Artificial intelligence - Project 3
- Limbaje de planificare -

Dorofte Andrei, Iacob Liviu

12/01/2021

1 Problem description

- Problema este construita cu o firma de curierat in minte. Zona pe care actioneaza aceasta firma este reprezentata sub forma de matrice. Scopul fiecarei masini este de a livra pachetul la destinatie si de a ajunge mai apoi acasa. Firma poate avea mai multe masini. De asemenea, pot fi mai multe pachete de livrat la mai multe destinatii. Politica firmei nu specifica faptul ca un pachet trebuie dus doar de o singura masina, drept urmare, o masina poate lua un pachet si sa il lase pe drum, urmand sa fie ridicat de alta masina.
- Fiecare sofer trebuie pe parcursul traseului sa manance ca sa aiba energie pentru livra comenzile, iar mai apoi trebuie sa alimenteze masina pe care o conduce. Fiecare sofer are o singura masina pe care o conduce si fiecare sofer poate manca doar o sigura data in timpul programului. Pentru a alimenta, acesta trebuie sa manance inainte.
- Pentru a ridica un pachet, soferul trebuie sa fie liber si sa ajunga la locul unde se afla pachetul. Faptul ca soferul este liber inseamna ca nu are niciun pachet deja la el. Soferul nu este conditionat de mancare sau de alimentarea masinii pentru a ridica pachetul.
- Pentru a lasa pachetul la destinatie, soferul trebuie sa ajunga la destinatie si sa aiba coletul la el. Dupa ce lasa coletul, acesta devine liber si poate ridica alt pachet. Totodata, dupa ce lasa un pachet, incepe sa fie vitezoman, dorind sa ajunga cat mai repede acasa.
- Pe traseu se mai afla camere. Daca un sofer face toate cele 4 miscari posibile (sus, jos, stanga, dreapta) pe parcursul traseului si acesta este vitezoman, asta inseamana ca soferul conduce agresiv si va primi o amenda. Dupa primirea amenzii, acesta va conduce mai prudent, fiind imposibil sa mai primeasca inca o amenda.
- Fiecare actiune are un cost, excluzand amenda, care este inevitabila daca soferul indeplineste conditiile sanctiunii. Scopul este de a rezolva problema prin a livra toate pachetele la destinatie.

2 Code description

- Domeniul problemei este definit prin predicate, actiuni si functii.
  Predicatele folosite sunt:
- IS-CAR ?x -> este adevarat daca obiectul x este masina este masina.
- IS-ROW ?x -> este adevarat daca obiectul x este rand din matrice.
- IS-COLUMN ?x -> este adevarat daca obiectul x este coloana din matrice.
- NEXT-ROW ?r1 ?r2 -> este adevarat daca obiectul r2 este randul urmator pentru r1.
- NEXT-COLUMN ?c1 ?c2 -> este adevarat daca obiectul c2 este coloana urmatoare pentru c1.
- car-at ?x ?r ?c -> este adevarat daca obiectul x reprezentand masina se afla la pozitia r (rand) si c (coloana).
- is-package ?p -> este adevarat daca obiectul p este pachet.
- has-package ?c ?p -> este adevarat daca masina c are pachetul p asupra lui.
- empty ?c -> este adevarat daca masina c este libera (nu are niciun pachet asupra lui).
- package-at ?p ?row ?col -> este adevarat daca pachetul p se afla la locatia row si col.
- is-fined ?c -> este adevarat daca masina c a luat amenda.
- is-speedster ?c -> este adevarat daca soferul de pe masina c este vitezoman.
- is-fueled ?c -> este adevarat daca masina c este alimentata.
- has-eaten ?c -> este adevarat daca soferul masinii c a mancat.
- rest-at ?row ?col -> este adevarat daca exista un restaurant la pozitia row si col.
- camera-at ?row ?col -> este adevarat daca exista camera RADAR la pozitia row si col.
- gas-station ?row ?col -> este adevarat daca exista o benzinarie la pozitia row si col.
- moved-up ?c -> este adevarat daca masina c s-a miscat in sus pe parcursul traseului.
- moved-down ?c -> este adevarat daca masina c s-a miscat in jos pe parcursul traseului.
- moved-left ?c -> este adevarat daca masina c s-a miscat in stanga pe parcursul traseului.
- moved-right ?c -> este adevarat daca masina c s-a miscat in dreapta pe parcursul traseului.
  Actiunile folosite sunt:
- getPackage
  Parametri: ?car ?p ?row ?col
  Preconditii:
  ?car sa fie masina
  ?row sa fie rand
  ?col sa fie coloana
  ?car sa fie pe pozitia ?row ?col
  ?p este pachet
  ?p este pe pozitia ?row ?col

?car e goala, adica nu are

Efect:

?p nu se mai afla la pozitia ?row ?col

?car are pachetul ?p la bord

?car nu mai e libera

creste costul cu 20

- dropPackage

  Parametri: ?car ?p ?row ?col

  Preconditii:

  ?car sa fie masina

  ?row sa fie rand

  ?col sa fie coloana

  ?car sa fie pe pozitia ?row ?col

  ?car sa aiba pachetul ?p la bord

  ?car sa nu fie disponibila

  Efect:

  ?p se afla la pozitia ?row ?col

  ?car nu mai are pachetul ?p la bord

  ?car e disponibila

  soferul masinii ?car devine vitezoman

  creste costul cu 20

- caughtOnCamera

  Parametri: ?car ?row ?col

  Preconditii:

  ?car sa fie masina

  ?row sa fie rand

  ?col sa fie coloana

  ?car sa fie pe pozitia ?row ?col

  camera RADAR sa fie pe pozitia ?row ?col

  ?car sa nu fie amendata deja

  soferul masinii ?car sa fie vitezoman

  ?car sa se fi miscat in toate cele 4 directii posibile

  Efect:

  ?car este amendata

  soferul masinii ?car nu mai e vitezoman si conduce prdent

  Acesta actiune nu are cost.

- eatSomething

  Parametri: ?car ?row ?col

  Preconditii:

  ?car sa fie masina

?row sa fie rand

?col sa fie coloana

?car sa fie pe pozitia ?row ?col

sa fie restaurant pe pozitia ?row ?col

?car sa nu fi mancat deja

Efect:

?car a mancat

costul acestei actiuni este 0

restaurantul de pe pozitia ?row ?col a ramas fara mancare

- fuelSomeGas

  Parametri: ?car ?row ?col

  Preconditii:

  ?car sa fie masina

  ?row sa fie rand

  ?col sa fie coloana

  ?car sa fie pe pozitia ?row ?col

  sa fie benzinarie pe pozitia ?row ?col

  ?car sa ni fi fost alimentata deja

  soferul masinii ?car trebuie sa fie mancat

  ?car trebuie sa aiba un pachet la bord

  Efect:

  ?car este alimentata

  Aceasta actiune are costul 0

  Benzinaria de pe pozitia ?row ?col a ramas fara carburant

- Si actiunile de miscare, care sunt move-up, move-down, move-left, move-right.

  Acestea sunt foarte similare, diferenta fiind doar preconditia NEXT-ROW sau NEXT-COLUMN, parametrul old-row si new-row sau old-column si new-column si setarea predicatului moved-up, moved-down, moved-left, respectiv moved-left pentru fiecare actiune. Un exemplu pentru una dintre ele ar fi:

  Parametri: ?car ?old-row ?new-row ?col

  Preconditii:

  ?car sa fie masina

  ?old-row sa fie rand

  ?new-row sa fie rand

  ?col sa fie coloana

  ?new-row sa fie urmatorul rand pentru ?new-row

  ?car sa fie pe pozitia ?old-row ?col soferul masinii ?car sa NU fie vitezoman sau sa NU fie camera RADAR pe pozitia

  ?old-row ?col

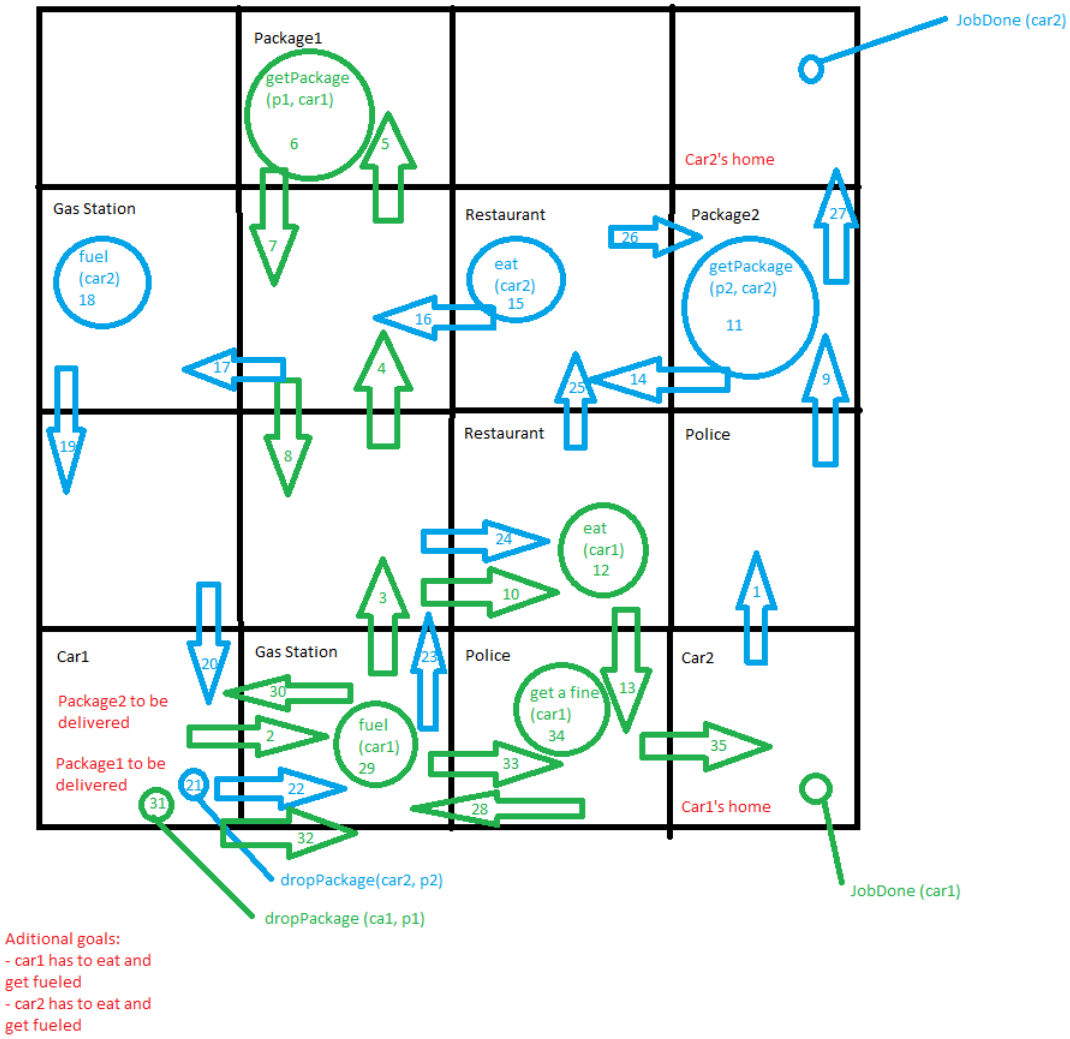
  Efect:

  ?car nu se mai afla la pozitia ?old-row ?col

  ?car este la pozitia ?new-row ?col

  se activeaza flag-ul pentru miscarea in aceasta directie costul acestei actiuni este de 20

- Poza cu descrierea vizuala a problemei 1:



Problem1 code:

```
;; Problema

(define (problem pe-ntruca)
    (:domain trip-to-romania)

    (:objects
     car1 car2
     p1 p2
     row1 row2 row3 row4
     col1 col2 col3 col4)

    (:init
```

```
        (IS-CAR car1)
        (IS-CAR car2)
        (IS-ROW row1)
        (IS-ROW row2)
        (IS-ROW row3)
        (IS-ROW row4)
        (NEXT-ROW row1 row2)
        (NEXT-ROW row2 row3)
        (NEXT-ROW row3 row4)
        (IS-COLUMN col1)
        (IS-COLUMN col2)
        (IS-COLUMN col3)
        (IS-COLUMN col4)
        (NEXT-COLUMN col1 col2)
        (NEXT-COLUMN col2 col3)
        (NEXT-COLUMN col3 col4)

        (is-package p1)
        (is-package p2)
        (package-at p1 row1 col2)
        (package-at p2 row2 col4)
        (empty car1)
        (empty car2)

        ;;initial state
        (car-at car1 row4 col1)
        (car-at car2 row4 col4)

        (rest-at row3 col3)
        (rest-at row2 col3)
        (not (has-eaten car1))
        (not (has-eaten car2))

        (gas-station row2 col1)
        (gas-station row4 col2)
        (not (is-fueled car1))
        (not (is-fueled car2))

        (not (is-fined car1))
        (not (moved-up car1))
        (not (moved-down car1))
        (not (moved-left car1))
        (not (moved-right car1))
        (camera-at row3 col4)

        (not (is-fined car2))
        (not (moved-up car2))
        (not (moved-down car2))
        (not (moved-left car2))
        (not (moved-right car2))
        (camera-at row4 col3)

        (= (total-cost) 0)
        (= (cost car1) 0)
```

```
67        (= (cost car2) 0)
68       )
69
70       (:goal
71        (and
72         (car-at car1 row4 col4)
73         (car-at car2 row1 col4)
74         (has-eaten car1)
75         (has-eaten car2)
76         (is-fueled car1)
77         (is-fueled car2)
78         ;;(is-fined car2)
79         ;;(is-fined car1)
80         (package-at p1 row4 col1)
81         (package-at p2 row4 col1)
82        )
83       )
84
85       (:metric minimize (total-cost))
86       ;;(:metric minimize (cost car1))
87       ;;(:metric minimize (cost car2))
88   )
```

Pentru rularea primei probleme, am rulat cu 2 euristici: ff si hgc. Ca algoritm de cautare am folosit atat astar cat si greedy. Traseul pentru astar cu hgc este mai mare cu 20, ceea ce inseamna ca se face o mutare in plus fata de ff. Pentru greedy, hgc are un scor mai mare cu 80, ceea ce inseamna 4 mutari in plus. Algoritmul de cautare astar returneaza o solutie cu un cost mult mai mic fata de greddy, folosin aceeasi euristica (ff: A* 580, greedy 720, hgc: A* 600, greedy 800)

```
[t=4.49317s, 17564 KB] g=580, 161201 evaluated, 47429 expanded, 305 reopened
[t=4.49331s, 17564 KB] New best heuristic value for ff: 0
[t=4.49334s, 17564 KB] g=600, 161209 evaluated, 47431 expanded, 305 reopened
[t=4.4934s, 17564 KB] Solution found!
[t=4.49344s, 17564 KB] Actual search time: 4.48066s
move-up car2 row4 row3 col4 (20)
move-right car1 row4 col1 col2 (20)
move-up car1 row4 row3 col2 (20)
move-up car1 row3 row2 col2 (20)
move-up car1 row2 row1 col2 (20)
getpackage car1 p1 row1 col2 (20)
move-down car1 row1 row2 col2 (20)
move-down car1 row2 row3 col2 (20)
move-up car2 row3 row2 col4 (20)
move-right car1 row3 col2 col3 (20)
getpackage car2 p2 row2 col4 (20)
eatsomething car1 row3 col3 (0)
move-down car1 row3 row4 col3 (20)
move-left car2 row2 col4 col3 (20)
eatsomething car2 row2 col3 (0)
move-left car2 row2 col3 col2 (20)
move-left car2 row2 col2 col1 (20)
fuelsomegas car2 row2 col1 (0)
move-down car2 row2 row3 col1 (20)
move-down car2 row3 row4 col1 (20)
droppackage car2 p2 row4 col1 (20)
move-right car2 row4 col1 col2 (20)
move-up car2 row4 row3 col2 (20)
move-right car2 row3 col2 col3 (20)
move-up car2 row3 row2 col3 (20)
move-right car2 row2 col3 col4 (20)
move-up car2 row2 row1 col4 (20)
move-left car1 row4 col3 col2 (20)
fuelsomegas car1 row4 col2 (0)
move-left car1 row4 col2 col1 (20)
droppackage car1 p1 row4 col1 (20)
move-right car1 row4 col1 col2 (20)
move-right car1 row4 col2 col3 (20)
caughtoncamera car1 row4 col3 (0)
move-right car1 row4 col3 col4 (20)
[t=4.49346s, 17564 KB] Plan length: 35 step(s).
[t=4.49346s, 17564 KB] Plan cost: 600
[t=4.49346s, 17564 KB] Expanded 47432 state(s).
[t=4.49346s, 17564 KB] Reopened 305 state(s).
[t=4.49346s, 17564 KB] Evaluated 161212 state(s).
[t=4.49346s, 17564 KB] Evaluations: 161212
[t=4.49346s, 17564 KB] Generated 491713 state(s).
[t=4.49346s, 17564 KB] Dead ends: 4028 state(s).
[t=4.49346s, 17564 KB] Expanded until last jump: 47422 state(s).
```

Problem2 code:

```
;; Problema 2

(define (problem pe-ntruca2)
    (:domain trip-to-romania)

    (:objects
     car1 car2
     p1 p2 p3
     row1 row2 row3 row4
     col1 col2 col3 col4)

    (:init
     (IS-CAR car1)
     (IS-CAR car2)
     (IS-ROW row1)
     (IS-ROW row2)
     (IS-ROW row3)
     (IS-ROW row4)
```

```
        (NEXT-ROW row1 row2)
        (NEXT-ROW row2 row3)
        (NEXT-ROW row3 row4)
        (IS-COLUMN col1)
        (IS-COLUMN col2)
        (IS-COLUMN col3)
        (IS-COLUMN col4)
        (NEXT-COLUMN col1 col2)
        (NEXT-COLUMN col2 col3)
        (NEXT-COLUMN col3 col4)

        (is-package p1)
        (is-package p2)
        (is-package p3)
        (package-at p1 row1 col2)
        (package-at p2 row2 col4)
        (package-at p3 row3 col3)
        (empty car1)
        (empty car2)

        ;;initial state
        (car-at car1 row2 col1)
        (car-at car2 row3 col4)

        (rest-at row3 col2)
        (rest-at row4 col1)
        (not (has-eaten car1))
        (not (has-eaten car2))

        (gas-station row1 col1)
        (gas-station row4 col1)
        (not (is-fueled car1))
        (not (is-fueled car2))

        (not (is-fined car1))
        (not (moved-up car1))
        (not (moved-down car1))
        (not (moved-left car1))
        (not (moved-right car1))
        (camera-at row1 col2)

        (not (is-fined car2))
        (not (moved-up car2))
        (not (moved-down car2))
        (not (moved-left car2))
        (not (moved-right car2))
        (camera-at row4 col2)

        (= (total-cost) 0)
        (= (cost car1) 0)
        (= (cost car2) 0)
      )

      (:goal
```

```
      (and
       (car-at car1 row4 col4)
       (car-at car2 row1 col4)
       (has-eaten car1)
       (has-eaten car2)
       (is-fueled car1)
       (is-fueled car2)
       ;;(is-fined car2)
       ;;(is-fined car1)
       (package-at p1 row1 col1)
       (package-at p2 row4 col1)
       (package-at p3 row4 col4)
      )
     )

     (:metric minimize (total-cost))
     ;;(:metric minimize (cost car1))
     ;;(:metric minimize (cost car2))
)
```

```
[t=2.10335s, 13724 KB] g=580, 59069 evaluated, 14700 expanded, 277 reopened
[t=2.10338s, 13724 KB] Solution found!
[t=2.10342s, 13724 KB] Actual search time: 2.09162s
move-left car2 row3 col4 col3 (20)
move-left car2 row3 col3 col2 (20)
eatsomething car2 row3 col2 (0)
move-right car1 row2 col1 col2 (20)
move-up car2 row3 row2 col2 (20)
move-up car2 row2 row1 col2 (20)
move-right car1 row2 col2 col3 (20)
getpackage car2 p1 row1 col2 (20)
move-right car1 row2 col3 col4 (20)
getpackage car1 p2 row2 col4 (20)
move-down car1 row2 row3 col4 (20)
move-down car1 row3 row4 col4 (20)
move-left car1 row4 col4 col3 (20)
move-left car1 row4 col3 col2 (20)
move-left car1 row4 col2 col1 (20)
eatsomething car1 row4 col1 (0)
fuelsomegas car1 row4 col1 (0)
move-left car2 row1 col2 col1 (20)
droppackage car1 p2 row4 col1 (20)
move-up car1 row4 row3 col1 (20)
move-right car1 row3 col1 col2 (20)
move-right car1 row3 col2 col3 (20)
getpackage car1 p3 row3 col3 (20)
move-down car1 row3 row4 col3 (20)
move-right car1 row4 col3 col4 (20)
droppackage car1 p3 row4 col4 (20)
fuelsomegas car2 row1 col1 (0)
droppackage car2 p1 row1 col1 (20)
move-down car2 row1 row2 col1 (20)
move-right car2 row2 col1 col2 (20)
move-right car2 row2 col2 col3 (20)
move-right car2 row2 col3 col4 (20)
move-up car2 row2 row1 col4 (20)
[t=2.10345s, 13724 KB] Plan length: 33 step(s).
[t=2.10345s, 13724 KB] Plan cost: 580
[t=2.10345s, 13724 KB] Expanded 14701 state(s).
[t=2.10345s, 13724 KB] Reopened 277 state(s).
[t=2.10345s, 13724 KB] Evaluated 59072 state(s).
[t=2.10345s, 13724 KB] Evaluations: 59072
[t=2.10345s, 13724 KB] Generated 163220 state(s).
[t=2.10345s, 13724 KB] Dead ends: 2803 state(s).
[t=2.10345s, 13724 KB] Expanded until last jump: 14683 state(s).
```

Problem3 code:

```
;; Problema 1

```

```
(define (problem pe-ntruca)
    (:domain trip-to-romania)

    (:objects
     car1 car2 car3
     p1
     row1 row2 row3 row4
     col1 col2 col3 col4)

    (:init
     (IS-CAR car1)
     (IS-CAR car2)
     (IS-CAR car3)
     (IS-ROW row1)
     (IS-ROW row2)
     (IS-ROW row3)
     (IS-ROW row4)
     (NEXT-ROW row1 row2)
     (NEXT-ROW row2 row3)
     (NEXT-ROW row3 row4)
     (IS-COLUMN col1)
     (IS-COLUMN col2)
     (IS-COLUMN col3)
     (IS-COLUMN col4)
     (NEXT-COLUMN col1 col2)
     (NEXT-COLUMN col2 col3)
     (NEXT-COLUMN col3 col4)

     (is-package p1)
     (package-at p1 row2 col4)
     (empty car1)
     (empty car2)
     (empty car3)

     ;;initial state
     (car-at car1 row1 col3)
     (car-at car2 row2 col1)
     (car-at car3 row3 col4)

     (rest-at row3 col3)
     (rest-at row1 col2)
     (rest-at row4 col2)
     (not (has-eaten car1))
     (not (has-eaten car2))
     (not (has-eaten car3))

     (gas-station row1 col1)
     (gas-station row1 col4)
     (gas-station row2 col2)
     (not (is-fueled car1))
     (not (is-fueled car2))
     (not (is-fueled car3))

     (not (is-fined car1))
```

```
      (not (moved-up car1))
      (not (moved-down car1))
      (not (moved-left car1))
      (not (moved-right car1))
      (camera-at row2 col3)

      (not (is-fined car2))
      (not (moved-up car2))
      (not (moved-down car2))
      (not (moved-left car2))
      (not (moved-right car2))

      (not (is-fined car3))
      (not (moved-up car3))
      (not (moved-down car3))
      (not (moved-left car3))
      (not (moved-right car3))

      (= (total-cost) 0)
      (= (cost car1) 0)
      (= (cost car2) 0)
    )

    (:goal
     (and
      (car-at car1 row3 col1)
      (car-at car2 row3 col2)
      (car-at car3 row3 col3)
      (has-eaten car1)
      (has-eaten car2)
      (has-eaten car3)
      (is-fueled car1)
      (is-fueled car2)
      (is-fueled car3)
      ;;(is-fined car2)
      ;;(is-fined car1)
      (package-at p1 row4 col4)
     )
    )

    (:metric minimize (total-cost))
    ;;(:metric minimize (cost car1))
    ;;(:metric minimize (cost car2))
)
```

```
[t=316.719s, 995704 KB] g=560, 18778788 evaluated, 4524741 expanded, 1834 reopened
[t=316.719s, 995704 KB] New best heuristic value for cg: 0
[t=316.719s, 995704 KB] g=580, 18778798 evaluated, 4524742 expanded, 1834 reopened
[t=316.72s, 995704 KB] Solution found!
[t=316.72s, 995704 KB] Actual search time: 316.707s
move-left car3 row3 col4 col3 (20)
move-down car2 row2 row3 col1 (20)
eatsomething car3 row3 col3 (0)
move-right car3 row3 col3 col4 (20)
move-up car3 row3 row2 col4 (20)
getpackage car3 p1 row2 col4 (20)
move-up car3 row2 row1 col4 (20)
fuelsomegas car3 row1 col4 (0)
move-left car3 row1 col4 col3 (20)
droppackage car3 p1 row1 col3 (20)
getpackage car1 p1 row1 col3 (20)
move-left car1 row1 col3 col2 (20)
move-down car3 row1 row2 col3 (20)
caughtoncamera car3 row2 col3 (0)
move-down car3 row2 row3 col3 (20)
eatsomething car1 row1 col2 (0)
move-left car1 row1 col2 col1 (20)
fuelsomegas car1 row1 col1 (0)
move-down car1 row1 row2 col1 (20)
move-down car1 row2 row3 col1 (20)
droppackage car1 p1 row3 col1 (20)
getpackage car2 p1 row3 col1 (20)
move-right car2 row3 col1 col2 (20)
move-down car2 row3 row4 col2 (20)
eatsomething car2 row4 col2 (0)
move-up car2 row4 row3 col2 (20)
move-up car2 row3 row2 col2 (20)
fuelsomegas car2 row2 col2 (0)
move-down car2 row2 row3 col2 (20)
move-down car2 row3 row4 col2 (20)
move-right car2 row4 col2 col3 (20)
move-right car2 row4 col3 col4 (20)
droppackage car2 p1 row4 col4 (20)
move-left car2 row4 col4 col3 (20)
move-left car2 row4 col3 col2 (20)
move-up car2 row4 row3 col2 (20)
[t=316.72s, 995704 KB] Plan length: 36 step(s).
[t=316.72s, 995704 KB] Plan cost: 580
[t=316.72s, 995704 KB] Expanded 4524743 state(s).
[t=316.72s, 995704 KB] Reopened 1834 state(s).
[t=316.72s, 995704 KB] Evaluated 18778805 state(s).
[t=316.72s, 995704 KB] Evaluations: 18778805
[t=316.72s, 995704 KB] Generated 79243413 state(s).
[t=316.72s, 995704 KB] Dead ends: 0 state(s).
[t=316.72s, 995704 KB] Expanded until last jump: 4524740 state(s).
```

Domain code:

```
;;2 masini merg in trip to romania. Scopul este sa ajunga
;;la Cluj. Daca o masina a mers in toate directiile, atunci o poate
;;opri politia pentru condus agresiv. Top 10 moduri de a lua amenda an ;;2021 @ UTCN.

(define (domain trip-to-romania)

  (:requirements :strips :action-costs)

  (:predicates
   (IS-CAR ?x)
   (IS-ROW ?x)
   (IS-COLUMN ?x)
   (NEXT-ROW ?r1 ?r2)
   (NEXT-COLUMN ?c1 ?c2)
   (car-at ?x ?r ?c)

   (is-package ?p)
```

```
    (has-package ?c ?p)
    (empty ?c)
    (package-at ?p ?row ?col)
    ;;(package-to ?p ?row ?col)

    (is-fined ?c)
    (is-speedster ?c)
    (is-fueled ?c)
    (has-eaten ?c)
    (rest-at ?row ?col)
    (camera-at ?row ?col)
    (gas-station ?row ?col)
    (moved-up ?c)
    (moved-down ?c)
    (moved-left ?c)
    (moved-right ?c)
   )

   (:functions
    (cost ?c) - number
    (total-cost) - number
   )

   (:action getPackage
    :parameters (?car ?p ?row ?col)
    :precondition (and (IS-CAR ?car)
                       (IS-ROW ?row)
                       (IS-COLUMN ?col)
                       (car-at ?car ?row ?col)
                       (is-package ?p)
                       (package-at ?p ?row ?col)
                       (not (has-package ?car ?p))
                       (empty ?car)
                  )
    :effect (and (not (package-at ?p ?row ?col))
                 (has-package ?car ?p)
                 (not (empty ?car))
                 (increase (total-cost) 20)
            )
   )

   (:action dropPackage
    :parameters (?car ?p ?row ?col)
    :precondition (and (IS-CAR ?car)
                       (IS-ROW ?row)
                       (IS-COLUMN ?col)
                       (car-at ?car ?row ?col)
                       (has-package ?car ?p)
                       (not (empty ?car))
                  )
    :effect (and (package-at ?p ?row ?col)
                 (not (has-package ?car ?p))
                 (empty ?car)
                 (is-speedster ?car)
```

```
                (increase (total-cost) 20)
            )
    )

    (:action caughtOnCamera
      :parameters (?car ?row ?col)
      :precondition (and (IS-CAR ?car)
                         (IS-ROW ?row)
                         (IS-COLUMN ?col)
                         (car-at ?car ?row ?col)
                         (camera-at ?row ?col)
                         (not (is-fined ?car))
                         (is-speedster ?car)
                         (moved-up ?car)
                         (moved-down ?car)
                         (moved-left ?car)
                         (moved-right ?car)
                     )
      :effect (and (is-fined ?car)
                   (not (is-speedster ?car))
              )
    )

    (:action eatSomething
      :parameters (?car ?row ?col)
      :precondition (and (IS-CAR ?car)
                         (IS-ROW ?row)
                         (IS-COLUMN ?col)
                         (car-at ?car ?row ?col)
                         (rest-at ?row ?col)
                         (not (has-eaten ?car))
                     )
      :effect (and (has-eaten ?car)
                   (increase (total-cost) 0)
                   (not (rest-at ?row ?col))
              )
    )

    (:action fuelSomeGas
      :parameters (?car ?row ?col)
      :precondition (and (IS-CAR ?car)
                         (IS-ROW ?row)
                         (IS-COLUMN ?col)
                         (car-at ?car ?row ?col)
                         (gas-station ?row ?col)
                         (not (is-fueled ?car))
                         (has-eaten ?car)
                         (not (empty ?car))
                     )
      :effect (and (is-fueled ?car)
                   (increase (total-cost) 0)
                   (not (gas-station ?row ?col))
              )
    )
```

```
126
127     (:action move-down
128       :parameters (?car ?old-row ?new-row ?col)
129       :precondition (and (IS-CAR ?car)
130                          (IS-ROW ?old-row)
131                          (IS-ROW ?new-row)
132                          (IS-COLUMN ?col)
133                          (NEXT-ROW ?old-row ?new-row)
134                          (car-at ?car ?old-row ?col)
135                          (or (not (is-speedster ?car))
136                              (not (camera-at ?old-row ?col))
137                          )
138                     )
139       :effect (and (not (car-at ?car ?old-row ?col))
140                    (car-at ?car ?new-row ?col)
141                    (moved-down ?car)
142                    (increase (total-cost) 20)
143               )
144     )
145
146     (:action move-up
147       :parameters (?car ?old-row ?new-row ?col)
148       :precondition (and (IS-CAR ?car)
149                          (IS-ROW ?old-row)
150                          (IS-ROW ?new-row)
151                          (IS-COLUMN ?col)
152                          (NEXT-ROW ?new-row ?old-row)
153                          (car-at ?car ?old-row ?col)
154                          (or (not (is-speedster ?car))
155                              (not (camera-at ?old-row ?col))
156                          )
157                     )
158       :effect (and (not (car-at ?car ?old-row ?col))
159                    (car-at ?car ?new-row ?col)
160                    (moved-up ?car)
161                    (increase (total-cost) 20)
162                    ;;(increase (cost ?car) 20)
163               )
164     )
165
166     (:action move-right
167       :parameters (?car ?row ?old-col ?new-col)
168       :precondition (and (IS-CAR ?car)
169                          (IS-ROW ?row)
170                          (IS-COLUMN ?old-col)
171                          (IS-COLUMN ?new-col)
172                          (NEXT-COLUMN ?old-col ?new-col)
173                          (car-at ?car ?row ?old-col)
174                          (or (not (is-speedster ?car))
175                              (not (camera-at ?row ?old-col))
176                          )
177                     )
178       :effect (and (not (car-at ?car ?row ?old-col))
179                    (car-at ?car ?row ?new-col)
```

```
                (moved-right ?car)
                (increase (total-cost) 20)
            )
    )

    (:action move-left
      :parameters (?car ?row ?old-col ?new-col)
      :precondition (and (IS-CAR ?car)
                         (IS-ROW ?row)
                         (IS-COLUMN ?old-col)
                         (IS-COLUMN ?new-col)
                         (NEXT-COLUMN ?new-col ?old-col)
                         (car-at ?car ?row ?old-col)
                         (or (not (is-speedster ?car))
                             (not (camera-at ?row ?old-col))
                         )
                     )
      :effect (and (not (car-at ?car ?row ?old-col))
                   (car-at ?car ?row ?new-col)
                   (moved-left ?car)
                   (increase (total-cost) 20)
            )
    )

)
```