Tema 1 – Bot Clean

Vlad Bogolin

IA 2016

1 Scopul temei

Scopul temei îl reprezintă familiarizarea cu conceptul de planificare și implementarea unui algoritm pentru a rezolva o problemă de planificare.

2 Bot Clean

2.1 Descrierea problemei

Obiectivul temei este acela de a curăța un mediu format din mai multe camere interconectate. Pentru a realiza acest lucru, avem la dispoziție un robot pe care îl putem folosi pentru a face curățenie. În momentul în care robotul curăță o camera, el primește o recompensă egală cu dimensiunea camerei. Așadar, scopul final este acela de a elabora un plan pentru a acumula o recompensă cât mai mare într-un anumit interval de timp. În continuare vom face o descriere a fiecărui element din universul problemei.

Mediul Mediul este reprezentat printr-un graf neorientat (muchiile sunt bidirecționale). Nodurile sunt reprezentate de camere sau de depozite de substțante, iar muchiile au un cost asociat care reprezintă timpul necesar parcurgerii distanței dintre cele două noduri adiacente. Mediul nu se modifică pe toată durata simulării.

Camerele O cameră este reprezentată printr-un nod în graf. O cameră poate fi la un anumit moment de timp curată sau murdară. În cazul în care camera este murdară, ea are asociată o listă de substanțe necesare curățării cât și cantitățile necesare din fiecare substanță. De asemenea, camerele au dimensiuni diferite, iar timpul de curățare este egal cu dimensiunea camerei. Camerele sunt identificate printr-un identificator unic din intervalul [0, numărul maxim de noduri din graf). Nu pot exista camere pe care robotul nu le poate curăța - vezi detalii Robot.

Depozite Depozitele sunt noduri speciale în graf din care robotul poate acumula substanțe de curățare. Depozitul dispune de o cantitate infinită din fiecare substanță, iar în orice depozit se găsesc toate substanțele de curățare. Depozitele sunt identificate printr-un identificator unic din intevalul [0, numărul maxim de noduri din graf).

Robotul Robotul se află inițial într-o cameră de start, care va fi specificată (vezi secțiunea Date de intrare). Robotul dispune de un compartiment în care poate transporta substanțe de curățare. Robotul poate transporta oricâte substanțe de curățare, dar are o capacitate maximă asociată fiecărui tip de substanță. Inițial robotul are în stoc capacitatea maximă din fiecare substanță de curățare. Capacitatea este aceeași pentru toate tipurile de substanțe. Acțiunile pe care le poate efectua robotul sunt:

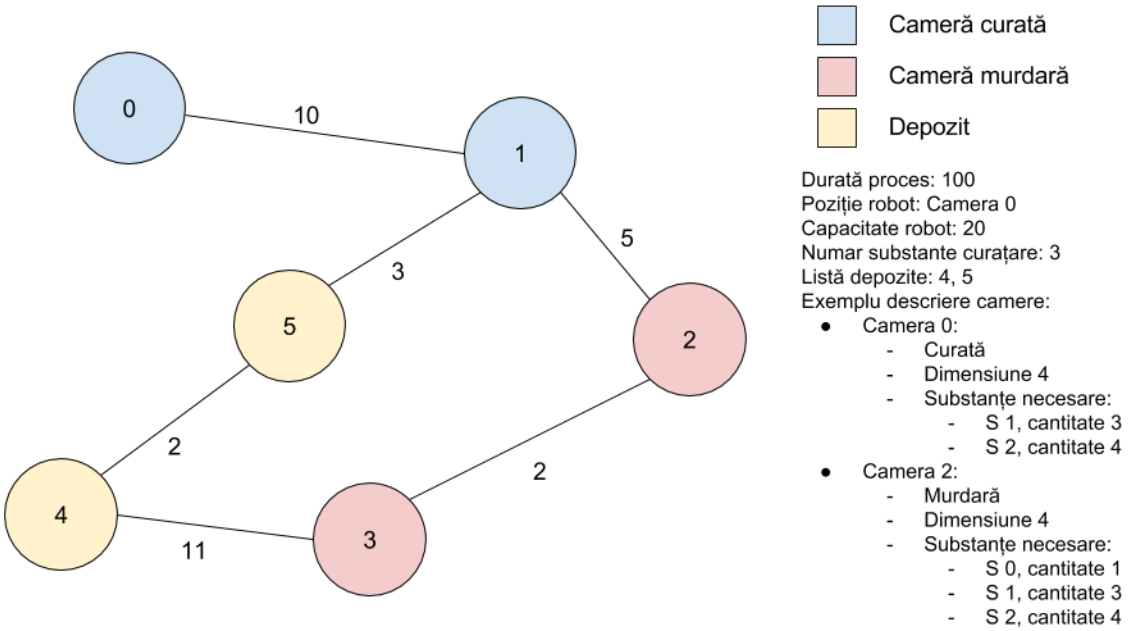
- Deplasarea dintr-o cameră în alta. Acest lucru consumă un timp egal cu costul muchiei dintre cele două camere adiacente.
- Curățarea unei anumite camere dacă camera este murdară, iar cantitatea pentru fiecare substanță necesară este suficientă. În cazul în care robotul nu are o cantitate suficientă dintr-o anumită substanță, curățarea camerei nu se poate efectua. Timpul necesar curățării este egal cu dimensiunea camerei. Timpul necesar curățării nu depinde de numărul de substanțe folosite pentru curățare. În momentul în care procesul de curățare al unei camere s-a terminat robotul va primi o recompensă egală cu timpul de curățare.

- Acumularea de substanțe de curățare. Acestă acțiune se poate efectua doar dacă robotul se află într-un depozit. Robotul poate stoca oricâte tipuri de substanțe diferite, dar are o capacitate maximă asociată fiecărui tip de substanță de curățare. Capacitatea maximă de stocare este aceeași pentru toate tipurile de substanțe. Acumularea de substanțe se efectuează într-o unitate de timp indiferent de numărul de substanțe acumulate.

2.2 Date de intrare

Datele de intrare se vor citi dintr-un fișier care are următoarea formă:

- pe prima linie `X` - numărul total de noduri din graf, `T` - durata maximă a procesului de curățare, `S`- numărul de substanțe diferite, `C` - capacitatea robotului, `M` - numărul de depozite, `N` - numărul de camere, `P` - numărul de muchii din graf, `Start` - poziția de start a robotului specificată printr-un indice al unei camere.
- următoarea linie conține `M` elemente care reprezintă indicii depozitelor
- urmatoarele `P` linii conțin triplete de forma `I`, `J`, `Cost` ce reprezintă faptul că între nodul `I` și nodul `J` există o muchie de cost `Cost`
- următoarele `N` linii conțin descrierea camerelor: prima poziție indică indicele camerei, a doua poziție indică dacă camera este murdară sau nu (1 - murdar, 0 - curat), a treia poziție conține dimensiunea camerei, a patra poziție conține numărul de substanțe ce sunt necesare pentru a curăța camera, iar apoi urmează o listă de substanțe și cantități ce sunt necesare pentru a curăța camera



Exemplu fișier intrare pentru figura de mai sus

```
6 100 3 20 2 4 6 0
4 5
0 1 10
1 2 5
2 3 2
3 4 11
4 5 2
1 5 3
0 0 4 2 1 3 2 4
1 0 2 2 0 2 1 4
2 1 4 3 0 1 1 3 2 4
3 1 6 1 0 2
```

2.3 Reprezentarea cunoștințelor cu ajutorul predicatelor

În această secțiune vom defini predicate folositoare pentru a defini starea lumii cu ajutorul lor:

- `location(Room)` cu semnificația că robotul se află în camera `Room`
- `dimension(Room, N)` care indică faptul că dimensiunea camerei `Room` este `N`
- `substance(Room, Substance, N)` care indică faptul că este necesară o cantitate `N` de substanță `Substance` pentru a curăța camera `Room`
- `isDirty(Room)` care indică dacă camera `Room` este murdară sau curată
- `edge(Node1, Node2, Cost)` care indică faptul că între nodul `Node1` și `Node2` există o muchie de cost `Cost`
- `isRoom(Node)` care indică dacă nodul `Node` este o cameră.
- `isWarehouse(Node)` care indică dacă nodul `Node` este un depozit.
- `capacity(N)` specifică faptul că robotul are capacitatea `N`
- `carries(Substance, N)` care specifică faptul că robotul are în stoc o cantitate `N` de substanță `Substance`
- `equal(N1, N2)` predicat care este adevărat pentru orice două numere naturale `N1` și `N2` care sunt egale (`N1 = N2`).
- `sum(N1, N2, Sum)` predicat care este adevărat dacă oricare ar fi `N1` și `N2` două numere naturale, suma lor este `Sum`.

2.4 Operatori

Planurile conțin următorii operatori:

- `Move(Room1, Room2)` care reprezintă acțiunea prin care robotul se mută din camera `Room1` în camera `Room2`. Acțiunea `Move` reușește doar dacă există o legătură directă între `Room1` și `Room2`.
- `Clean(Room)` care reprezintă acțiunea prin care robotul curață camera `Room`. Acțiunea `Clean` reușește doar dacă cantitatea necesară din fiecare substanță de curățare din inventarul robotului este mai mare sau egală cu cantitatea necesară din substanța respectivă pentru a curăța camera `Room`. Dacă acțiunea reușește, inventarul robotului se modifică în mod corespunzător.
- `Refill(Warehouse)` care reprezintă acțiunea prin care robotul își reîncarcă inventarul de substanțe de curățare la capacitate maximă din depozitul `Warehouse`. Acțiunea reușește doar dacă robotul se află în depozitul `Warehouse`.

2.5 Planificator

Planificatorul robotului primește următoarele date de intrare:

- starea lumii: aranjarea camerelor în univers (graful), lista substanțelor necesare curățării fiecărei camere, indicele depozitelor, poziția curentă, etc

Rezultatul acțiunii de planificare conține o secvență de operatori `Move`, `Clean` și `Refill`.

3 Cerințe

3.1 [0.3p] Cerința 1: Descriere STRIPS

Folosind predicatele enumerate în secțiunea anterioară și eventual alte predicate descrieți următorii operatori folosind STRIPS: `Move`, `Clean`, `Refill`.

3.2 [0.7p] Cerința 2: Planificare

Să se implementeze funcția `makeplan(filename)` care construiește planul pentru a putea curăța toate camerele. Parametrul `filename` reprezintă un fișier care codifică starea universului (vezi secțiunea Date de intrare). Planificatorul returnează un plan format dintr-o listă de operatori în care fiecare operator este reprezentat printr-un șir de caractere. Planificatorul va fi folosit în modul următor:

```
p <- makeplan(filename)
CÂT TIMP mai sunt camere murdare
        ȘI planul p mai conține operatori
        ȘI timpul nu s-a epuizat EXECUTĂ
    o1 <- remove_first(p)
    DACĂ se poate aplica o1 ATUNCI
        EXECUTĂ o1
    ALTFEL
        întrerupe execuția planului
calculează scorul final
```

Scorul final este calculat prin procesarea planului întors de functia (makeplan) ca fiind recompensa pe care o are robotul în timpul (T). Orice acțiune care se execută după mai mult de (T) unitați de timp va fi ignorată.

Pentru implementarea acestei cerințe se poate folosi orice strategie de planificare (căutare înainte, căutare înapoi, altceva).

3.3 [max 0.2p] Cerința 3: Bonus

Se va acorda un bonus de maxim 0.2p pentru cele mai bune 50% scoruri obținute pe un test de dimensiuni relativ mari.

4 Trimiterea temei

Tema se va trimite într-o arhivă `.zip` care are următorul nume:

`Nume_Prenume_Grupa_IA_T2.zip`

În arhivă trebuie să existe cel puțin următoarele 3 fișiere:

- `strips.pdf` care conține rezolvarea cerinței 1.
- `bot_clean.py` care conține rezolvarea cerinței 2. Este **obligatoriu** ca funcția `makeplan` să fie definită în acest fișier.
- `readme.txt` care conține descrierea algoritmului de planificare folosit.