Implement a Planning Search

Christopher Ohara

AIND – Project 3 – Heuristic Analysis

Introduction:

The goal of this project was to design various solutions to an Air Cargo Transport System. The initial portion of the project is to compare common uninformed non-heuristic algorithms for BFS, DFS and UCS. These are compared to informed A* search algorithms with added heuristics to attempt optimization of arrival at goal states.

Three tables are formulated for each of the problem set comparisons. The data reported is the search method, nodes expanded, goal tests, new nodes, execution time (in seconds) and the path length. The optimal path length is defined as a the shortest path (related to cost/loss function). The optimal path length is weighed against the amount of time taken to arrive at the goal. Note: any algorithm that took longer than 15 minutes (the specifications were ten minutes) has it's results omitted, since they were not obtained and are not meaningful outside of comparison.

Some considerations are given for other tasks and embedded systems that might make use of schedulers, memory and path cost with respect to execution time. However, for this project, the goal will always be the results of the optimal path length.

Optimal Sequence of Actions: A* Optimality

Problem 1:

```
Solving Air Cargo Problem 1 using astar_search with h_ignore_preconditions...
Expansions   Goal Tests   New Nodes
    41          43         170
Plan length: 6  Time elapsed in seconds: 0.03788611400523223
Load(C1, P1, SFO)
Fly(P1, SFO, JFK)
Unload(C1, P1, JFK)
Load(C2, P2, JFK)
Fly(P2, JFK, SFO)
Unload(C2, P2, SFO)
```

Problem 2:

```
Solving Air Cargo Problem 2 using astar_search with h_ignore_preconditions...
Expansions   Goal Tests   New Nodes
   1438        1440       13188
Plan length: 9  Time elapsed in seconds: 4.720634611992864

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

Problem 3:

Solving Air Cargo Problem 3 using astar_search with **h_ignore_preconditions...**

14

Expansions	Goal Tests	New Nodes
4912	4914	43692

Plan length: 12 Time elapsed in seconds: 16.633936437996454

Load(C1, P1, SFO)

Fly(P1, SFO, ATL)

Load(C3, P1, ATL)

Fly(P1, ATL, JFK)

Unload(C3, P1, JFK)

Load(C2, P2, JFK)

Fly(P2, JFK, ORD)

Load(C4, P2, ORD)

Fly(P2, ORD, SFO)

Unload(C4, P2, SFO)

Unload(C2, P2, SFO)

Unload(C1, P1, JFK)

Analysis:

Problem 1

Search Method	Expansions	Goal Tests	New Nodes	Execution Time (sec)	Path Length
Breadth First Search	43	56	180	0.037	6
Breadth First Tree Search	1458	1459	5960	0.948	6
Depth First Graph Search	21	22	84	0.015	20
Depth Limited Search	101	271	414	0.091	50
Uniform Cost Search	55	57	224	0.038	6
Recursive Best First Search	4229	4230	17023	2.792	6
Greedy Best First Graph Search	7	9	28	0.005	6
A* Search h1	55	57	224	0.038	6
A* Search h (ignore preconditions)	41	43	170	0.038	6
A* Search h (pg level_sum)	11	13	50	1.066	6

Problem 1 – Results

Uninformed Results:

As can be seen from the *Problem 1 – Results* table, the Greedy Best First Graph Search resulted in the shorted path (six) in only 5 milliseconds. Compared to the other results, this is extremely fast, as it has a minimal amount of expansions. The Depth Limited Search algorithm has the longest path length before completion. Interestingly, the Depth First Graph search, while having a longer path length, operates relatively quickly (15ms) and with a small number of expansions. A final note is that the Recursive Best First Search has the most expansions (4229) and the longest execution time (almost 3 seconds).

Informed Results:

All of the A* algorithms return the optimal path length, as expected based on the AIMA reading and video lectures. The A* Ignore Preconditions heuristic algorithm seems to perform the best. It has 41 expansions with an execution time of 38ms. The Level Sum version has much less expansion (11) but takes over a second to complete. The Level Sum does have appropriate applications, and they will be briefly introduced in the "Problem 2 analysis."

Problem 2

Search Method	Expansions	Goal Tests	New Nodes	Execution Time (sec)	Path Length
Breadth First Search	3343	4609	30509	14.888	9
Breadth First Tree Search	X	X	X	X	X
Depth First Graph Search	624	625	5602	4.245	619
Depth Limited Search	X	X	X	X	X
Uniform Cost Search	4836	4838	43881	12.943	9
Recursive Best First Search	X	X	X	X	X
Greedy Best First Graph Search	867	869	7768	2.389	13
A* Search h1	4836	4838	43881	13.061	9
A* Search h (ignore preconditions)	1438	1440	13188	4.871	9
A* Search h (pg level_sum)	85	87	831	172.744	9

Problem 2 – Results

Uninformed Results:

As can be seen from the *Problem 2– Results* table, the optimal path length is 9. However, both of the algorithms that arrive there (BFS and UCS) require the most time, at approximately 15 and 13 seconds, respectively. This is because they both have the highest number of expansions. The fastest execution time, at only 2.4 seconds is the Greedy BFGS again. However, it does not return the shortest path length. Therefore, depending on the situation, a trade-off should be made. Whenever the engineer is designing a system, trade-offs will need to made with respect to cost of path, execution time and memory. For an embedded systems MCU platform (like Arduino), the least algorithm that uses the least amount of resources is needed. A real time scheduler is going to need to the fastest time for embedded systems (reflective of a real air transportation cargo system). However, in this situation, since we are considering cargo moving large distances, we really need the shortest path.

Informed Results:

Once again, the Ignore Preconditions algorithm comes out on top. All three A* searches return the optimal path length, but ignoring the preconditions (flexible/relaxed) reduces the time to approximately 5 seconds by only completing around 1.4k expansions. Again, the Level Sum option uses the least amount of expansions, but at a time of over three minutes. Again, different applications require different specifications. With that regard, it would be important to state that in military or IoT web connectivity systems and situations, the Level Sum option might be chosen for security reasons (though the details of this are outside of the scope of this project).

Problem 3

Search Method	Expansions	Goal Tests	New Nodes	Execution Time (sec)	Path Length
Breadth First Search	14663	18098	129631	110.322	12
Breadth First Tree Search	X	X	X	X	X
Depth First Graph Search	408	409	3364	1.79	392
Depth Limited Search	X	X	X	X	X
Uniform Cost Search	18078	18080	158424	60.162	12
Recursive Best First Search	X	X	X	X	X
Greedy Best First Graph Search	4827	4829	42754	13.974	26
A* Search h1	18078	18080	158424	52.206	12
A* Search h (ignore preconditions)	4912	4914	43692	16.634	12
A* Search h (pg level_sum)	254	256	2324	745.166	12

Problem 3– Results

Uninformed Results:

As can be seen from the *Problem 3– Results* table, the optimal path length is 12. With the added complexity, the results are beginning to become more noticeable, which was probably the intention of the ND creators. The DFGS has the fastest execution time (1.79s) with only 408 expansions, but at a path length cost of 392. This results in the least optimal path. There are two options that return the optimal path (BFS and UCS) but with a trade-off of time and expansions. The UCS is the faster algorithm at one minute, but with 4k more expansions. All things considered, I think the most robust and optimal choice is going to be the UCS algorithm, since it will be more flexible for more embedded system tasks while having the optimal path and a better execution time than BFS. If path cost optimality is a minor concern, then the Greedy BFGS is a good choice (but things are going to be going to twice as many airports...).

Informed Results:

Once again, the Ignore Preconditions algorithm is going to be the best choice for a system where speed and path cost optimality are important. It returns the optimal path length of 12 in 16.6 seconds. The Level Sum has the least amount of expansion (254) but actually compiled in more than 12 minutes. In systems where execution time is pivotal (this is supposed to be an air traffic cargo system) we have to closely consider the specifications.

Conclusion:

In conclusion, the optimal algorithm is going to reflect the shortest path with the smallest execution time for an Air Transportation Cargo System. The A* searches guarantee a return of the optimal path (if one exists). With all things considered, I think the A* Ignore Preconditions heuristic is the best choice for this problem. Chapter 10 in the AIMA text by Norvig and Russell discuss reasons why a "relaxed" constraint algorithm can have a better performance. An important topic in machine learning and AI is following a path without "under-fitting or over-fitting." In this case, estimating where the goal will be provides the best chance at the heuristics minimizing the path cost.