Heuristic Analysis

Derek Wang

Jan 7, 2018

Introduction

In this analysis, I look at the search algorithm results for the air cargo problems defined in the Planning project. These problems involve airplanes, cargo, and airports and each contain a start state as well as an end state. The objective of the search engine is to find an optimal set of predefined moves that will take the state of each problem from the initial state to the goal state. The moves defined are loading and unloading specific cargo from specific planes as well as the flight of a plane from airport 1 to airport 2.

In this project, we test 3 air cargo problems defined thusly:

Air Cargo Action Schema:

Action(Load(c, p, a),
PRECOND: At(c, a) ∧ At(p, a) ∧ Cargo(c) ∧ Plane(p) ∧ Airport(a)
EFFECT: ¬ At(c, a) ∧ In(c, p))
Action(Unload(c, p, a),
PRECOND: In(c, p) ∧ At(p, a) ∧ Cargo(c) ∧ Plane(p) ∧ Airport(a)
EFFECT: At(c, a) ∧ ¬ In(c, p))
Action(Fly(p, from, to),
PRECOND: At(p, from) ∧ Plane(p) ∧ Airport(from) ∧ Airport(to)
EFFECT: ¬ At(p, from) ∧ At(p, to))

- Problem 1 initial state and goal:

Init(At(C1, SFO) ∧ At(C2, JFK)
∧ At(P1, SFO) ∧ At(P2, JFK)
∧ Cargo(C1) ∧ Cargo(C2)
∧ Plane(P1) ∧ Plane(P2)
∧ Airport(JFK) ∧ Airport(SFO))
Goal(At(C1, JFK) ∧ At(C2, SFO))

- Problem 2 initial state and goal:

Init(At(C1, SFO) ∧ At(C2, JFK) ∧ At(C3, ATL)
∧ At(P1, SFO) ∧ At(P2, JFK) ∧ At(P3, ATL)

∧ Cargo(C1) ∧ Cargo(C2) ∧ Cargo(C3)

∧ Plane(P1) ∧ Plane(P2) ∧ Plane(P3)

∧ Airport(JFK) ∧ Airport(SFO) ∧ Airport(ATL))

Goal(At(C1, JFK) ∧ At(C2, SFO) ∧ At(C3, SFO))

•Problem 3 initial state and goal:

Init(At(C1, SFO) ∧ At(C2, JFK) ∧ At(C3, ATL) ∧ At(C4, ORD)

∧ At(P1, SFO) ∧ At(P2, JFK)

∧ Cargo(C1) ∧ Cargo(C2) ∧ Cargo(C3) ∧ Cargo(C4)

∧ Plane(P1) ∧ Plane(P2)

∧ Airport(JFK) ∧ Airport(SFO) ∧ Airport(ATL) ∧ Airport(ORD))

Goal(At(C1, JFK) ∧ At(C3, JFK) ∧ At(C2, SFO) ∧ At(C4, SFO))

From this information, we use a selection of different search algorithms and test which methods give us the most optimal set of moves to reach the goal state in the shortest amount of time.

Search Algorithms

Search Algorithm Number	Name
1	Breadth First Search
2	Breadth First Tree Search
3	Depth First Graph Search
4	Depth Limited Search
5	Uniform Cost Search
6	Recursive Best First Search with H1 Heuristic
7	Greedy Best First Graph Search with H1 Heuristic
8	A* Search with H1 Heuristic
9	A* Search with H_ignore_preconditions Heuristic
10	A* Search with H_planning_graph_level sum Heuristic

Results

Results for Problem 1

Search Algorithm	Expansions	Goal Tests	New Nodes	Plan Length	Optimal Solution	Time Elapsed in seconds
Breadth First Search	43	56	180	6	Yes	0.031068708000020706
Breadth First Tree Search	1458	1459	5960	6	Yes	1.1829839349999816
Depth First Graph Search	21	22	84	20	No	0.02139601499999344
Depth Limited Search	101	271	414	50	No	0.09165641099997401
Uniform Cost Search	55	57	224	6	Yes	0.039154332000009962
Recursive Best First Search with H1 Heuristic	4229	4230	17023	6	Yes	2.7354172289999497
Greedy Best First Graph Search with H1 Heuristic	7	9	28	6	Yes	0.005522788000007495
A* Search with H1 Heuristic	55	57	224	6	Yes	0.043041897000023255
A* Search with H_ignore_ preconditions Heuristic	41	43	170	6	Yes	0.043351859999972923
A* Search with H_planning_graph_ level sum Heuristic	11	13	50	6	Yes	1.8013326079999956

Results for Problem 2

Search Algorithm	Expansions	Goal Tests	New Nodes	Plan Length	Optimal Solution	Time Elapsed in seconds
Breadth First Search	3343	4609	30509	9	Yes	8.179257624999991
Breadth First Tree Search	-	-	-	-	No	Longer than 10 minutes
Depth First Graph Search	624	625	5602	619	No	3.543490482999914
Depth Limited Search	-	-	-	-	No	Longer than 10 minutes
Uniform Cost Search	4794	4796	43518	9	Yes	11.48640493500011
Recursive Best First Search with H1 Heuristic	-	-	-	-	No	Longer than 10 minutes
Greedy Best First Graph Search with H1 Heuristic	17	19	147	12	No	0.0428933139999117
A* Search with H1 Heuristic	4794	4796	43518	9	Yes	11.697200055000167
A* Search with H_ignore_ preconditions Heuristic	1399	1401	12807	9	Yes	4.4618870520000655
A* Search with H_planning_graph_ level sum Heuristic	94	96	920	9	Yes	219.996714312

Results for Problem 3

Search Algorithm	Expansions	Goal Tests	New Nodes	Plan Length	Optimal Solution	Time Elapsed in seconds
Breadth First Search	14663	18098	129631	12	Yes	41.60328053199987
Breadth First Tree Search	-	-	-	-	No	Longer than 10 minutes
Depth First Graph Search	408	409	3364	392	No	1.822351468000079
Depth Limited Search	-	-	-	-	No	Longer than 10 minutes
Uniform Cost Search	18126	18128	158838	12	Yes	49.76327066700014
Recursive Best First Search with H1 Heuristic	-	-	-	-	No	Longer than 10 minutes
Greedy Best First Graph Search with H1 Heuristic	4464	4466	39243	32	No	13.525112700000136
A* Search with H1 Heuristic	18126	18128	158838	12	Yes	42.115399212998454
A* Search with H_ignore_ preconditions Heuristic	4746	4748	42105	12	Yes	15.391898824999998
A* Search with H_planning_graph_ level sum Heuristic	282	284	2587	12	Yes	876.2282488580071

Non Heuristic Planning Searches

The search algorithms that required no heuristics consisted of Search Algorithms 1 – 5. Of these algorithms, the fastest to arrive at a solution was consistently the Depth First Graph Search (DFGS). DFGS used considerably less resources and time to arrive at a solution, however the plan length typically exceeded the other algorithms by a significant percentage, and thus yielded a sub-optimal end plan. For Problem 1 this was less noticeable, but when faced with more complex problems, the plan length explodes by a considerable amount.

The next fastest non-heuristic search algorithm was Breadth First Search, which was consistently in second place in terms of run time as well as requiring slightly less resources than the comparable Uniform Cost Search. Both Breadth First Search and Uniform Cost Search appeared to reach the same conclusion, which appeared to be the optimal plan as the resultant plan length was matched for the lowest seen. This shows the most optimal solution using the least amount of time and

considering resources among the algorithms 1, 3, and 5 should be Breadth First Search. The conclusions found with this experiment are supported in the AIMA textbook, which notes that Breadth First Search will always find an optimal solution(Norvig 3rd Edition, pg 88), Uniform Cost Search expands nodes in order of optimal path cost resulting in the first goal being the optimal solution. (Norvig 3rd Edition, pg 85)

I also tried some testing with algorithms 2 and 4, Breadth First Tree Search and Depth Limited Search, but both ended up taking an extremely long amount of time to finish. From problem 1's results, I reasoned that algorithm 2 uses significantly more resources than the other 4 algorithms while also completing the path in more time than any of the other 4. This indicated that 2 should be the worst performing algorithm of the non-heuristic set. Algorithm 4 on the other hand used much less resources and time than algorithm 2 but was still beat out by algorithms 3 and 5. Additionally, it would also overshoot the plan length which leads to an evaluation of less than desired performance. These observations are supported by notable sources, which states that depth first tree search is not complete and often repeats states and redundant paths (Norvig 3rd Edition, pg 86). Additionally, depth limited search generally will not find a good depth limit until the problem is already solved and also runs into two different failures; where either no solution is found or no solution within the depth limit is found.

In general, if you are looking to find any solution regardless of how optimal it is and are heavily concerned about system resources, Depth First Search would be the best choice for a non-heuristic approach. However, if the desired results are the optimal solution and you have a good amount of system resources, then Breadth First Search would appear to be the best option.

Heuristic Planning Searches - A* Search

The A* Search algorithms consisted of algorithms 6 – 10 and provided a varying range of performance. Of these algorithms, the worst performing was 6, Recursive Best First Search using the H1 heuristic. This algorithm used significantly more resources than any other algorithm and although it could arrive at a correct answer, it required much more time to complete the search. In contrast, algorithm 7 or greedy best first graph search with the H1 heuristic used considerably less resources and time than the others. However, in problems 2 and 3 it was shown to provide plan lengths that were sub optimal, increasing the deviation as time went on. This observation supports the analysis that Greedy Breadth First search is incomplete (Norvig 3rd Edition, pg 86), and proceeding as along the path of "closest" to the goal nodes will not always evaluate to the optimal solution. While this could be acceptable for very simple problems, the drop in efficiency in relation to difficulty meant that it was not an ideal algorithm.

This left the A* algorithms that utilized different heuristics. These algorithms were consistently able to search to the optimal solution within a timely manner. Of the searches seen by algorithms 8, 9, and 10 or H1, H ignoring preconditions, and the planning graph level sum, the one that performed the best in terms of run time was the ignoring preconditions heuristic. Ignoring preconditions resulted in

the fastest of the A* Searches that consistently gave the optimal solution while also being something of a middle ground in terms of resource usage. While this algorithm did not use as much resources as A* with H1, it still used more than the planning graph algorithm. In comparison, the planning graph took significantly more time than the other two algorithms but used considerably less resources to come to the solution.

It conclusion, if the computer running the search requires low resource usage and does not have a hard requirement for run time, the best option to use would be algorithm 10 or the planning graph. However, if run time is a concern while system resources are not, the ignoring preconditions would then be the best choice. It should be noted that although the planning graph did use up much more time than the ignoring preconditions heuristic, it was still able to find the optimal solution rather quickly when compared to the other algorithms that found no solution in a much longer time.

Optimal Solutions

Problem 1 – Length 6

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 – Length 9

Load(C1, P1, SFO)

Fly(P1, SFO, JFK)

Unload(C1, P1, JFK)

Load(C2, P2, JFK)

Fly(P2, JFK, SFO)

Unload(C2, P2, SFO)

Load(C3, P3, ATL)

Fly(P3, ATL, SFO)

Unload(C3, P3, SFO)

Problem 3 – Length 12

Load(C2, P2, JFK)

Fly(P2, JFK, ORD)

Load(C4, P2, ORD)

Fly(P2, ORD, SFO)

Unload(C4, P2, SFO)

Load(C1, P1, SFO)

Fly(P1, SFO, ATL)

Unload(C2, P2, SFO)

Load(C3, P1, ATL)

Fly(P1, ATL, JFK)

Unload(C3, P1, JFK)

Unload(C1, P1, JFK)

Comparing A* with non-heuristic Search Results

In general, the advantage that A* search appears to possess is the significantly reduced system resource usage. Less nodes need to be made, less expansions need to be performed, and less goals need to be tested when using A* Search. The comparison for these metrics should be considered non-trivial, and for complex problems that require huge state space searching, A* will be significantly better. For these simpler examples, we can see that non-heuristic search can obtain results in a similar amount of time but at the cost of more resources. In addition, the amount of resources used by non-heuristic search appears to explode as complexity increases, making it not viable for more complex or interesting problems. Considering future problems which will be largely more complex than this homework's example, the adoption of A* search especially with the planning graph would be beneficial as it significantly reduces the resources used and maintains a decent run time.