Documentatie – Tema 1 IA

*Apelarea programului*

Apelul trebuie sa urmeze urmatoarea structura:

canibali-misionari.py [cale folder input] [cale folder output] [NSOL] [Timp de timeout (format MM:SS)]

Sau, alternativ:

canibali-misionari.py [cale folder output] [NSOL] [Timp de timeout (format MM:SS)]

In al doilea caz se va considera ca folderul de input este chiar folder Input din arhiva.

*Explicarea euristicilor folosite*

Euristica 1: numarul de canibali de pe mal + numarul de misionari de pe mal – 1

Euristica 2: (numarul de canibali de pe mal + numarul de misionari de pe mal)/capacitatea barcii

Euristica neadmisibila: hrana de pe mal initial+hrana de pe mal final

Ambele euristici (1 si 2) sunt admisibile deoarece se obtin prin relaxarea conditiilor problemei, mai exact a conditiei conform careia in barca nu pot exista mai multi canibali decat misionari (nehraniti). Astfel, calculez numarul de drumuri de care as avea nevoie pentru a muta toti oamenii, in orice formula. Acest numar va fi intotdeauna mai mic sau egal decat h\*.

In cazul euristicii neadmisbile, in cazul starii [0,1,2,1,1] (de forma [Canibali, Misionari, Hrana mal initial, Hrana mal final, Indicator barca la mal]) h-ul estimat va fi 3, pe cand h-ul real este 1. Este nevoie de o singura mutare (barca sa plece cu 1 misionar) pentru a ajunge in starea finala [0,0,2,1,0].

*Fisiere de input*

Fisierul input\_a contine un input pentru care nu exista solutie, fisierele input\_c si input\_c\_bis contin inputuri care au solutii si nu se blocheaza pe niciun algoritm pentru un timeout de 1 minut, iar fisierul input\_d contine un input care are solutie si se blocheaza cu UCS si A\* (cu euristica banala, euristica 1 si euristica neadmisibila) pentru un timeout de 1 minut, insa gaseste solutia cu A\* (euristica 1), A\* optimizat si IDA\*. Nu am creat un fisier input\_b cu o stare initiala care sa fie si finala deoarece un astfel de scenariu nu are sens in contextul acestei probleme de traversare a raului.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Input**  *input\_c si input\_c\_bis* |  | **UCS** | **A\*** | | | | **A\* optimizat** | | | |
| Banala | Admisib 1 | Admisib 2 | Neadmisib | Banala | Admisib 1 | Admisib 2 | Neadmisib |
| N1=3  N2=3  K=3  M=2  Nr=30  Mal initial=vest  Mal final=est | Lungime drum | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 |
| Cost drum | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 | 14 |
| Nr. max. noduri memorie | 1827 | 1279 | 789 | 1303 | 1394 | 143 | 137 | 130 | 137 |
| Nr. total de succesori | 4036 | 2918 | 1591 | 2474 | 3263 | 714 | 539 | 640 | 793 |
| Timp executie | 1.47 s | 1.0 s | 0.16 s | 0.42 s | 0.61 s | 0.08 s | 0.09 s | 0.09 s | 0.06 s |
| N1=4  N2=5  K=0  M=3  Nr=5  Mal initial=vest  Mal final=est | Lungime drum | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 |
| Cost drum | 17 | 17 | 17 | 17 | 17 | 17 | 17 | 17 | 17 |
| Nr. max. noduri memorie | 203 | 176 | 200 | 216 | 183 | 13 | 15 | 14 | 13 |
| Nr. total de succesori | 977 | 843 | 610 | 757 | 1032 | 123 | 106 | 103 | 133 |
| Timp executie | 0.13 s | 0.12 s | 0.04 s | 0.09 s | 0.13 s | 0.01 s | 0.01 s | 0.01 s | 0.01 s |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Input** |  | **IDA\*** | | | |
| Banala | Admisib 1 | Admisib 2 | Neadmisib |
| N1=3  N2=3  K=3  M=2  Nr=30  Mal initial=vest  Mal final=est | Lungime drum | 10 | 10 | 10 | 10 |
| Cost drum | 14 | 14 | 14 | 14 |
| Nr. max. noduri memorie | 12 | 11 | 11 | 12 |
| Nr. total de succesori | 6660 | 3228 | 4683 | 11776 |
| Timp executie | 0.41 s | 0.17 s | 0.23 s |  |
| N1=4  N2=5  K=0  M=3  Nr=5  Mal initial=vest  Mal final=est | Lungime drum | 10 | 10 | 10 | 10 |
| Cost drum | 17 | 17 | 17 | 17 |
| Nr. max. noduri memorie | 11 | 11 | 11 | 12 |
| Nr. total de succesori | 4424 | 1893 | 3353 | 5395 |
| Timp executie | 0.25 s | 0.12 s | 0.14 s | 0.24 s |

Din datele colectate putem observa ca totii algoritmii returneaza aceeasi solutie (aceeasi lungime al drumului si acelasi cost).

Din punct de vedere al timpului de executie si al numarului de succesori calculati A\* optimizat este cel mai eficient algoritm, insa din punct de vedere al memoriei IDA\* are cel mai mic numar maxim de noduri retinute in memorie la un moment dat. Pe de alta parte, IDA\*, precum si UCS, ating cele mai mari valori pentru numarul total de succesori calculati.