Artificial intelligence - Project 3
- Planning utilizand PDDL -

Andriescu Antoino

10/01/2021

1 Problema propusa

1.1 Descriere problema

Problema propusa consta in serviciul de transport de masini. Presupunem ca o reprezentanta auto din Romania are sediul in cateva orase. Sunt masini in toate sediile din toate orasele. Daca un client doreste sa cumpere o masina, iar aceasta se afla la sediul dintr-un alt oras, compania trebuie sa transporte masina la sediul din care s-a efectuat achizitia. Presupunand ca mai multe masini au fost achizitionate in orase diferite, acestea trebuie transportate in anumite orase. Compania are un singur camion care trebuie sa transporte masinile, iar acesta poate transporta o singura masina la un moment dat. Din acest motiv distanta pe care o parcurge camionul trebuie sa fie cat mai mica, ca transporturle sa se faca cat mai eficient.

Stim deja distantele dintre orase, unde este si unde trebuie sa ajunga fiecare masina si din ce oras pleaca camionul.

Masinile sunt numerotate de la 1 la 8, iar orasele in care compania are sediu sunt: Cluj-Napoca, Piatra-Neamt, Oradea, Iasi, Bucuresti, Timisoara si Constanta.

1.2 Starile initiale

Masina 1 este in Cluj-Napoca.

Masina 2 este in Piatra-Neamt.

Masina 3 este in Timisoara.

Masina 4 este in Piatra-Neamt.

Masina 5 este in Bucuresti.

Masina 6 este in Iasi.

Masina 7 este in Timisoara.

Masina 8 este in Bucuresti.

Camionul este in Piatra-Neamt.

1.3 Starile finale

Masina 1 trebuie sa ajunga in Bucuresti.

Masina 2 trebuie sa ajunga in Cluj-Napoca.

Masina 3 trebuie sa ajunga in Cluj-Napoca.

Masina 4 trebuie sa ajunga in Cluj-Napoca.

Masina 5 trebuie sa ajunga in Cluj-Napoca.

Masina 6 trebuie sa ajunga in Timisoara.

Masina 7 trebuie sa ajunga in Constanta.

Masina 8 trebuie sa ajunga in Iasi.

2 Solutia propusa

2.1 Domain

```
(define (domain transport)

        (:requirements :strips :action-costs :typing :negative-preconditions :equality)

        (:types
                car
                city
                truck
        )

        (:predicates
                (at ?obj - car ?loc - city)
                (in ?obj - car ?trk - truck)
                (incity ?trk - truck ?loc - city)
                (incarcat ?t - truck )
        )

        (:functions
                (DISTANTA ?loc1 - city ?loc2 - city) - number
                (total-cost) - number
        )

        (:action load-vehicle
                    :parameters (?obj - car ?trk - truck ?loc - city)
                    :precondition (and (incity ?trk ?loc) (at ?obj ?loc) (not (incarcat ?trk)))
                    :effect (and (not (at ?obj ?loc)) (in ?obj ?trk) (incarcat ?trk))
          )

          (:action unload-vehicle
                :parameters (?obj - car ?trk - truck ?loc - city)
                :precondition (and (incity ?trk ?loc) (in ?obj ?trk))
                :effect (and (not (in ?obj ?trk)) (at ?obj ?loc) (not (incarcat ?trk)))
        )

        (:action drive-vehicle
                :parameters (?trk - truck ?locfrom - city ?locto - city)
                :precondition (and (incity ?trk ?locfrom))
                :effect (and (not (incity ?trk ?locfrom)) (incity ?trk ?locto) (increase (total-cost) (

        )
)
```

In partea de domain am notat prima data tipurile de obiecte pe care le avem in problema: car, city si truck. Predicatele folosite sunt: "at" care indica masina si in ce oras este, "in" care indica daca masina este in camion, "incity" care indica in ce oras este camionul si "incarcat" care indica daca camionul este incarcat cu o masina sau nu. Am folosit functia "DISTANTA" pentru a declara distanta dintre doua orase si functia "total-cost" in care se salveaza distanta totala pe care a parcurs-o camionul. Aceasta din urma va fi initializata cu 0.

Actiunile pe care le poate face camionul sunt:

- load-vehicle: indica faptul ca masina a fost incarcata in camion si urmeaza sa fie transportata
- unload-vehicle: indica faptul ca masina este descarcata din camion in orasul in care trebuie sa ajunga
- drive-vehicle: indica faptul ca camionul se deplaseaza dintr-un oras in altul

2.2 Problem

```
(define (problem transport_masini)
        (:domain transport)

        (:objects
                car1 car2 car3 car4 car5 car6 car7 car8 - car
                cluj piatra oradea bucuresti timisoara iasi constanta - city
                truck - truck
        )

        (:init
                (incity truck piatra)

                (at car1 cluj)
                (at car2 piatra)
                (at car3 timisoara)
                (at car4 piatra)
                (at car5 bucuresti)
                (at car8 bucuresti)
                (at car7 timisoara)
                (at car6 iasi)

                (= (total-cost) 0)

                (= (DISTANTA cluj cluj) 10000)
                (= (DISTANTA cluj piatra) 327)
                (= (DISTANTA cluj oradea) 152)
                (= (DISTANTA cluj bucuresti) 440)
                (= (DISTANTA cluj timisoara) 334)
                (= (DISTANTA cluj iasi) 427)
                (= (DISTANTA cluj constanta) 706)

                (= (DISTANTA piatra piatra) 10000)
                (= (DISTANTA piatra cluj) 327)
                (= (DISTANTA piatra oradea) 479)
                (= (DISTANTA piatra bucuresti) 342)
                (= (DISTANTA piatra timisoara) 599)
                (= (DISTANTA piatra iasi) 142)
                (= (DISTANTA piatra constanta) 500)
```

```
39
40                  (= (DISTANTA oradea oradea) 10000)
41                  (= (DISTANTA oradea cluj) 152)
42                  (= (DISTANTA oradea piatra) 479)
43                  (= (DISTANTA oradea bucuresti) 592)
44                  (= (DISTANTA oradea timisoara) 168)
45                  (= (DISTANTA oradea iasi) 621)
46                  (= (DISTANTA oradea constanta) 858)
47
48                  (= (DISTANTA bucuresti bucuresti) 10000)
49                  (= (DISTANTA bucuresti cluj) 440)
50                  (= (DISTANTA bucuresti piatra) 342)
51                  (= (DISTANTA bucuresti oradea) 592)
52                  (= (DISTANTA bucuresti timisoara) 562)
53                  (= (DISTANTA bucuresti iasi) 393)
54                  (= (DISTANTA bucuresti constanta) 266)
55
56                  (= (DISTANTA timisoara timisoara) 10000)
57                  (= (DISTANTA timisoara cluj) 334)
58                  (= (DISTANTA timisoara piatra) 599)
59                  (= (DISTANTA timisoara oradea) 168)
60                  (= (DISTANTA timisoara bucuresti) 562)
61                  (= (DISTANTA timisoara iasi) 734)
62                  (= (DISTANTA timisoara constanta) 828)
63
64                  (= (DISTANTA iasi iasi) 10000)
65                  (= (DISTANTA iasi cluj) 427)
66                  (= (DISTANTA iasi piatra) 142)
67                  (= (DISTANTA iasi oradea) 621)
68                  (= (DISTANTA iasi bucuresti) 393)
69                  (= (DISTANTA iasi timisoara) 734)
70                  (= (DISTANTA iasi constanta) 430)
71
72                  (= (DISTANTA constanta constanta) 10000)
73                  (= (DISTANTA constanta cluj) 706)
74                  (= (DISTANTA constanta piatra) 500)
75                  (= (DISTANTA constanta oradea) 858)
76                  (= (DISTANTA constanta bucuresti) 266)
77                  (= (DISTANTA constanta timisoara) 828)
78                  (= (DISTANTA constanta iasi) 430)
79          )
80
81          (:goal (and (at car1 bucuresti) (at car2 cluj ) (at car3 cluj) (at car4 cluj) (at car5 cluj) (at
82
83
84          (:metric minimize (total-cost))
85  )
```

In partea de problem am introdus toate datele pe care ni le ofera problema. Cele opt masini le-am notat cu car1, car2, car3, ... , car 8 si le-am declarat de tipul "car", am notat cele sapte orase si le-am declarat de tipul "city" si camionul l-am declarat de tipul "truck"

In parte de ":init" am mentionat in ce oras este fiecare masina si camionul, am initializat costul total al drumului cu "0" si am scris distanta dintre fiecare doua orase.

La ":goal" am declarat locurile in care vrem sa ajunga fiecare masina

3 Rezultate experimentale

3.1 A* search + FF heuristic

```
(load-vehicle car2 truck piatra)
(drive-vehicle truck piatra cluj)
(unload-vehicle car2 truck cluj)
(drive-vehicle truck cluj oradea)
(drive-vehicle truck oradea timisoara)
(load-vehicle car7 truck timisoara)
(drive-vehicle truck timisoara constanta)
(unload-vehicle car7 truck constanta)
(drive-vehicle truck constanta bucuresti)
(load-vehicle car8 truck bucuresti)
(drive-vehicle truck bucuresti iasi)
(unload-vehicle car8 truck iasi)
(load-vehicle car6 truck iasi)
(drive-vehicle truck iasi timisoara)
(unload-vehicle car6 truck timisoara)
(load-vehicle car3 truck timisoara)
(drive-vehicle truck timisoara oradea)
(drive-vehicle truck oradea cluj)
(unload-vehicle car3 truck cluj)
(load-vehicle car1 truck cluj)
(drive-vehicle truck cluj bucuresti)
(unload-vehicle car1 truck bucuresti)
(load-vehicle car5 truck bucuresti)
(drive-vehicle truck bucuresti cluj)
(unload-vehicle car5 truck cluj)
(drive-vehicle truck cluj piatra)
(load-vehicle car4 truck piatra)
(drive-vehicle truck piatra cluj)
(unload-vehicle car4 truck cluj)
; cost = 4722 (general cost)
```

3.2 A* search + Additive heuristic

```
(load-vehicle car2 truck piatra)
(drive-vehicle truck piatra cluj)
(unload-vehicle car2 truck cluj)
(load-vehicle car1 truck cluj)
(drive-vehicle truck cluj bucuresti)
(unload-vehicle car1 truck bucuresti)
(load-vehicle car5 truck bucuresti)
(drive-vehicle truck bucuresti cluj)
(unload-vehicle car5 truck cluj)
(drive-vehicle truck cluj piatra)
(load-vehicle car4 truck piatra)
(drive-vehicle truck piatra cluj)
(unload-vehicle car4 truck cluj)
(drive-vehicle truck cluj oradea)
```

```
(drive-vehicle truck oradea timisoara)
(load-vehicle car3 truck timisoara)
(drive-vehicle truck timisoara cluj)
(unload-vehicle car3 truck cluj)
(drive-vehicle truck cluj bucuresti)
(load-vehicle car8 truck bucuresti)
(drive-vehicle truck bucuresti iasi)
(unload-vehicle car8 truck iasi)
(load-vehicle car6 truck iasi)
(drive-vehicle truck iasi timisoara)
(unload-vehicle car6 truck timisoara)
(load-vehicle car7 truck timisoara)
(drive-vehicle truck timisoara constanta)
(unload-vehicle car7 truck constanta)
; cost = 4910 (general cost)
```

3.3 A* search + Context-enhanced additive heuristic

```
(load-vehicle car2 truck piatra)
(drive-vehicle truck piatra cluj)
(unload-vehicle car2 truck cluj)
(load-vehicle car1 truck cluj)
(drive-vehicle truck cluj bucuresti)
(unload-vehicle car1 truck bucuresti)
(load-vehicle car8 truck bucuresti)
(drive-vehicle truck bucuresti iasi)
(unload-vehicle car8 truck iasi)
(drive-vehicle truck iasi piatra)
(load-vehicle car4 truck piatra)
(drive-vehicle truck piatra cluj)
(unload-vehicle car4 truck cluj)
(drive-vehicle truck cluj oradea)
(drive-vehicle truck oradea timisoara)
(load-vehicle car3 truck timisoara)
(drive-vehicle truck timisoara oradea)
(drive-vehicle truck oradea cluj)
(unload-vehicle car3 truck cluj)
(drive-vehicle truck cluj bucuresti)
(load-vehicle car5 truck bucuresti)
(drive-vehicle truck bucuresti cluj)
(unload-vehicle car5 truck cluj)
(drive-vehicle truck cluj oradea)
(drive-vehicle truck oradea timisoara)
(load-vehicle car7 truck timisoara)
(drive-vehicle truck timisoara constanta)
(unload-vehicle car7 truck constanta)
(drive-vehicle truck constanta iasi)
(load-vehicle car6 truck iasi)
(drive-vehicle truck iasi timisoara)
(unload-vehicle car6 truck timisoara)
; cost = 5461 (general cost)
```

3.4 Weighted A* search + FF heuristic

```
(load-vehicle car2 truck piatra)
(drive-vehicle truck piatra cluj)
(unload-vehicle car2 truck cluj)
(load-vehicle car1 truck cluj)
(drive-vehicle truck cluj bucuresti)
(unload-vehicle car1 truck bucuresti)
(load-vehicle car5 truck bucuresti)
(drive-vehicle truck bucuresti cluj)
(unload-vehicle car5 truck cluj)
(drive-vehicle truck cluj piatra)
(load-vehicle car4 truck piatra)
(drive-vehicle truck piatra cluj)
(unload-vehicle car4 truck cluj)
(drive-vehicle truck cluj timisoara)
(load-vehicle car3 truck timisoara)
(drive-vehicle truck timisoara cluj)
(unload-vehicle car3 truck cluj)
(drive-vehicle truck cluj bucuresti)
(load-vehicle car8 truck bucuresti)
(drive-vehicle truck bucuresti iasi)
(unload-vehicle car8 truck iasi)
(load-vehicle car6 truck iasi)
(drive-vehicle truck iasi timisoara)
(unload-vehicle car6 truck timisoara)
(load-vehicle car7 truck timisoara)
(drive-vehicle truck timisoara constanta)
(unload-vehicle car7 truck constanta)
; cost = 4924 (general cost)
```

3.5 Lazy Greedy search + FF heuristic

```
(load-vehicle car2 truck piatra)
(drive-vehicle truck piatra bucuresti)
(unload-vehicle car2 truck bucuresti)
(drive-vehicle truck bucuresti piatra)
(load-vehicle car4 truck piatra)
(drive-vehicle truck piatra bucuresti)
(drive-vehicle truck bucuresti cluj)
(unload-vehicle car4 truck cluj)
(drive-vehicle truck cluj bucuresti)
(load-vehicle car2 truck bucuresti)
(drive-vehicle truck bucuresti cluj)
(unload-vehicle car2 truck cluj)
(drive-vehicle truck cluj bucuresti)
(load-vehicle car5 truck bucuresti)
(drive-vehicle truck bucuresti cluj)
(unload-vehicle car5 truck cluj)
(load-vehicle car1 truck cluj)
(drive-vehicle truck cluj bucuresti)
(unload-vehicle car1 truck bucuresti)
(load-vehicle car8 truck bucuresti)
(drive-vehicle truck bucuresti cluj)
(drive-vehicle truck cluj iasi)
```

```
(unload-vehicle car8 truck iasi)
(load-vehicle car6 truck iasi)
(drive-vehicle truck iasi cluj)
(drive-vehicle truck cluj oradea)
(drive-vehicle truck oradea timisoara)
(unload-vehicle car6 truck timisoara)
(load-vehicle car3 truck timisoara)
(drive-vehicle truck timisoara oradea)
(drive-vehicle truck oradea cluj)
(unload-vehicle car3 truck cluj)
(drive-vehicle truck cluj constanta)
(drive-vehicle truck constanta timisoara)
(load-vehicle car7 truck timisoara)
(drive-vehicle truck timisoara constanta)
(unload-vehicle car7 truck constanta)
; cost = 7962 (general cost)
```

3.6 Rezultate

In urma testarii programului cu diferiti algoritmi de cautare si diferite euristici am observat ca drumul cel mai eficient a fost gasit de A* cu euristica FF, dar acesta a gasit si cel mai greu solutia (25 secunde). Algoritmul de cautare A* a fost mult mai rapid impreuna cu euristicile Additive si Context-enhanced additive, programul gasind solutiile in mai putin de 1 secunda, insa distantele finale sunt mai mari. In cazul euristicii Context-enhanced additive diferenta este mult mai vizibila, distanta crescand cu 739 km.

La compilarea programului cu algoritmul Weighted A* si euristica FF avem un cost total apropiat de A* cu euristica Additive, dar fata de toate celelalte teste acesta a fost pe departe cel mai rapid, gasind solutia in aproximativ 0.006 secunde. Aceasta combinatie ar putea fi utila in situatia in care am avea un numar mare de masini si de orase, unde complexitatea gasirii unei solutii este mult mai mare.

Ultimul test a fost realizat cu algoritmul Lazy Greedy care, desi a fost rapid in gasirea solutiei, a gasit cea mai neeficienta solutie, cu un cost total de 7962. Cu o diferenta de 3240 km fata de cel mai eficient algoritm, nu il pot recomanda pentru problema propusa deoarece alte combnatii au reusit sa gaseasca in mai putin timp o solutie mult mai eficienta.

4 Concluzie

In urma testelor efectuate asupra programului consider ca cel mai eficient din punct de vedere al costului si al timpului este algoritmul Wighted A* cu euristica FF deoarece a reusit sa gaseasca o solutie cu o distanta mica si intr-un timp cat mai scurt. Desi primul test a fost cel mai eficient, nu este justificat de cele 25 de secunde necesare pentru aflarea solutiei.