Inteligenta Artificiala - Proiectul 3
-Planning utilizand PDDL -

Malaescu Bianca

9/1/2022

PDDL

1 Livrare comenzi de mancare

1.1 Descrierea problemei

Obiectul problemei il reprezinta serviciul de livrare comenzi al unui restaurant.Acesta dispune de livratori pe bicicleta cat si de livratori cu masina, a caror responsabilitate este sa livreze comenzile care apar.

Exista cateva conditii pentru fiecare livrare: :

Livratorii pe bicicleta nu pot face livrari pe ploaie si nu pot transporta comenzi mari.

Livratorii cu masina pot face livrari doar daca au destul combustibil. Daca raman fara combustibil ei trebuie sa alimenteze.

Actiunile de livare au urmatoarele costuri: drumurile livratorilor cu bicicleta sunt platite cu 7 lei, livrarile cu masina cu cu 9 lei, iar alimentarea masinilor necesita 8 lei, . Solutia ideala a problemei are costul cel mai mic.

In problema propusa regasim 5 comenzi care trebuie livrate. 4 dintre ele se afla la restaurant, dar una dintre ele se afla la o alta adresa. Aceasta era in curs de transport, dar livratorul a facut pana, deci necesita sa fie preluata de un alt livrator. 3 dintre aceste comenzi sunt mari, celelalte fiind mici. Disponbili sunt 2 livratori cu masini si unul pe biciccleta. Unul dintre soferi nu mai are combustibil suficient. Soferul fara benzina se afla la restaurant, iar ceilalti 2 livratori se afla la o alta adresa unde au lasat comenzi precedente. Vremea nu este una ploioasa. Avand in vedere costurile de transport si alimentarea pentru masini, se cere sa se gaseasca solutia optima a problemei - care are costul cel mai mic si un timp de rezolvare scurt.

1.2 Rezolvare in PDDL

1.3 Domeniu

Cod:

```
(define (domain livrare-mancare)

(:requirements :strips :action-costs)

(:types comanda vehicul - object
                sofer biciclist - vehicul
                locatie

)

(:functions
(total-cost) - number
)


```

```
(:predicates
(la-locatie ?obj - object ?loc - locatie)
(in-vehicul ?coma - comanda ?sof - vehicul)
(comanda-mare ?com - comanda)
(fara-benzina ?sof - sofer)
(ploua)
)



(:action preia_comanda_soferul
:parameters (?sof - sofer ?com - comanda ?loc - locatie)
:precondition (and (la-locatie ?sof ?loc)
(la-locatie ?com ?loc))
:effect (and (not (la-locatie ?com ?loc))
(in-vehicul ?com ?sof))
)

(:action preia_comanda_biciclistul
:parameters (?sof - biciclist ?com - comanda ?loc - locatie)
:precondition (and (la-locatie ?sof ?loc)
(la-locatie ?com ?loc)
(not(comanda-mare ?com))
)
:effect (and (not (la-locatie ?com ?loc))
(in-vehicul ?com ?sof))
)


(:action livreaza_comanda
:parameters (?sof - vehicul ?com - comanda ?loc - locatie)
:precondition (and (la-locatie ?sof ?loc)
(in-vehicul ?com ?sof))
:effect (and(not (in-vehicul ?com ?sof))(la-locatie ?com ?loc))
)

(:action conduce-masina
:parameters (?sof - sofer ?loc1 - locatie ?loc2 - locatie)
:precondition (and(la-locatie ?sof ?loc1) (not(fara-benzina ?sof)))
:effect (and (la-locatie ?sof ?loc2) (not(la-locatie ?sof ?loc1)) (increase (total-cost) 9))
)

(:action conduce-bicicleta
:parameters (?sof - biciclist ?loc1 - locatie ?loc2 - locatie)
:precondition (and(la-locatie ?sof ?loc1)(not(ploua)))
:effect (and (la-locatie ?sof ?loc2) (not(la-locatie ?sof ?loc1)) (increase (total-cost) 7))
)

(:action alimenteaza
:parameters (?sof - sofer)
:precondition (fara-benzina ?sof)
:effect (and(not(fara-benzina ?sof ))(increase (total-cost) 8))
)

```

```
72
73 )
```

Explicatie:

Am declarat type-urile problemei pentru a fi mai usoara implementarea actiunilor. Predicatele reprezinta proprietati ale type-urilor, in functie de tipul acestora (ex: faptul ca o comanda este mare e o proprietate a obiectelor de tip "comanda").

Urmatoarea parte de cod reprezinta actiunile, descrise fiecare prin parametrii, preconditiile si efectele lor. Acestea au nume sugestive pentru a putea fi intelese usor. Actiunile descrise in cod se refera la: preluarea comenzilor de catre livratori pe bicicleta/cu masina, conducerea vehiculelor, livrarea comenzilor si alimentarea masinilor.

1.4 Problema

```
1  (define (problem comenzi)
2
3  (:domain livrare-mancare)
4  (:objects
5       comanda123 - comanda
6       comanda45 - comanda
7       comanda66 - comanda
8       comanda12 - comanda
9       comanda3 - comanda
10
11      bic1 - biciclist
12      sofer2 - sofer
13      sofer3 - sofer
14
15      restaurant - locatie
16      dorobantilor - locatie
17      mihali - locatie
18      baritiu - locatie
19     bucuresti - locatie
20
21 )
22
23 (:init
24 (not(ploua))
25 (= (total-cost) 0)
26
27 (not(comanda-mare comanda123))
28 (not(comanda-mare comanda66))
29 (comanda-mare comanda45)
30 (comanda-mare comanda12)
31 (comanda-mare comanda3)
32
33 (la-locatie comanda123 baritiu)
34 (la-locatie comanda45 restaurant)
35 (la-locatie comanda12 restaurant)
36 (la-locatie comanda66 restaurant)
37 (la-locatie comanda3 restaurant)
38
39 (la-locatie sofer2 restaurant)
```

```
(la-locatie sofer3 mihali)
(la-locatie bic1 mihali)
(fara-benzina sofer2)
(not(fara-benzina sofer3))


)

(:goal
(and(la-locatie comanda123 dorobantilor)(la-locatie comanda45 mihali)(la-locatie comanda66 dorobantilor)
)

(:metric minimize (total-cost))

)
```

Explicatie:

In prima parte a codului problemei am declarat obiectele: 5 comenzi, un livrator pe bicicleta, 2 livratori cu masini si 5 locatii din oras.

In partea de initializare am setat costul total la 0, dupa care am stabilit starea vremii (nu ploua) si am scris predicatele pentru toate obiectele declarate mai sus. Pentru fiecare comanda am detaliat daca este mare sau nu (cele mari nu pot fi livrate pe bicicleta) si i-am precizat locatia. Am continuat cu locatia fiecarui livrator, iar pentru livratorii care conduc masinile restaurantului am precizat daca mai au combustibil sau nu.

In goal am precizat ca fiecare comanda sa ajunga la locatia de livrare, iar ultima linie reprezinta minimizarea costului total al livrarii comenzilor.

1.5 Rezultate experimentale - rulare cu diferite euristici si algoritmi de cautare

Algoritm de cautare: Astar, Euristica: FF
T : 0.0176 s

```
(alimenteaza sofer2)
(conduce-bicicleta bic1 mihali baritiu)
(preia_comanda_soferul sofer2 comanda66 restaurant)
(preia_comanda_soferul sofer2 comanda45 restaurant)
(preia_comanda_soferul sofer2 comanda3 restaurant)
(preia_comanda_soferul sofer2 comanda12 restaurant)
(conduce-masina sofer2 restaurant bucuresti)
(livreaza_comanda sofer2 comanda12 bucuresti)
(conduce-masina sofer2 bucuresti dorobantilor)
(preia_comanda_biciclistul bic1 comanda123 baritiu)
(conduce-bicicleta bic1 baritiu dorobantilor)
(livreaza_comanda sofer2 comanda66 dorobantilor)
(livreaza_comanda sofer2 comanda3 dorobantilor)
(conduce-masina sofer2 dorobantilor mihali)
(livreaza_comanda bic1 comanda123 dorobantilor)
(livreaza_comanda sofer2 comanda45 mihali)
; cost = 49 (general cost)
```

Algoritm de cautare: Astar, Euristica: Additive
T : 0.0285 s

```
(alimenteaza sofer2)
(conduce-bicicleta bic1 mihali dorobantilor)
(preia_comanda_soferul sofer2 comanda66 restaurant)
(preia_comanda_soferul sofer2 comanda45 restaurant)
(preia_comanda_soferul sofer2 comanda3 restaurant)
(preia_comanda_soferul sofer2 comanda12 restaurant)
(conduce-masina sofer2 restaurant dorobantilor)
(livreaza_comanda sofer2 comanda66 dorobantilor)
(livreaza_comanda sofer2 comanda3 dorobantilor)
(conduce-masina sofer2 dorobantilor bucuresti)
(livreaza_comanda sofer2 comanda12 bucuresti)
(conduce-masina sofer2 bucuresti mihali)
(conduce-bicicleta bic1 dorobantilor baritiu)
(preia_comanda_biciclistul bic1 comanda123 baritiu)
(conduce-bicicleta bic1 baritiu dorobantilor)
(livreaza_comanda bic1 comanda123 dorobantilor)
(livreaza_comanda sofer2 comanda45 mihali)
; cost = 56 (general cost)
```

Algoritm de cautare: Astar, Euristica: Blind
T : 10.19637 s

```
(preia_comanda_soferul sofer2 comanda66 restaurant)
(preia_comanda_soferul sofer2 comanda45 restaurant)
(preia_comanda_soferul sofer2 comanda3 restaurant)
(preia_comanda_soferul sofer2 comanda12 restaurant)
(alimenteaza sofer2)
(conduce-masina sofer2 restaurant baritiu)
(preia_comanda_soferul sofer2 comanda123 baritiu)
(conduce-masina sofer2 baritiu dorobantilor)
(livreaza_comanda sofer2 comanda66 dorobantilor)
(livreaza_comanda sofer2 comanda3 dorobantilor)
(livreaza_comanda sofer2 comanda123 dorobantilor)
(conduce-masina sofer2 dorobantilor bucuresti)
(livreaza_comanda sofer2 comanda12 bucuresti)
(conduce-masina sofer2 bucuresti mihali)
(livreaza_comanda sofer2 comanda45 mihali)
; cost = 44 (general cost)
```

Algoritm de cautare: Astar, Euristica: Context-Enhanced additive
T : 0.0083 s

```
(alimenteaza sofer2)
(conduce-bicicleta bic1 mihali baritiu)
(preia_comanda_biciclistul bic1 comanda123 baritiu)
(conduce-bicicleta bic1 baritiu dorobantilor)
(preia_comanda_soferul sofer2 comanda66 restaurant)
(preia_comanda_soferul sofer2 comanda45 restaurant)
(preia_comanda_soferul sofer2 comanda3 restaurant)
(conduce-masina sofer2 restaurant dorobantilor)
(conduce-masina sofer3 mihali restaurant)
(preia_comanda_soferul sofer3 comanda12 restaurant)
(conduce-masina sofer3 restaurant bucuresti)
(livreaza_comanda sofer2 comanda66 dorobantilor)
(livreaza_comanda sofer2 comanda3 dorobantilor)
(conduce-masina sofer2 dorobantilor mihali)
```

```
(livreaza_comanda sofer3 comanda12 bucuresti)
(livreaza_comanda bic1 comanda123 dorobantilor)
(livreaza_comanda sofer2 comanda45 mihali)
; cost = 58 (general cost)
```

Algoritm de cautare: Laza Greedy, Euristica:FF
T : 0.0067 s

```
(conduce-masina sofer3 mihali restaurant)
(conduce-bicicleta bic1 mihali baritiu)
(preia_comanda_soferul sofer3 comanda66 restaurant)
(preia_comanda_soferul sofer3 comanda3 restaurant)
(conduce-masina sofer3 restaurant dorobantilor)
(alimenteaza sofer2)
(preia_comanda_biciclistul bic1 comanda123 baritiu)
(conduce-bicicleta bic1 baritiu dorobantilor)
(preia_comanda_soferul sofer2 comanda45 restaurant)
(preia_comanda_soferul sofer2 comanda12 restaurant)
(conduce-masina sofer2 restaurant bucuresti)
(livreaza_comanda sofer2 comanda12 bucuresti)
(conduce-masina sofer2 bucuresti mihali)
(livreaza_comanda sofer3 comanda66 dorobantilor)
(livreaza_comanda sofer3 comanda3 dorobantilor)
(livreaza_comanda bic1 comanda123 dorobantilor)
(livreaza_comanda sofer2 comanda45 mihali)
; cost = 58 (general cost)
```

In urma rularii cu diferite euristici si algoritmi de cautare, solutia cea mai eficienta din punctul de vedere al costului e cea gasita folosind algoritmul A-star impreuna cu euristica "Blind". Din pacate, este si cea mai rea varianta avand in vedere timpul. Cele 10 secunde reprezinta timpul necesar pentu a rula de 1492 de ori programul folosind algoritmul de cautare Greedy si euristica FF. Aceasta solutie este cea mai rapida insa si cea mai proasta din punct de vedere al costului.

Acelasi cost maxim (58) l-am obtinund ruland A-star si euristica Context-Enhanced Additive. La mijloc din punct de vedere al timpului, cat si din punct de vedere al costului problemei se afla solutia obtinuta ruland A-star si euristia Additive

In concluzie, consider ca solutia cea mai buna din toate punctele de vedere o obtinem ruland A-star si euristica FF, cu o durata de 0.00176 secunde si un cost de 49. Solutia cea mai buna din punct de vedere al costului nu se justifica, ajungand la un timp foarte mare de 10 secunde.