Planning and Scheduling Project

Storing Groceries b-it-bots @Home

Project Report

Sushma Devaramani, Vajra Ganeshkumar, Supriya Vadiraj

Hochschule Bonn-Rhein-Sieg

Contents

1 Approach **3**
1.1 HTN Planner 3
1.1.1 Installation Instructions: 3

2 Operators and Methods **5**
2.1 Analysis: 6

3 References **7**

Chapter 1

Approach

1.1 HTN Planner

We have chosen *Pyhop*, a HTN planner. Pyhop is a Hierarchical Task Network Planner which is written in python. It is developed by Dana S. Nau. It is selected as it is easy to implement and generates plan in approximately 0.37seconds [2].

In *pyhop* the methods and operators used are defined as python functions.

1.1.1 Installation Instructions:

Pyhop is available on github. It can be installed using the following commands

- To clone the github repository: git clone https://github.com/oubiwann/pyhop.git
- cd pyhop
- sudo python setup.py install

The first task for us was to get the planner running. We used the simple travel example, which was written by the author to check its functioning. The planning problem written in pyhop is executed similar to a python script[2].

The problem selected and the results are as shown in the below pictures.

```
TASK:         METHODS:
travel        travel_by_foot, travel_by_taxi

********************************************************************************
Call hop.plan(state1,[('travel','me','home','park')]) with different verbosity l
evels
********************************************************************************

- If verbose=0 (the default), Pyhop returns the solution but prints nothing.

- If verbose=1, Pyhop prints the problem and solution, and returns the solution:
** hop, verbose=1: **
   state = state1
   tasks = [('travel', 'me', 'home', 'park')]
** result = [('call_taxi', 'me', 'home'), ('ride_taxi', 'me', 'home', 'park'), (
'pay_driver', 'me')]

- If verbose=2, Pyhop also prints a note at each recursive call:
** hop, verbose=2: **
   state = state1
   tasks = [('travel', 'me', 'home', 'park')]
depth 0 tasks [('travel', 'me', 'home', 'park')]
depth 1 tasks [('call_taxi', 'me', 'home'), ('ride_taxi', 'me', 'home', 'park'),
 ('pay_driver', 'me')]
depth 2 tasks [('ride_taxi', 'me', 'home', 'park'), ('pay_driver', 'me')]
depth 3 tasks [('pay_driver', 'me')]
depth 4 tasks []
** result = [('call_taxi', 'me', 'home'), ('ride_taxi', 'me', 'home', 'park'), (
'pay_driver', 'me')]

- If verbose=3, Pyhop also prints the intermediate states:
** hop, verbose=3: **
   state = state1
   tasks = [('travel', 'me', 'home', 'park')]
depth 0 tasks [('travel', 'me', 'home', 'park')]
depth 0 method instance ('travel', 'me', 'home', 'park')
depth 0 new tasks: False
depth 0 new tasks: [('call_taxi', 'me', 'home'), ('ride_taxi', 'me', 'home', 'pa
rk'), ('pay_driver', 'me')]
depth 1 tasks [('call_taxi', 'me', 'home'), ('ride_taxi', 'me', 'home', 'park'),
 ('pay_driver', 'me')]
depth 1 action ('call_taxi', 'me', 'home')
depth 1 new state:
    state1.owe(' =', {'me': 0})
    state1.dist(' =', {'home': {'park': 8}, 'park': {'home': 8}})
    state1.cash(' =', {'me': 20})
    state1.loc(' =', {'me': 'home', 'taxi': 'home'})
depth 2 tasks [('ride_taxi', 'me', 'home', 'park'), ('pay_driver', 'me')]
depth 2 action ('ride_taxi', 'me', 'home', 'park')
depth 2 new state:
    state1.owe(' =', {'me': 5.5})
    state1.dist(' =', {'home': {'park': 8}, 'park': {'home': 8}})
    state1.cash(' =', {'me': 20})
    state1.loc(' =', {'me': 'park', 'taxi': 'park'})
depth 3 tasks [('pay_driver', 'me')]
depth 3 action ('pay_driver', 'me')
depth 3 new state:
    state1.owe(' =', {'me': 0})
    state1.dist(' =', {'home': {'park': 8}, 'park': {'home': 8}})
    state1.cash(' =', {'me': 14.5})
    state1.loc(' =', {'me': 'park', 'taxi': 'park'})
depth 4 tasks []
depth 4 returns plan [('call_taxi', 'me', 'home'), ('ride_taxi', 'me', 'home', '
park'), ('pay_driver', 'me')]
** result = [('call_taxi', 'me', 'home'), ('ride_taxi', 'me', 'home', 'park'), (
'pay_driver', 'me')]
```

Chapter 2

Operators and Methods

Operators:

1. goto(r,x,y): robot goes from x to y
   - Precond: robot is at x
   - Effects: robot is at y
2. open(r,x) : robot opens cupboard door
   - Precond: robot is at x, hand empty
   - Effect: x is open
3. pick(r,o,x) : robot picks o from x
   - Precond: handempty, robot at x, o is at x
   - Effect: pick o from x, holding object x, object o $\neg$ on place x
4. place(r,o,x) : robot places o in x
   - Precond: holding object o , robot is at x
   - Effect: object o is on y
5. look_table(r): robot locates table
6. look_cupboard(r): robot located cupboard
7. perceiveObjects(r,y): robot perceives objects on table

Methods:
The robot performs the tasks sequentially.

1. **case 1:**
   (open,r, x), (goto,r, x, y), (pick,r,o,y), (goto,r, y, x), (place,r,o,x)
2. **case 2:**
   (look_cupboard,r), (goto, r, ,x), (open, r, x), (look_table, r), (goto, r, x, y), (pick, r,o,y), (goto, r, y, x), (place, r, o, x), (goto, r, x, y)
3. **case 3:**
   (look_cupboard,r), (goto, r, ,x), (open, r, x), (look_table, r), (goto, r, x, y),(perceiveObjects, y) (pick, r,o,y), (goto, r, y, x), (place, r, o, x), (goto, r, x, y)
4. **case 4:**
   (look_cupboard,r), (goto, r, , x), (open,r, x), (look_table,r), (goto,r, x, y), (perceiveObjects, y, i), (pick,r, o, y, i), (goto,r, y, x), (place,r,o,x, i), (goto,r, x, y)

2.1 Analysis:

- The planner generates one plan for each case. The plan generated approximately took 0.37seconds
- The operators and methods were specific to each case. The planner do not allow a general specification of tasks (i.e., each problem has to be modeled specifically).
- Lack of knowledge concerning optimality of the generated plan.
- Difficulty in invoking operators recursively in a single task. For instance, the recursive task of picking and placing an object involves invoking the "pick" and "place" operators until all the objects are picked. We found it difficult to handle such cases using pyhop planner.

Chapter 3

References

1. Github link for pyhop–`https://bitbucket.org/dananau/pyhop/src/195ab6320571?at=default`

2. HTN Planning – `http://www.cs.umd.edu/~nau/papers/nau2013game.pdf`