Inteligență Artificială 2017 - Tema 2

Livrare comenzi

Vlad Bogolin

1. Introducere

În ultimul timp cumpărăturile online au devenit din ce în ce mai populare. Totuşi, timpul de aşteptare pentru livrarea unei comenzi poate fi relativ mare. Pentru a îmbunătăţi acest aspect, o idee ar fi utilizarea unei drone autonome pentru îmbunatăţirea timpului de livrare.

2. Problemă

Având la dispoziţie o dronă, o listă de comenzi ale clienţilor şi o disponibilitate a produselor în mai multe depozite, se doreşte realizarea unui plan de livrare a mărfurilor astfel încât toate comenzile să fie onorate.

2.1 Harta

Universul problemei este reprezentat printr-o hartă bidimensională (o matrice de dimensiune **n** x **m**). Drona poate zbura deasupra oricărei celule din hartă. Harta nu este ciclică, iar drona nu poate zbura în afara ei.

2.2 Drona

Iniţial, drona se află într-o anumită celulă de pe hartă. Drona dispune de un compartiment. În acest compartiment se poate depozita un produs. Drona poate zbura între oricare două poziţii de pe hartă. Ea va alege întotdeauna cea mai scurtă rută pentru un anumit traseu. Drona poate primi trei tipuri de comenzi: zbor dintr-un punct în alt punct, încărcare produs (în momentul în care se află într-un depozit) sau livrare produs (în momentul în care se află la un client care a comandat acel tip de produs).

2.3 Produse

Există mai multe tipuri de produse. Fiecare depozit dispune de o anumită listă de produse. Dacă un produs se găseşte într-un anumit depozit, atunci stocul lui este nelimitat.

2.4 Comenzi

O comandă este definita ca o pereche *(client, produs)* - nu se poate comanda mai mult de un produs într-o singură comandă. În momentul în care un client doreşte mai multe produse, se va considera că va plasa mai multe comenzi (acest lucru se aplică şi dacă un client doreşte mai multe produse de acelaşi tip).

3. Cerințe

3.1 Cerinta 1 - Descriere STRIPS [0.3p]

Realizaţi un document PDF cu descrierea STRIPS a problemei prezentate în secţiunea precedenta. Mai exact, se doreşte descrierea următorilor operatori:

- *Fly(startCell, stopCell)* - reprezintă acţiunea prin care drona se deplasează din celula *startCell* în celula *stopCell.*
- *Load(productId)* - reprezintă acţiunea prin care drona încarcă produsul *productId* în spaţiu de stocare.
- *Deliver(productId)* - reprezintă acţiunea prin care drona livrează produsul *productId*.

Pentru realizarea descrierii operatorilor se vor folosi următoarele predicate:

- *Position(cellId)* - drona se află în celula *cellId.*
- *Warehouse(cellId)* - în celula *cellId* se găseşte un depozit.
- *Client(cellId)* - în celula *cellId* se află un client.
- *hasProduct(warehouseId, productId)* - în depozitul *warehouseId* se găseşte produsul *productId.*
- *Order(clientId, productId)* - clientul *clientId* a comandat produsul *productId.*
- *Carries(productId)* - drona transportă produsul *productId.*
- *Empty()* - spatiul de stocare al dronei este gol.

Dacă este necesar, pe lângă predicatele de mai sus puteţi adăuga alte predicate şi explicaţi de ce este nevoie de ele.

3.2 Cerinţa 2 - Implementare [0.7p]

Implementaţi folosind limbajul de programare Python funcţia ***makePlan(scenario)*** care primeşte ca parametru un scenariu şi întoarce un plan valid care conduce la onorarea tuturor comenzilor. În cazul în care nu există un plan care să conducă la onorarea tuturor comenzilor (de exemplu,

un produs nu există în niciun depozit), funcţia va întoarce valoarea *False*. Pentru ca un plan să fie valid trebuie ca toate comenzile să fie onorate.

În implementarea algoritmului **este obligatoriu** să folosiţi o strategie care să realizeze o căutare înapoi (backward search). De asemenea, este **obligatoriu** ca algoritmul sa folosească aplicarea de operatori. Aşadar, operatorii trebuie definiţi în cod (folosind un mod de reprezentare la alegere) şi sa fie folosiţi în planificare.

3.2.1 Specificaţii

Funcţia va primi un singur argument ca parametru, mai exact un dicţionar cu următoarea structură:

- *dimensions* - un tuplu cu 2 valori *n* şi *m* care reprezintă dimensiunea hărţii.
- *warehouses* - o listă cu toate depozitele disponibile. Identificatorul unui depozit este dat de poziţia lui pe hartă ca fiind un tuplu de forma *(i, j)* cu $i \geq 0$ şi $i < n$, $j \geq 0$ şi $j < m$.
- *number_of_products* - numărul total de produse. Produsele vor fi codificate cu un indice de la 0 la numărul total de produse.
- *available_products* - o listă de tupluri *(warehouseId, productId)* care semnifică că în depozitul *warehouseId* se găseşte produsul *productId*, stocul fiind nelimitat.
- *clients* - o lista cu toti clienţii disponibili. Identificatorul unui client este dat de poziţia lui pe hartă ca fiind un tuplu de forma *(i, j)* cu $i \geq 0$ şi $i < n$, $j \geq 0$ şi $j < m$.
- *orders* - listă de tupluri de forma *(clientId, productId)*.
- *initial_position* - poziţia de start a dronei. Poziţia este specificată printr-un tuplu *(i, j)* cu $i \geq 0$ şi $i < n$, $j \geq 0$ şi $j < m$.

Rezultatul întors de **makePlan** este reprezentat de o listă de operatori *Fly*, *Load*, *Deliver* aşa cum au fost definiţi în secţiunea 3.1. Identificatorul unei celule este definit ca un tuplu *(i, j)* unde *i* reprezinta linia din hartă şi *j* coloana (vezi secţiunea 5 pentru un exemplu).

3.3 Bonus [max 0.2p]

Pentru această cerinţă se consideră că timpul necesar pentru zborul dronei este egal cu distanţa dintre poziţia de start şi poziţia de stop (drona foloseşte cea mai scurta cale dintre start şi stop, aşadar distanţa este egală cu distanţa Euclidiana dintre cele două poziţii). Optimizaţi planul obţinut pentru a minimiza timpul total de livrare. Timpul total de livrare este definit ca suma tuturor timpilor pentru toate operaţiile de tip “Fly”. Se consideră că încărcarea şi descărcarea dronei au un timp neglijabil. Discutaţi optimizările aduse într-un fişier separat facând comparaţie cu varianta neoptimizată.

4. Trimiterea temei

Tema se va trimite într-o arhivă *.zip* cu numele:

Nume_Prenume_Grupa_IA_T2.zip

Arhiva va conţine cel puţin următoarele fişiere:

- *strips.pdf* - pentru rezolvarea cerinţei 1.
- *deliver.py* - în care va fi definită funcţia *makePlan* pentru rezolvarea cerinţei 2.
- *bonus.py* - în care va fi definită funcţia *makePlan* pentru bonus. În acest caz în *readme* trebuie să se regăsească şi explicaţiile şi comparaţiile (grafice, tabele, etc)
- *readme*

5. Exemplu de scenariu

{
'available_products': [((0, 2), 1), ((0, 2), 2), ((0, 2), 0), ((0, 2), 3)],
'initial_position': (2, 1),
'dimensions': (5, 5),
'warehouses': [(0, 2)],
'clients': [(0, 3), (0, 1), (4, 3)],
'number_of_products': 4,
'orders': [((0, 1), 0), ((4, 3), 3), ((0, 3), 0)]
}

Posibil output pentru scenariu:
['Fly((2, 1), (0,2))', 'Load(0)', 'Fly((0, 2), (0, 1))', 'Deliver(0)', 'Fly((0, 1), (0, 2))', 'Load(0)', 'Fly((0, 2), (0, 3))', 'Deliver(0)', 'Fly((0, 3), (0, 2))', 'Load(3)', 'Fly((0, 2), (4, 3))', 'Deliver(3)']