Heuristic Analysis

Dovydas Čeilutka

January 6, 2018

In this analysis an optimal plan for Problems 1, 2 and 3 is provided.

1 Optimal plans

The optimal plans for the three problems was found using `run_search.py` script.

1.1 Optimal plan for Problem 1

The optimal plan for Problem 1 is:

1. Load(C1, P1, SFO)
2. Load(C2, P2, JFK)
3. Fly(P2, JFK, SFO)
4. Unload(C2, P2, SFO)
5. Fly(P1, SFO, JFK)
6. Unload(C1, P1, JFK)

The plan has a length of 6.

1.2 Optimal plan for Problem 2

The optimal plan for Problem 2 is:

1. Load(C2, P2, JFK)
2. Load(C1, P1, SFO)
3. Load(C3, P3, ATL)
4. Fly(P2, JFK, SFO)
5. Unload(C2, P2, SFO)
6. Fly(P1, SFO, JFK)
7. Unload(C1, P1, JFK)
8. Fly(P3, ATL, SFO)
9. Unload(C3, P3, SFO)

The plan has a length of 9.

1.3 Optimal plan for Problem 3

The optimal plan for Problem 3 is:

1. Load(C2, P2, JFK)
2. Load(C1, P1, SFO)
3. Fly(P2, JFK, ORD)
4. Load(C4, P2, ORD)
5. Fly(P1, SFO, ATL)
6. Load(C3, P1, ATL)
7. Fly(P1, ATL, JFK)
8. Unload(C1, P1, JFK)
9. Unload(C3, P1, JFK)
10. Fly(P2, ORD, SFO)
11. Unload(C2, P2, SFO)
12. Unload(C4, P2, SFO)

The plan has a length of 12.

2 Comparison of results for the non-heuristic search strategies

2.1 Problem 1

Ability to find optimal plan Most of the search strategies for the Problem 1 find the optimal plan - depth first graph search and depth limited search find plans that are not optimal.

Speed Most of the search strategies find the solution to Problem 1 within 1 second on a modern laptop. Recursive best first search takes much longer than other asearch strategies and finds the optimal solution in 2.5 seconds.

Overall performance Greedy best first graph search finds the optimal solution, expands fewest nodes, does fewest node tests, generates fewest new nodes and runs fastest, therefore it has the best overall performance for Problem 1.

Table 1: Comparison of the non-heuristic search strategy performance on Problem 1

Search Strategy	Plan Length	Expansions	Goal Tests	New Nodes	Time
Breadth First Search	6	43	56	180	0.03
Breadth First Tree Search	6	1458	1459	5960	0.86
Depth First Graph Search	12	12	13	48	0.01
Depth Limited Search	50	101	271	414	0.09
Uniform Cost Search	6	55	57	224	0.04
Recursive Best First Search	6	4229	4230	17029	2.53
Greedy Best First Graph Search	6	7	9	28	0.01

2.2 Problem 2

Ability to find optimal plan Only two search strategies - breadth first search and uniform cost search - find optimal plan for Problem 2 within a reasonable timeframe (2 hours) running on a modern laptop.

Speed Most of the search strategies manage to find the solution to Problem 1 within 2 hours running on a modern laptop. However, the difference in speed between the search strategies is very clear - greedy best first graph search is by far the fastest of all search strategies, but it didn't manage to find the optimal path. Out of the two search strategies, which managed to find the optimal path, both breadth first search and uniform cost search are very close in terms of speed, but uniform cost search is a bit faster.

Overall performance The greedy best first graph search finds the expands fewest nodes, does fewest node tests, generates fewest new nodes and runs fastest, but doesn't find the optimal solution. If speed and memory usage are important factors and finding the optimal solution is not necessary, then it is the best search strategy.

In case where finding the optmal solution is necessary, breadth first search or uniform cost search strategies are very close in all the performance metrics, therefore either one can be used.

Table 2: Comparison of the non-heuristic search strategy performance on Problem 2

Search Strategy	Plan Length	Expansions	Goal Tests	New Nodes	Time
Breadth First Search	9	3343	4609	30509	13.12
Breadth First Tree Search	NA	NA	NA	NA	NA
Depth First Graph Search	1444	1669	1670	14863	13.21
Depth Limited Search	50	222719	2053741	2054119	1561.31
Uniform Cost Search	9	4852	4854	44030	12.65
Recursive Best First Search	NA	NA	NA	NA	NA
Greedy Best First Graph Search	21	990	992	8910	2.52

2.3 Problem 3

Ability to find optimal plan The results are very similar as in Problem 2 - only two search strategies - breadth first search and uniform cost search - find optimal plan for Problem 3 within a reasonable timeframe (2 hours) running on a modern laptop.

Speed Four strategies managed to find the solution within a reasonable timeframe (2 hours) running on a modern laptop and three failed. The depth first graph search is by far the fastest, but didn't find the optimal path. Out of the two search strategies, which managed to find the optimal path, uniform cost search is about 30% faster than breadth first search.

Overall performance The depth first graph search finds the expands fewest nodes, does fewest node tests, generates fewest new nodes and runs fastest, but doesn't find the optimal solution. If speed and memory usage are important factors and finding the optimal solution is not necessary, then it is the best search strategy.

In case where finding the optmal solution is necessary, uniform cost search strategy is faster than breadth first search, but expand and generate more nodes, therefore the strategy should be chosen based on what is more important - speed or memory. The fact that the uniform cost search is faster than the breadth first search is rather surprising as in the analysed cases where the step cost is the same, the uniform cost search should be identical to breadth cost search with an exception of the how the node epansion works (uniform cost expands more nodes). This suggests that uniform cost search should be slower than breadth first search. The proposed explanation for this weird result is that the actual implementation of the both strategies are causing this issue.

Table 3: Comparison of the non-heuristic search strategy performance on Problem 3

Search Strategy	Plan Length	Expansions	Goal Tests	New Nodes	Time
Breadth First Search	12	14663	18098	129631	150.25
Breadth First Tree Search	NA	NA	NA	NA	NA
Depth First Graph Search	571	592	593	4927	4.04
Depth Limited Search	NA	NA	NA	NA	NA
Uniform Cost Search	12	18235	18237	159716	104.76
Recursive Best First Search	NA	NA	NA	NA	NA
Greedy Best First Graph Search	22	5614	5616	49429	34.95

3 Comparison of results for heuristic search result strategies

3.1 Problem 1

Ability to find optimal plan All of the heuristic search strategies find the optimal plan.

Speed A* with search level sum heuristic is the slowest of the three algorithms, but still manages to find the optimal solution in less than 1 second.

Overall performance All of the heuristic search strategies find the optimal plan. A* with search level sum heuristic is the most memory efficient, but slowest. A* search with H1 heuristic and A* search with ignore preconditions heuristic are much faster, but use about five times more memory.

Table 4: Comparison of the non-heuristic search strategy performance on Problem 1

Search Strategy	Plan Length	Expansions	Goal Tests	New Nodes	Time
A* Search H1	6	55	57	224	0.04
A* Search Ignore Preconditions	6	41	43	170	0.04
A* Search Level Sum	6	11	13	50	0.64

3.2 Problem 2

Ability to find optimal plan All of the heuristic search strategies find the optimal plan.

Speed A* search with ignore preconditions heuristic is the fastest, A* search with H1 heuristic is about two and a half times slower and A* with search level sum heuristic is about seventeen times slower.

Overall performance All of the heuristic search strategies find the optimal plan. A* search with ignore preconditions heuristic is the fastest, but also uses twenty times more memory than A* with search level sum heuristic, which is the most memory efficient, but slowest. A* search with H1 heuristic uses the most memory and is the second fastest.

Table 5: Comparison of the non-heuristic search strategy performance on Problem 2

Search Strategy	Plan Length	Expansions	Goal Tests	New Nodes	Time
A* Search H1	9	4852	4854	44030	13.18
A* Search Ignore Preconditions	9	1450	1452	13303	5.07
A* Search Level Sum	9	86	88	841	85.84

3.3 Problem 3

Ability to find optimal plan All of the heuristic search strategies find the optimal plan.

Speed A* search with ignore preconditions heuristic is the fastest, A* search with H1 heuristic is about two and a half times slower and A* with search level sum heuristic is about fourteen times slower.

Overall performance All of the heuristic search strategies find the optimal plan. A* search with ignore preconditions heuristic is the fastest, but also uses fifteen times more memory than A* with search level sum heuristic, which is the most memory efficient, but slowest. A* search with H1 heuristic uses the most memory and is the second fastest.

Table 6: Comparison of the non-heuristic search strategy performance on Problem 3

Search Strategy	Plan Length	Expansions	Goal Tests	New Nodes	Time
A* Search H1	12	18235	18237	159716	116.88
A* Search Ignore Preconditions	12	5040	5042	44944	41.97
A* Search Level Sum	12	318	320	2934	594.95

4 Comparison of results for all search result strategies

4.1 Ability to find optimal plan

All of the heuristic search strategies find the optimal plan, but only two of non-heuristic search strategies do: breadth first search and uniform cost search.

4.2 Speed

A* search with ignore preconditions heuristic is the fastest of the heuristic search strategies, while uniform cost search is the fastest of the non-heuristic search strategies.

Overall performance While comparing the heuristic vs non-heuristic search strategies it is clear that heuristic strategies are supperior to non-heuristic ones for all problems. A* search with ignore preconditions heuristic is several times faster and memory efficient than the uniform cost search (the best non-heuristic strategy). In case where memory efficiency is very important, A* level sum search does fewest node expansions and creates fewest nodes significantly outperforming any non-heuristic strategy (including non-optimal ones).

5 Conclusion and recommendations

The detailed analysis of the search strategies showed that the heuristic search strategies have a clear advantage. Based on the analysis done in this paper A* search with ignore preconditions heuristic is determined to be the best search strategy for all cases, except when memory efficiency is very important, where A* level sum search is the best.