Build a Forward-Planning Agent

Mohammad Al-Fetyani

April 2020

Introduction

Planning is an important topic in AI because intelligent agents are expected to automatically plan their own actions in uncertain domains. Planning and scheduling systems are commonly used in automation and logistics operations, robotics and self-driving cars, and for aerospace applications like the Hubble telescope and NASA Mars rovers.

This project is split between implementation and analysis. First we will combine symbolic logic and classical search to implement an agent that performs progression search to solve planning problems. Then we will experiment with different search algorithms and heuristics, and use the results to answer questions about designing planning systems.

I implemented and experimented different search algorithms and heuristics on four different types of Air Cargo problem. In total, there are 11 different search algorithms used in this project.

Problem Definition

Figure 1 shows an air cargo transport problem involving loading and unloading cargo and flying it from place to place. The problem can be defined with three actions: *Load*, *Unload*, and *Fly*. The actions affect two predicates: $In(c, p)$ means that cargo c is inside plane p, and $At(x, a)$ means that object x (either plane or cargo) is at airport a.

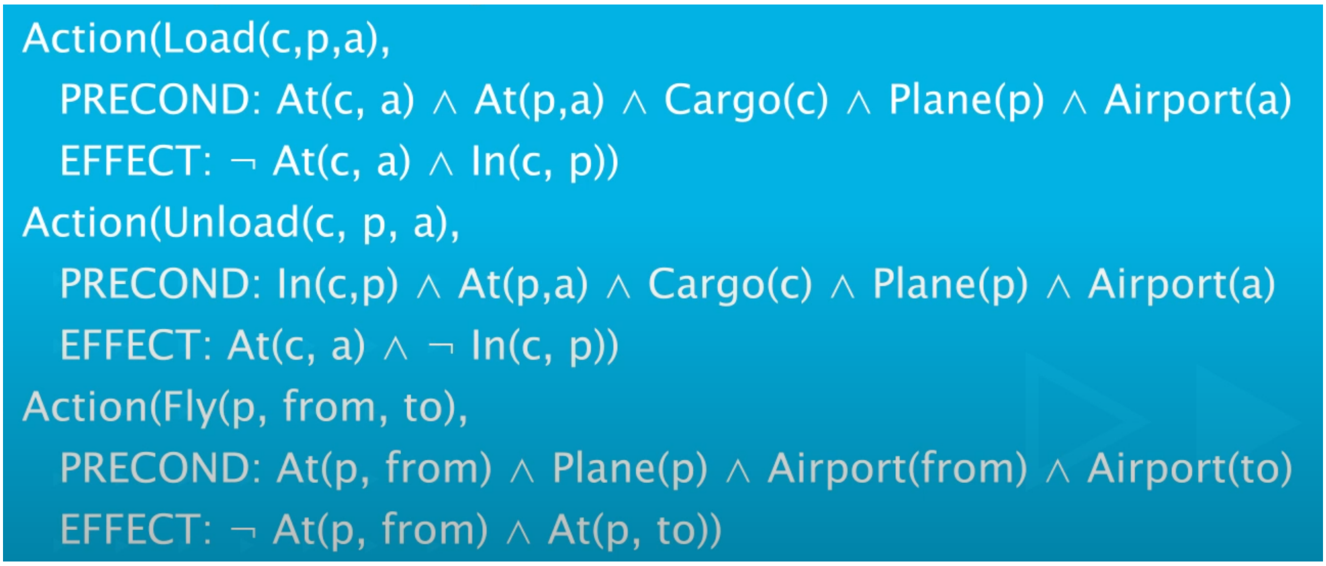

Figure 1: Air cargo transport problem.

Results

The results of running all search algorithms on problem 1 and problem 2 are presented in Table 1. Problems 3 and 4 results are presented in Table 2. The algorithms abbreviation shown in the tables are as follows:
BFS - Breadth First Search
DFGS - Depth First Graph Search
UFS - Uniform Cost Search
GBFGS1 - Greedy Best First Graph Search with *h_unmet_goals*
GBFGS2 - Greedy Best First Graph Search with *h_pg_levelsum*
GBFGS3 - Greedy Best First Graph Search with *h_pg_maxlevel*
GBFGS4 - Greedy Best First Graph Search with *h_pg_setlevel*
AS1 - A* Search with *h_unmet_goals*
AS2 - A* Search with *h_pg_levelsum*
AS3 - A* Search with *h_pg_maxlevel*
AS4 - A* Search with *h_pg_setlevel*

Algorithm	Problem 1			Problem 2		
	# Actions	# Expansions	Time elapsed (s)	# Actions	# Expansions	Time elapsed (s)
BFS	20	43	0.00613	72	3343	2.02763
DFGS	20	21	0.00340	72	624	3.19528
UFS	20	60	0.01012	72	5154	3.38656
GBFGS1	20	7	0.00162	72	17	0.01935
GBFGS2	20	6	0.52969	72	9	12.2725
GBFGS3	20	6	0.39826	72	27	24.6956
GBFGS4	20	6	0.68100	72	9	17.1734
AS1	20	50	0.00957	72	2467	2.25946
AS2	20	28	1.36174	72	357	315.49
AS3	20	43	1.40070	72	2887	1811.941
AS4	20	33	1.60234	72	1037	1540.38

Table 1: Results of problem 1 & 2.

Algorithm	Problem 3			Problem 4		
	# Actions	# Expansions	Time elapsed (s)	# Actions	# Expansions	Time elapsed (s)
BFS	88	14663	10.564	104	99736	94.689
DFGS	88	408	1.1965	-	-	-
UFS	88	18510	14.348	104	113339	117.03
GBFGS1	88	25	0.03631	104	29	0.0591
GBFGS2	88	14	28.179	104	17	50.196
AS1	88	7388	8.3706	104	34330	55.012
AS2	88	369	510.12	104	1208	2833.0

Table 2: Results of problem 3 & 4.

The analysis between the number of nodes expanded against number of actions in the domain shown in the above two tables illustrate one observation that in general there is no obvious relationship between the number of expanded nodes and the number of actions. The tables also illustrate the relationship between the search time and the number of actions in which we can say in general as the number of actions increase, the search time also increases.

The analysis between the length of the plans returned by each algorithm on all search problems is presented in Table 3.

Algorithm	Path Length			
	Problem 1	Problem 2	Problem 3	Problem 4
BFS	6	9	12	14
DFGS	20	619	392	-
UFS	6	9	12	14
GBFGS1	6	9	15	18
GBFGS2	6	9	14	17
GBFGS3	6	9	-	-
GBFGS4	6	9	-	-
AS1	6	9	12	14
AS2	6	9	12	15
AS3	6	9	-	-
AS4	6	9	-	-

Table 3: Path length of all problems.

(Q1) Which algorithm or algorithms would be most appropriate for planning in a very restricted domain (i.e., one that has only a few actions) and needs to operate in real time?
ANS: I would go for BFS or UFS algorithm because they provide optimal values in approximately no time. This is obvious from Problem 1 where there is few actions to consider.

(Q2) Which algorithm or algorithms would be most appropriate for planning in very large domains (e.g., planning delivery routes for all UPS drivers in the U.S. on a given day)
ANS: In this case, I would choose GBFGS algorithm because it is very fast in larger domains and does give reasonable and really good values as observed from all problems.

(Q3) Which algorithm or algorithms would be most appropriate for planning problems where it is important to find only optimal plans?
ANS: I would consider BFS or UFS for small domains and AS for large domains.