AIND Planning Assignment

Guruprasad Sridharan

May 2017

1 Problem Definition

Given three problems in the Air Cargo domain, we are asked to provide a planning search algorithm to find an optimal plan for solving these problems. The Air Cargo Problem Action Schema is as follows.
$Action(Load(c, p, a)$,
$PRECOND : At(c, a) \land At(p, a) \land Cargo(c) \land Plane(p) \land Airport(a)$,
$EFFECT : \neg At(c, a) \land In(c, p))$
$Action(Unload(c, p, a)$,
$PRECOND : In(c, p) \land At(p, a) \land Cargo(c) \land Plane(p) \land Airport(a)$,
$EFFECT : At(c, a) \land \neg In(c, p))$
$Action(Fly(p, from, to)$,
$PRECOND : At(p, from) \land Plane(p) \land Airport(from) \land Airport(to)$,
$EFFECT : \neg At(p, from) \land At(p, to))$

1.0.1 Problem 1

Init State:
$(At(C1, SFO) \land At(C2, JFK)$
$\land At(P1, SFO) \land At(P2, JFK)$
$\land Cargo(C1) \land Cargo(C2)$
$\land Plane(P1) \land Plane(P2)$
$\land Airport(JFK) \land Airport(SFO))$

Goal State:
$(At(C1, JFK) \land At(C2, SFO))$

Optimal Plan:
Load(C1, P1, SFO)
Load(C2, P2, JFK)
Fly(P1, SFO, JFK)
Fly(P2, JFK, SFO)
Unload(C1, P1, JFK)

Unload(C2, P2, SFO)

1.0.2 Problem 2

Init State:
$Init(At(C1, SFO) \land At(C2, JFK) \land At(C3, ATL)$
$\land At(P1, SFO) \land At(P2, JFK) \land At(P3, ATL)$
$\land Cargo(C1) \land Cargo(C2) \land Cargo(C3)$
$\land Plane(P1) \land Plane(P2) \land Plane(P3)$
$\land Airport(JFK) \land Airport(SFO) \land Airport(ATL))$

Goal State:
$(At(C1, JFK) \land At(C2, SFO) \land At(C3, SFO))$

Optimal Plan:
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(C1, P1, JFK)
Unload(C2, P2, SFO)
Unload(C3, P3, SFO)

1.0.3 Problem 3

Init State:
$(At(C1, SFO) \land At(C2, JFK) \land At(C3, ATL) \land At(C4, ORD)$
$\land At(P1, SFO) \land At(P2, JFK)$
$\land Cargo(C1) \land Cargo(C2) \land Cargo(C3) \land Cargo(C4) \land Plane(P1) \land Plane(P2)$
$\land Airport(JFK) \land Airport(SFO) \land Airport(ATL) \land Airport(ORD))$

Goal State:
$(At(C1, JFK) \land At(C3, JFK) \land At(C2, SFO) \land At(C4, SFO))$

Optimal Plan:
Load(C1, P1, SFO)
Load(C2, P2, JFK)
Fly(P1, SFO, ATL)
Fly(P2, JFK, ORD)
Load(C3, P1, ATL)
Load(C4, P2, ORD)
Fly(P1, ATL, JFK)

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

2 Benchmarking Setup

All the search algorithms were run on a machine with the following configuration

- Ubuntu 14.04 x86_64 architecture
- 128 GB GDDR3 RAM
- 40 cores of Intel(R) Xeon(R) CPU E5-2690 v2 @ 3.00GHz
- PyPy3.5

3 Non-Heuristic Methods

Problem 1

Algorithm	Optimal	Expansions	Goal Tests	New Nodes	Time
Breadth First	Yes	43	56	180	0.34
Depth First	No	12	13	48	0.13
Uniform Cost	Yes	55	57	224	0.35

Problem 2

Algorithm	Optimal	Expansions	Goal Tests	New Nodes	Time
Breadth First	Yes	3343	4609	30509	62.09
Depth First	No	582	583	5211	9.75
Uniform Cost	Yes	4852	4854	44030	86.52

Problem 3

Algorithm	Optimal	Expansions	Goal Tests	New Nodes	Time
Breadth First	Yes	14663	18098	129631	312.14
Depth First	No	627	628	5176	11.84
Uniform Cost	Yes	18235	18237	159716	374.37

For all three problems, depth first search didn't find the optimal plan. But at the same time, it was the fastest search algorithm with the fewest expansions, goal tests and new nodes. It requires lesser memory as well to execute. For bigger problem state spaces, it performs poorly. The solution provided by depth first search is farther from the optimal solution as the problem state space grows.

Breadth first search provides the optimal plan for all three problems. It takes more time and memory than depth first search but requires less time and memory than uniform cost search.

Uniform cost search also provides the optimal plan for all three problems. It actually does the most number of expansions and goal tests. It is memory intensive compared to other algorithms since it uses a priority queue to compute the path with shortest cost to reach the goal.

Depth first search should be used whenever the agent has constrained resources for computation and finding an optimal solution is not important and any sub-optimal solution which is off by a factor of 2 to 3 works. Breadth first search should be used whenever optimal solution is required since it is more time and space efficient compared to uniform cost search. Even though BFS is complete and optimal, number of nodes expanded approaches O(b^d) where b is branching factor and d is depth of tree. Thus it is not suitable for problems with high branching factor.

3.1 Heuristic Methods

Problem 1

Algorithm	Optimal	Expansions	Goal Tests	New Nodes	Time
A^*_H1	Yes	55	57	224	0.34
A^*_H_Ignore_Precond	Yes	41	43	170	0.86
A^*_H_PG_Level_Sum	Yes	11	13	50	2.04

Problem 2

Algorithm	Optimal	Expansions	Goal Tests	New Nodes	Time
A^*_H1	Yes	4852	4854	44030	84.57
A^*_H_Ignore_Precond	Yes	1450	1452	13303	64.24
A^*_H_PG_Level_Sum	Yes	86	88	841	534.52

Problem 3

Algorithm	Optimal	Expansions	Goal Tests	New Nodes	Time
A^*_H1	Yes	18235	18237	159716	365.99
A^*_H_Ignore_Precond	Yes	5040	5042	44944	229.15
A^*_H_PG_Level_Sum	Yes	86	88	841	NA

A^* algorithm was tested with three heuristic functions H1, H_Ignore_Preconditions and H_PG_Level_Sum. All three methods performed very well and found the optimal plan for all the problems.

As evidenced by the data presented in the tables, A^* with Level Sum heuristic performs the fewest number of expansions and goal tests. It is the heuristic function that requires least amount of memory. At the same time, Level Sum heuristic takes the maximum time to execute. This was because every time the

routine for heuristic ran, the planning graph hd to be constructed. For problem 3, the program ran for more than 30 minutes and had to be terminated.

A^* with Ignore Preconditions heuristic took the least amount of time to execute. A^* with H1 heuristic was the most memory intensive and did the most number of expansions and goal tests. This wasn't surprising because H1 was a dummy heuristic which doesn't approximate the true distance to the goal correctly. Moreover, it is not admissible. Because of this, the search might be lead astray. Generally, A^* should perform better than BFS even with any reasonable admissible heuristic since it is goal directed. The Ignore Preconditions heuristic is a very good estimator of distance to goal since it approximately computes the number of actions discarding their preconditions that need to done to meet the conditions of the goal state. It will always underestimate the true distance to the goal state ie. it's admissible and hence it is guaranteed to find the optimal solution.

4 Best Search Algorithm

Considering the speed of execution, optimality of solution, memory requirements and goal tests for each search algorithm, the best algorithm for solving the air cargo problems is A^* with Ignore Preconditions heuristic. It compares favourably with uninformed non-heuristic search algorithms and beats the best performing Breadth First Search algorithm by quite a margin. It requires less memory but performs much faster than breadth first search. Refer the graph attached below to look at the run time comparison for all six search algorithms analysed so far (both heuristic and non-heuristic).