2.1: elitizmus, kríženie ruleta(fitness), mutácia ruleta(fitness), náhodná generácia 104, 108

Evolve3: elitizmus, kríženie turnaj, mutácia turnaj, náhodná generácia 142

Evolve4: elitizmus, kríženie turnaj, kríženie ruleta,(sila) náhodná generácia 77

Evolve4.1: elitizmus, kríženie turnaj, kríženie ruleta,(fitness) náhodná generácia 48, 55, 70

Evolve5: elitizmus, mutácia ruleta(sila), mutácia turnaj, náhodná generácia 487

Evolve5.1: elitizmus, mutácia ruleta(fitness), mutácia turnaj, náhodná generácia 478

Evolve6: kombinácia 4.1 pre fitness najlepšieho prvku hodnotu menšiu ako maximálna možná fitness -1 a 2.1 ak sa rovná maximálnej možnej fitness bez jednej, chcel som použiť to najlepšie z kríženia ak je ešte ďaleko od maxima a pridať mutáciu keď sa blíži k maximu aby sa nezaseklo na lokálnom maxime 68, 94, 79

Evolve7: náhodné gény vygenerované pre každú populáciu 199, 194

Na záver sa vykoná hrabanie záhradky novou generáciou mníchov, vyhodnotí sa ich fitness a sila a zistí sa, či sa v novej populácii nenašiel mních, ktorý pohrabal celú záhradku. Ak sa našiel, ukončí sa vykonávanie procedúry a prípadne sa vyhodnotia populácie a mních, v opačnom prípade sa opäť zavolá procedúra newPopulation(), ale s novou populáciou ako parametrom.

Riešenie som postupne vylepšoval.

Testovanie

Zhodnotenie

Používateľská príručka