AIND Planning Project: Written Analysis

Jason Mancuso

April 2017

Optimal Plans

Problem 1

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

Problem 2

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

Problem 3

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

Results

Naive Search

Table 1: Naive results by algorithm for problem 1.

Algorithm	Node Expansions	Goal Tests	Time Elapsed (s)	Optimality
BFS	43	56	0.0289	T
DFS	21	22	0.0114	F
Uniform Cost	55	57	0.0353	T

Table 2: Naive results by algorithm for problem 2.

Algorithm	Node Expansions	Goal Tests	Time Elapsed (s)	Optimality
BFS	3343	4609	12.0841	T
DFS	624	625	2.9079	F
Uniform Cost	4852	4854	40.0885	T

Table 3: Naive results by algorithm for problem 3.

Algorithm	Node Expansions	Goal Tests	Time Elapsed (s)	Optimality
BFS	14663	18098	92.2627	T
DFS	408	409	1.5411	F
Uniform Cost	18235	18237	381.0992	T

Analysis

According to the above results, breadth-first search and uniform cost search find an optimal solution, while depth-first search does not. While DFS finds its solution much faster than either of the others, BFS is the fastest search algorithm that yields an optimal solution. If finding an optimal solution is necessary, BFS is recommended due to speed and memory considerations.

It is noteworthy that BFS visits less nodes than UCS. BFS and UCS perform a similar number of goal tests. DFS visits far less nodes and performs far fewer goal tests, which is why it is so much faster than BFS and UCS.

Heuristic Search

Table 4: Heuristic results for problem 1.

Heuristic	Node Expansions	Goal Tests	Time Elapsed (s)	Optimality
H1	55	57	0.0373	T
Ignore Preconditions	41	43	0.0457	T
Level-Sum	11	13	3.015	T

Analysis

A* search finds an optimal solution with all of the heuristics used, as all are formally admissible (they do not overestimate the true cost). For problem 3, I did not execute A* with the Level-Sum heuristic according to the project instructions, as it would have taken longer than ten minutes to finish its search. Note that the Level-Sum heuristic, although painfully slow to execute, dramatically improves the search as measured

Table 5: Heuristic results for problem 2.

Heuristic	**Node Expansions**	**Goal Tests**	**Time Elapsed (s)**	**Optimality**
H1	4852	4854	42.34	T
Ignore Preconditions	1506	1508	13.62	T
Level-Sum	86	88	724.9	T

Table 6: Heuristic results for problem 3.

Heuristic	**Node Expansions**	**Goal Tests**	**Time Elapsed (s)**	**Optimality**
H1	18235	18237	324.8	T
Ignore Preconditions	5118	5120	82.59	T
Level-Sum	-	-	-	-

by the number of nodes visited and goal test metrics. However, the time required to generate the planning graph far outweighs the efficiency improvement for these simple problems. Since each heuristic finds an optimal plan, I prefer the fastest, which was the Ignore Preconditions heuristic.

Comparing Naive with Heuristic Search

I compare the uninformed BFS search with A* search with the Ignore Preconditions heuristic.

Table 7: Heuristic versus Naive Search for problem 1.

Problem	**Algorithm**	**Node Expansions**	**Goal Tests**	**Time Elapsed (s)**
1	BFS	43	56	0.0289
	A* Ignore Preconditions	41	43	0.0457
2	BFS	3343	4609	12.08
	A* Ignore Preconditions	1506	1508	13.62
3	BFS	14663	18098	92.26
	A* Ignore Preconditions	5118	5120	82.59

Firstly, I note that A* visits fewer nodes and performs fewer goal tests than BFS. The difference in traversal efficiency is more noticeable as the complexity of the problem increases. As shown in Table 7, BFS performs faster in the less complex problems, i.e. problems 1 and 2. However, the efficiency gained by the heuristic-guided best-first approach of A* gives it an edge in problem 3.

Given this information, I would choose breadth-first search for simple planning problems and A* search for complex planning problems.