Udacity Artificial Intelligence Nanodegree

Implementing a Planning Search

Federico Bogado, December 31, 2017

In this project, the objective was to implement a planning search algorithm to solve problems of air cargo transportation.

The first part of the project consisted on implementing the objects and functions needed to model the problem. Then three different problems were solved with a couple of different search algorithms. The results were analysed to compare them.

The second part of the project was more focused on implementing the PlanningGraph object, that models a planning graph for a problem, so we can run an heuristic search on it. Two different heuristic functions were tested, and the results were analysed afterwards.

Optimal solutions for the problems:

- ### Poblem 1

```
Load(C1, P1, SFO)
Load(C2, P2, JFK)
Fly(P1, SFO, JFK)
Fly(P2, JFK, SFO)
Unload(C1, P1, JFK)
Unload(C2, P2, SFO)
```

- ### Problem 2

```
Load(C1, P1, SFO)
Load(C2, P2, JFK)
Load(C3, P3, ATL)
Fly(P1, SFO, JFK)
Fly(P2, JFK, SFO)
Fly(P3, ATL, SFO)
Unload(C3, P3, SFO)
Unload(C1, P1, JFK)
Unload(C2, P2, SFO)
```

- Problem 3

```
Load(C1, P1, SFO)
Load(C2, P2, JFK)
Fly(P1, SFO, ATL)
Load(C3, P1, ATL)
Fly(P2, JFK, ORD)
Load(C4, P2, ORD)
Fly(P2, ORD, SFO)
Fly(P1, ATL, JFK)
Unload(C4, P2, SFO)
Unload(C3, P1, JFK)
Unload(C1, P1, JFK)
Unload(C2, P2, SFO)
```

Problem solving analysis

This are the metrics obtained while trying to solve the problems with different search algorithms:

Problem 1						
Search Algorithm	**Heuristic function**	**Plan length**	**Time elapsed [s]**	**Expansions**	**Goal Tests**	**New Nodes**
breadth_first_search	---	**6**	0,1358	43	56	180
depth_first_graph_search	---	12	0,0296	12	13	48
uniform_cost_search	---	**6**	0,1219	55	57	224
astar_search	h_1	**6**	0,1532	55	57	224
astar_search	h_ignore_preconditions	**6**	0,1427	41	43	170
astar_search	h_pg_levelsum	**6**	0,7566	19	21	86

Problem 2						
Search Algorithm	**Heuristic function**	**Plan length**	**Time elapsed [s]**	**Expansions**	**Goal Tests**	**New Nodes**
breadth_first_search	---	**9**	29,5716	3343	4609	30509
depth_first_graph_search	---	1444	25,4775	1669	1670	14863
uniform_cost_search	---	**9**	42,6137	4852	4854	44030
astar_search	h_1	**9**	45,6548	4852	4854	44030
astar_search	h_ignore_preconditions	**9**	14,0753	1450	1452	13303
astar_search	h_pg_levelsum	**9**	99,6615	280	282	2707

Problem 3						
Search Algorithm	**Heuristic function**	**Plan length**	**Time elapsed [s]**	**Expansions**	**Goal Tests**	**New Nodes**
breadth_first_search	---	**12**	166,2885	14663	18098	129631
depth_first_graph_search	---	571	7,0604	592	593	4927
uniform_cost_search	---	**12**	184,2686	18235	18237	159716
astar_search	h_1	**12**	211,1742	18235	18237	159716
astar_search	h_ignore_preconditions	**12**	61,7397	5040	5042	44944
astar_search	h_pg_levelsum	**12**	590,9608	1335	1337	11645

Non heuristic search analysis

The problems were solved using three non heuristic searches: breadth first search, depth first search, and uniform cost search. Every time, the algorithms reached a solution, in a reasonable amount of time.

The first thing to be noticed is that the depth first search doesn't find an optimal solution (as expected). This is due to the own nature of the algorithm.

Both breadth first search and uniform cost search, beehive similarly: they get an optimal solution, they traverse similar amount of space of the tree graph, and they take similar amount of times. In a problem like this, where the cost of a solution is the amount of steps to get it, we can expect this kind of results, where this algorithms find the best solution possible.

Heuristic search

The first thing to notice on the heuristic search is that it return an optimal solution every time. This behavior is expected.

Another interesting thing to notice is that the portion of the tree explored is higher when the heuristic is simpler. This means that a more precise function saves tree exploration. Nevertheless, a more complicated a precise function can lead to a bigger execution time, because we have to constantly use the function. We can see that the last function explores a lot less nodes than the other two, but it takes longer to come up with a solution.

Overall comparison

When we look at all the metrics from all the test we can see that there is no optimal solution for all the problems.

The problems have not the same complexity so the methods to solve them present different behaviors.

We can see that for fairly easy problems, the best approach is to go with a non heuristic search, because they are by far the simplest and fastest. But if the complexity of the problem is a little higher, we should use a heuristic search, and we have develop the best possible heuristic we can, but we need to keep in mind that if the heuristic gets complex and computationally expensive, this can lead to a really long running time, even if we explore a lot less search space.

Conclusions

When we are trying to solve a planning problem, we need to take into account what we need to solve, so we can take the best possible actions to solve it.

The complexity of the problem is a big factor when deciding how to approach the solution.

Another thing to remember, is that even if more complex heuristics lead to search a lot shorter portion of the search tree, it can be computationally expensive, and therefore take long to find a solution.