Dokumentácia - Umelá Inteligencia
Zadanie 3 – zenová záhradka
TS + SA

Cvičenie: štvrtok 16.00
Cvičiaci: Ing. Boris Slíž
Meno: Martin Rudolf

Zadanie

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. Pokrytie zodpovedajúce presne prvému obrázku (priebežný stav) je napríklad takéto:

0	0	1	0	0	0	0	0	10	10	8	9
0	0	1	0	0	K	0	0	10	10	8	9
0	K	1	0	0	0	0	0	10	10	8	9
0	0	1	1	K	0	0	0	10	10	8	9
0	0	K	1	0	0	0	0	10	10	8	9
2	2	2	1	0	0	0	0	10	10	8	9
3	3	2	1	0	0	0	0	K	K	8	8
4	3	2	1	0	0	0	0	5	5	5	5
4	3	2	1	0	0	0	11	5	6	6	6
4	3	2	1	0	0	0	11	5	6	7	7

TabuSearch

Algoritmus dostáva ako vstupné argumenty rodičovskú inštanciu pohrabanej záhradky(candidate), inštanciu mnícha(monk) a počiatočný chromozóm ktorý na indexe 0 nesie premiešane (shuffle()) počiatočné štartovacie pozície, vytvorené všetky možné začiatočné pozície metódou startPositions() inštancie garden, a na indexe 1, pole zostavené z hodnôt 1, 0 čo udáva pravdepodobnosť otočenia pri narazení do prekážky. Na začiatku algoritmus inicializuje najlepšieho kandidáta na vstupnú hodnotu záhradky (sBest = candidate) a priradí ju do TabuListu. Začne sa cyklus v ktorom sa vytvoria susedia, zo susedov sa vyberie najlepší a porovnáva sa s celkovým najlepším. Ak je jeho ohodnotenie väčšie ako aktuálne najlepšie ohodnotenie sBest = bestCandidate, a priradí sa do TabuListu.

Na vytvorenie susedov potrebujeme najprv vygenerovať nový chromozóm o čo sa stará metóda inštancie monk neighbor(), táto metóda zamení 2 náhodné počiatočné pozície aktuálne najlepšie pohrabanej záhradky (sBest), ktoré sa priradia do chromozómu na index 0. Následne takto novo vygenerovaný chromozóm použijeme na vytvorenie susedov pomocou metódy getNeighbors(). Táto funkcia najprv vytvorí novú inštanciu záhradky, inštanciu mnícha pomocou už vygenerovaného chromozómu, a ten následne pohrabe záhradku neighmonk.solve(g). Pohrabanú záhradku priradí do poľa neighbors ktoré funkcia vraca.

Metóda inštancie mnícha solve() funguje nasledovne: na začiatku sa mních postaví na prvú pozíciu z indexu 0 v chromozóme, kontroluje vždy políčko pred sebou, ak je prázdne tak ho pohrabe ak je to políčko mimo záhradky vyberie si novú štartovaciu pozíciu, no ak pole s pozíciami je prázdne tak záhradka je pohrabaná ako sa len dala pre dané pozície a dané rotácie. Ak pri svojom hrabaní mních narazí na kameň musí sa rozhodnúť či sa otočí vpravo alebo vľavo. Táto voľba prebieha náhodne, resp. Na základe atribútu mnícha (rotationPool) čo je vlastne pole na indexe 1 v chromozóme. O samotnú rotáciu sa stará funkcia rotate(). No ešte pred samotným otočením sa skontroluje či políčka vpravo a vľavo sú voľné ak nie je voľné ani jedno políčko cyklus sa ukončí, ak je voľná len jedna možnosť tak si ju vyberie ak sú voľné oba možnosti náhodne si vyberie jednu ako bolo vyššie spomenuté.

Po vygenerovaní susedov sa vyberie najlepšie ohodnotený z nich. Pri výbere sa najprv pozrieme či sa nenachádza v TabuListe, ak áno tak hned prejdeme na druhého, ak nie tak porovnáme s najlepším kandidátom bestCandidate. Takto vybraného suseda následne porovnáme s aktuálnym sBest a priradíme do TabuListu. Nakoniec skontrolujeme či tabu list nepresiahol maximálnu dĺžku ak áno tak vyhodíme prvý prvok.

Funkcia na ohodnotenie záhradky spočíta všetky nenulové (pohrabané, kamenisté) políčka.

Simulované Žíhanie

Na začiatku sa inicializuje teplota, finálna teplota a alfa. V cykle while (aktuálna teplota > finalna teplota) spustime cyklus for in range(K) vytvoríme náhodného suseda ohodnotením aktuálneho a susedného prvku zistíme rozdiel medzi nimi, ak je susedný prvok väčší ako aktuálny najlepší, tak susedný sa stáva aktuálnym najlepším, ak nie, tak zbehne podmienka

ktorá s pravdepodobnosťou 1/(math.e) ** (diff /current_temp) určí suseda ako najlepšieho aktuálneho. algoritmus skončí ak podmienka s pravdepodobnosťou neprejde K – krát, alebo ak teplota nebude rovná minimálnej teplote. Čím je teplota nižšia tým je pravdepodobnosť priradenia horšieho suseda do najlepšieho menšia. Algoritmus vráti najlepšie ohodnotený prvok (best), teda je inštanciu najlepšie pohrabanej záhradky.

Zhodnotenie a testovanie

Tabu search ma tendenciu sa zaseknuť na lokálnom maxime, preto je potrebné ho spustiť viac krát. Mnou nainmplementovaný tabu search s parametrom pravdepodobnosti otáčania [0,1,1,1], s cyklenim 100-150, pri vytváraní 10 susedov, a velkosťou tabu listu 30 vráti ohodnotenie najlepšie pohrabanej záhradky v priemere 111/120.

Pri Simulovanom žíhanií je možnosť sa dostať z lokálného maxima, čo ma na svedomí teplota a velkosť rozdielu aktualneho a susedneho prvku. Teplota žíhania reprezentuje, že z okolia súčastného stavu vyberieme za nasledovníka stav, ktorý nie je lepši resp. najlepší. Rýchlosť algoritmu závisí od počiatočnej teploty a hodnoty alfa. Mnou implementované simulované žíhanie s parametrom pravdepodobnosti otáčania [0,1,1,1], počiatočnou tepolotou = 90 a hodnotou alfa = 0.01, priemerná hodnota pohrabanej záhradky je 118 za priemerny čas 183s.