FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ
SLOVENSKÁ TECHNICKÁ UNIVERZITA

UMELÁ INTELIGENCIA

Zadanie č.2 a)

(Zenová záhrada)

Meno: René Bukovina
Ročník: 3.
Cvičenie: pondelok 9:00
Cvičiaci: Ing. Martin Komák, Phd.

Obsah

Definovanie problému

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.

Uvedenú úlohu riešte pomocou evolučného algoritmu. (Je možné použiť aj ďalšie algoritmy, ako sú uvedené v probléme obchodného cestujúceho.) 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.

Popis algoritmu

Zadanie bolo riešené evolučným algoritmom, ako bolo uvedené v zdaní. Princíp algoritmu spočíva v tom, že:

1. Program dostane vstup od používateľa (šírka a výška mapy/záhrady, súradnice kameňov na mape, metóda výberu jedincov)
2. Program vytvorí, prvú generáciu náhodne, podľa vstupných údajov.
3. Program získa ohodnotenie všetkých jedincov pomocou fitness funkcie
4. Program použije zvolenú metódu výberu pre výber jedincov na kríženie
5. Program vytvorí novú generáciu pomocou kríženia vybraných jedincov
6. Program postupne prejde každého jedinca novej generácie s pravdepodobnosťou 10%, že daného jedinca aj zmutuje.
7. Pôvodnú generáciu nahradí novou generáciou a pokračuje znova od kroku č. 4 pokiaľ nenájde optimálne riešenie, alebo pokiaľ nevykoná 100 000 nových generácií, v takom prípade nakoniec vypíše jedinca s najlepším dosiahnutým ohodnotením

Za zmienku stojí fakt, že algoritmus si uchováva priebežného najlepšieho jedinca, pre prípad, že by nenašiel optimálny výsledok.

Popis použitých dátových štruktúr

Nakoľko bol program vytvorený vo vývojovom prostredí jazyka JavaScript, konkrétne v Node.js s nadstavbou TypeScript pre ošetrovanie typov, bolo by vhodné popísať si niektoré dátové štruktúry.

Štruktúra Population

Jednorozmerné pole obsahujúce objekty typu Individual, čiže jedincov.

```
type Population = Array<Individual>
```

Štruktúra Individual

Taktiež jednorozmerné pole obsahujúce objekty typu Gen

```
type Individual = Array<Gen>
```

Štruktúra Gen

Dátová štruktúra Gen predstavuje objekt, ktorý obsahuje vlastnosti position a decisions.

```
type Gen = {
      position:  číslo štartovacej pozície pre daný ťah
      decisions: pole 20-tich znakov, označujúce rozhodnutia pri
      prekážkach
}
```

Štruktúra GradedPopulation

Rozšírená štruktúra Population, ktorá je výstupom fitness funkcie a priradzuje jedincom ich dosiahnuté skóre zároveň s maticou postupností.

```
type GradedPopulation = {
      individual,
      score,
      grid
}
```

Funkcia movement()

Funkcia movement, sa nachádza v súbore movement.ts a ma nastarosti prakticky celú logiku pohybov mnícha pre daný ťah. Stručne popíšeme iba logiku funkcie nakoľko je jej zdrojový kód veľmi detailne okomentovaný v danom súbore

Vstupné argumenty:

- **Gen,** pre ktorý sa ma daný ťah vykonávať
- **Počiatočný smer**, z ktoré mních vyrazí
- **Mapa**
- **Index**, označujúci poradie ťahu zapisovaného do mapy

Logika:

1. Program skontroluje či je počiatočné pozícia obsadená alebo nie
2. Ak je obsadená funkcia skončí a vráti skóre s hodnotou 0

3. Ak nie je obsadená, tak podľa počiatočného smeru vyráta súradnice, z ktorých bude mních začínať.
4. Na dané políčko zapíše aktuálny index
5. Pokračuje na ďalšie políčko v danom smere
6. Ak nenarazil na prekážku alebo nevyšiel zo záhrady pokračuje na krok 4.
7. Ak narazil na prekážku, tak sa pozrie akým smerom sa môže otočiť. Ak ma na výber z viacerých smerov, tak vyberie prvú položku z poľa decisions pre daný gén. Ak na výber nemá tak sa obratí len tam kam môže, zmení aktuálny smer pohybu a pokračuje znova na krok 4.
8. Ak sa mních zasekol funkcia vráti dosiahnuté skoré aj informáciu o tom, že sa mních zasekol, následne ukončí skúmanie daného jedinca, nakoľko sa zasekol.
9. Ak mních vyšiel zo záhrady, tak funkcia vráti dosiahnuté skoré s informáciou, že mních môže pokračovať z ďalšieho políčka v poradí.

Funkcia fitness()

Vstupy do funkcie movement používa fitness funkcia, nakoľko je táto funkcia pomerne jednoducho napísaná, nie je dôvod si ju bližšie popisovať, jedinou logikou za týmto je, že sekvenčne prechádza všetky gény daného jedinca a posiela ich do funkcie movement. Taktiež funkcia fitness ukladá priebežne dosiahnuté skóre daného jedinca a zároveň aj jeho mapu pohybov, ktoré na konci vráti ako údaje pre novú generáciu.

Záver

Možnosti vylepšenia systému sú veľmi obmedzené, nakoľko je chod programu v momentálnom stave veľmi optimalizovaný a nie je tam preto veľa priestoru na zásadné zlepšovanie resp. vylepšovanie.

Aj napriek tomu, že sa jazyk JavaScript považuje za pomerne pomalý výpočtový jazyk, je program veľmi efektívny v hľadaný optimálneho riešenia a výsledok daného problému je schopný dodať do 10-tich sekúnd, ale v priemere je to menej ako 5 sekúnd. Navyše je program plne dynamicky, čiže nezáleží na tom aký vstup dostane.

Momentálnou limitáciou programu je maximálny počet vytvorených generácii a to 100 000, čo je ale zanedbateľné, nakoľko vieme tento limit veľmi rýchlo navýšiť.

Príručka pre spustenie programu

Potrebné: node.js (verzia 16 alebo viac)

Návod na spustenie:

- V adresári s projektom otvoríme ľubovoľný terminál
- Použijeme príkaz `npm i` alebo `npm install`
- Po dokončení inštalácie môžeme spustiť program príkazom `npm run start`
- Zadáme vstupné údaje a necháme vykonať program