Inteligenţă Artificială
2020

Tema 1

Pentru a determina drumul minim între o stare inițială și o stare scop într-un mediu definit prin stări, un agent poate utiliza mai mulți algoritmi eficienți de căutare, care explorează și salvează informații despre o mare parte din spațiul stărilor. Dacă însă numărul stărilor este considerabil, iar agentul nu are o putere de calcul mare, atunci trebuie să aplicăm alți algoritmi care se pot descurca și cu memorie limitată.

Scopul acestei teme este să aplicăm mai mulți algoritmi de căutare informată în spațiul stărilor pentru a determina drumul minim prin care un șoricel poate ajunge la brânză. Harta pe care șoricelul se mișcă va fi dată sub formă de graf printr-un fișier de intrare. Drumul pe care șoricelul îl va urma trebuie să ocolească obstacolele din mediu. Fiecare tranziție dintr-o stare în alta va avea asociat un cost, iar funcția euristică pe care o vom folosi în algoritmii de căutare informată va fi funcția euclidiană.

Cerinţe:

Cerința 0 (5p): Implementați funcțiile necesare pentru definirea mediului. Printre funcțiile implementate va trebui să aveți:

- `init_env` - care va citi datele din fișierul de intrare și le va salva sub formă de graf
- `get_next_states` - va genera toate stările succesoare pentru starea dată ca parametru
- `apply_action` - va aplica o acțiune asupra unei stări pentru a genera starea următoare

Cerința 1 (20p): Determinați drumul pe care trebuie să îl parcurgă un agent de la starea inițială la starea finală folosind algoritmul **Depth First Iterative Deepening**.

Cerința 2 (20p): Determinați drumul pe care trebuie să îl parcurgă un agent de la starea inițială la starea finală folosind algoritmul **Iterative Deepening A***.

Cerința 3 (20p): Determinați drumul pe care trebuie să îl parcurgă un agent de la starea inițială la starea finală folosind algoritmul **Learning Real Time A***.

Cerința 4 (10p): Implementați o altă funcție euristică potrivită pentru problema dată. Justificați alegerea făcută. Se exclude distanța Manhattan.

Cerinţa 5 (25p): Comparaţi metodele propuse între ele, din punctul de vedere al costului obţinut şi al timpului de execuţie. Evaluaţi diferenţa între funcţia euristică dată şi cea propusă de voi. Rezultatele vor fi scrise într-un document *PDF*. Acesta va trebui să includă grafice comparative după cost, timp şi funcţie euristică, alături de explicaţia acestor grafice.

Bonus (20p): Determinaţi drumul pe care trebuie să îl parcurgă un agent de la starea iniţială la starea finală folosind algoritmul **Branch and Bound**.

Formatul fişierului de intrare:

```
mouse_pos_x, mouse_pos_y
cheese_pos_x, cheese_pos_y
positions_no
pos_id, pos_x, pos_y, [obstacle]
…
edges_no
pos1_id, pos2_id, cost
…
```

- prima linie reprezintă poziţia în care se află şoricelul
- a doua linie reprezintă poziţia în care se află brânza
- a treia linie conţine numărul de poziţii posibile şi e urmată de informaţiile asociate acelor poziţii: ID-ul poziţiei, poziţia 2D şi un parametru opţional care specifică dacă poziţia reprezintă un obstacol
- o linie pentru numărul de muchii urmată de legăturile între poziţii cu costurile asociate

Rezultatul unui algoritm de calcul al drumului va consta într-o secvenţă de poziţii pe care trebuie să le parcurgă şoricelul de la poziţia iniţială la poziţia brânzei, alături de costul asociat drumului.

Pentru a testa algoritmii implementaţi puteţi folosi pe lângă cele 3 fişiere input propuse şi alte fişiere create de voi, cu menţiunea că trebuie incluse şi acelea în arhivă şi în graficele de la cerinţa 5. Fişierele date au fost generate pe baza unor hărţi 2D, din care au fost excluse unele poziţii.