Analysis of Heuristic and Non-Heuristic Approaches to Classical Planning in Project #3

Steven Bogacz

September 2017

1 Problems

1.1 Problem #1

1.1.1 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))
```

1.1.2 Solution

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

1.2 Problem #2

1.2.1 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))
```

1.2.2 Solution

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

```
Unload(C2, P2, SFO)
```

1.3   Problem #3

1.3.1   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))
```

1.3.2   Solution

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

2   Comparing and Contrasting Search Metrics

The comprehensive metrics collected over the course of this project are shown below. I made an effort to collect metrics for every combination of problem and search method, however, excessive runtime prevented me from exhausting every combination. Those that were unable to run within a reasonable time frame are marked as "N/A".

Problem	Search Function	Expansions	Goal Tests	New Nodes	Plan Length	Time Elapsed (s)
Air Cargo #1	BFS	43	56	180	6	0.5455
	BFTS	1458	1459	5960	6	11.5311
	DFGS	21	22	84	20	0.2319
	DLS	101	271	414	50	1.4493
	UCS	55	57	224	6	0.68
	RBFS (H1)	4229	4230	17023	6	33.3754
	GBFS (H1)	7	9	28	6	0.061
	A*(H1)	55	57	224	6	0.6538
	A*(H Ignore Preconditions)	41	43	170	6	0.3376
	A*(H PG Levelsum)	11	13	50	6	10.8796
Air Cargo #2	BFS	3343	4609	30509	9	133.5315
	BFTS	N/A	N/A	N/A	N/A	N/A
	DFGS	624	625	5602	619	38.4431
	DLS	222719	2053741	2054119	50	4429.4931
	UCS	4852	4854	44030	9	120.0568
	RBFS (H1)	N/A	N/A	N/A	N/A	N/A
	GBFS (H1)	990	992	8910	21	27.3664
	A*(H1)	4852	4854	44030	9	120.9446
	A*(H Ignore Preconditions)	1450	1452	13303	9	48.7596
	A*(H PG Levelsum)	86	88	841	9	1070.4881
Air Cargo #3	BFS	14663	18098	129631	12	661.2725
	BFTS	N/A	N/A	N/A	N/A	N/A
	DFGS	408	409	3364	392	20.4477
	DLS	N/A	N/A	N/A	N/A	N/A
	UCS	18235	18237	159716	12	395.2189
	RBFS (H1)	N/A	N/A	N/A	N/A	N/A
	GBFS (H1)	5613	5315	49420	21	153.8858
	A*(H1)	18235	18237	159716	12	397.1612
	A*(H Ignore Preconditions)	5040	5042	44944	12	159.8279
	A*(H PG Levelsum)	321	323	2965	12	4826.918

2.1 Non-heuristic Search Result Comparison

In this project we used 5 different non-heuristic search methods (with their respective abbreviations listed parenthetically):

- Breadth First Search (BFS)

- Breadth First Tree Search (BFTS)

- Depth First Graph Search (DFS)

- Depth Limited Search (DLS)

- Uniform Cost Search (UCS)

Perhaps it is easier to discuss the search methods starting by those which are demnonstrably less efficient. Breadth First tree search was one of the two search methods to fall over first (it failed to complete for both problems two and three). Both it and Recursive Best First Search required disproportionately more node expansions than any other search method. Problem #1 was the only one where heuristics could be collected for those two, and they expanded 13 and 42 times as many nodes as the next biggest expander (depth-limited search).

Depth-limited search and Depth-First Graph Search both returned sub-optimal plans, as is expected, as they are known to be neither complete, nor optimal (a summary can be found in Russel & Norvig Section

3.4.7). On the other hand, Breadth-First Search and Uniform Cost Search not only completed for every test case, but they also involved the least expansions out of the other non-heuristic based search methods. BFS was more computationaly expensive, taking more time on problems #2 and #3, while UCS took up more memory with more expansions, but took less time to arrive to a solution.

Ultimately the choice of uninformed search strategy would depend on the problem constraints. DFGS could be a good choice in situations where we need not guarantee an optimal solution while at the same time dealing with resource constraints. On the other hand, if resources are accessible, and optimality is desired, then UCS provides an optimal solution significantly faster than BFS, the other complete and optimal search strategy.

2.2   Heuristic Search Result Comparison

The five heuristic based search methods we used are listed below:

- Recursive Best First Search H 1 (RBFS)

- Greedy Best First Graph Search H 1 (GBFS)

- A* Search H 1 (A*)

- A* Search H Ignore Preconditions (A*)

- A* Search H PG Level Sum (A*)

RBFS was already discussed in the previous section, so I will focus on the other four methods here. Interestingly, all four were able to finish within a reasonable time frame, although there are significant tradeoffs to be considered. Greedy Best First Search and A* with the Ignore Preconditions Heuristic were the two fastest methods. GBFS has a substantial edge for smaller state spaces, but that advantage dissipates rapidly as the state space grows.

GBFS did not, however, find an optimal solution in the case of Problem #3. A* using the Level Sum Heuristic took far longer to compute (about an order of magnitude) but also expanded far less nodes, by about a factor of ten, which makes sense, as it will give graeater weight to paths that are more rapidly converging on a solution for all the goals in the goal state. Ultimately, it seems that the trade-off would be situation dependent: are we more constrained by available computation time, or are we running in a memory-limited environment?

Overall, the heuristic-based search methods were generally more performant than the non-heuristic based search methods. Even the Level Sum heuristic, despite having a significantly longer run time than the two optimal non-heuristic searches, BFS and UCS, came at a far lesser memory cost. A* with the Ignore Preconditions Heuristic not only arrived to the same optimal solution, but also did so in a quarter and half the time than each BFS and UCS respectively. Looking at the trend, it seems clear that this outperformace would grow along with the size of the state space.