Optimal Search per Problem

For all of the below, given our problem definition, we could switch the order of various actions and still be an optimal result. These are merely examples produced by A* Search with an Ignore Precondition Heuristic

Problem 1: Plan Length = 6

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

Problem 2: Plan 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: Plan Length = 12

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

Search Comparison

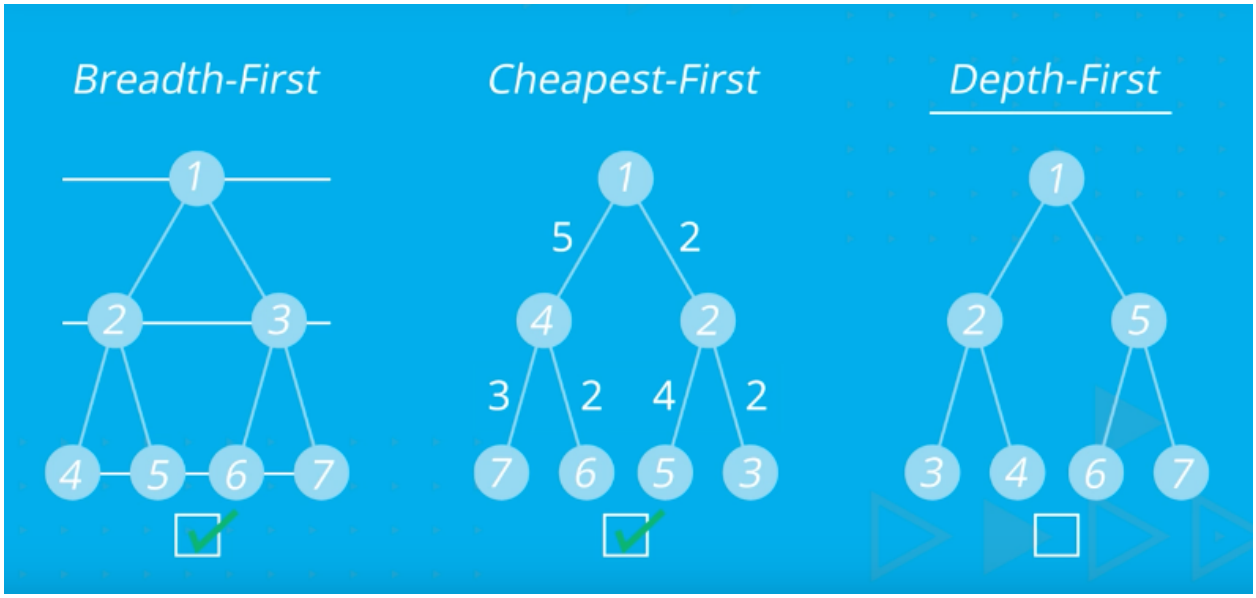
Below, I have a table with the detailed comparison of each of the searches.

Uninformed Search Review: the uninformed searches are Depth First Graph Search (DFS), Breadth First Search (BFS), and Uniform Cost Search (UCS). A_Star_H1 can be considered uninformed because the H is constant – meaning it is essentially the same as UCS, but I cover it during the informed search review section below. The key uninformed searches are:

1) **Depth First Search** is a non-optimal and incomplete search. It is, for this problem domain with solvable and non-infinite searches possible, far and away the least costly uninformed search, in terms of expansions, goal tests, new nodes, and thus performance, across all problems. This is because it only searches along the first path picked until it hits a goal or an end-point, even if that is a very inefficient path. *In Lesson 10: Chapter 23 Norvig covers why this happens – Depth-First Search is not optimal, it is*

not guaranteed to find the cheapest cost search. If there were goals in position 5, below, DFS would miss it. This is exactly what is happening in the planning problems.

2) **Breadth First Search** is an optimal and complete search which produces an optimal path no matter where the goal is. It utilizes fewer expansions than goal tests. This is because it follows the path simply left to right, and expands one level at a time. Once it goal tests a solution, it will stop and not need to expand any additional nodes. For example, if a solution is at node 4 in the below, it will not goal test 5, 6, or 7. The relative difference between goal tests and expansions is essentially how "early" in the last path level BFS found the result.
3) **Uniform Cost Search:** This is an optimal and complete search. It is also in general, an efficient uninformed search. This is because instead of just treating each path level as equal, it expands deeply by using a valuation of cost. In this case, though, the only measure of cost is path level, so Cheapest-First is essentially equivalent to breadth first, except in the specific implementations used.

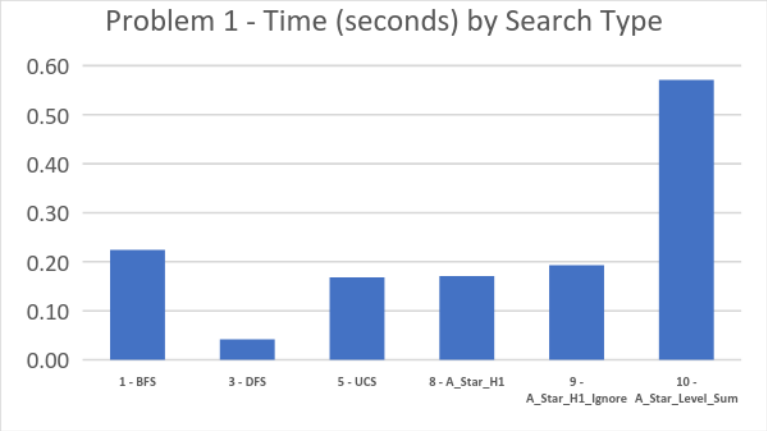


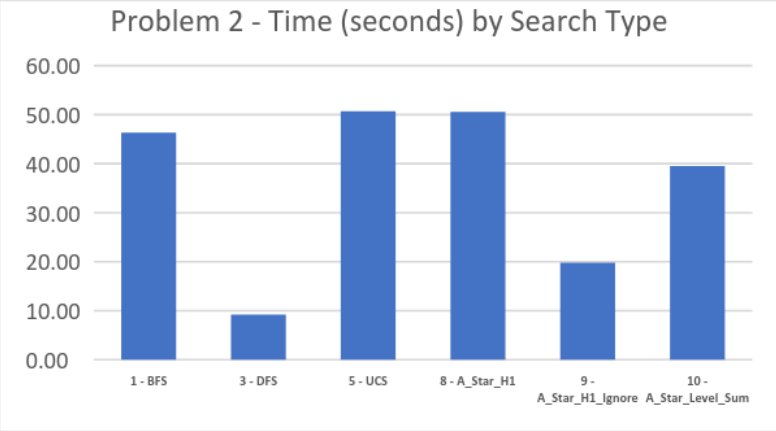
Informed Search Review: In this section I compare various A* Search (described as A_Star) heuristics. Each are described below:

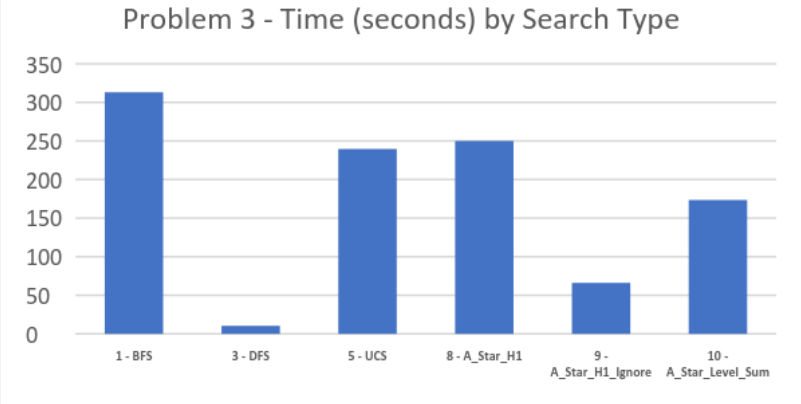
- **A_Star_H1:** This uses a fixed value for H, which means that A* Search is essentially an uninformed search. The additional H value is not any heuristic that is helping us reach our goal. Thus, we see results that look exactly equivalent to our Uniform Cost Search
- **A_Star_H1_Ignore_Preconditions:** This heuristic makes the search significantly more efficient. It heuristically evaluates each node based on how many of the goal states are missing in this node's state, ignoring any requirements to perform that action. In this context, it is saying, "for all of the locations cargo needs to be, how many aren't there?" ignoring all the steps of loading a plane, flying it, and unloading it. This is a fast heuristic to tell you if you are moving along the correct path, and eliminates paths that haven't accomplished parts of the goal. It dramatically reduces evaluation time for a blanket A* / UCS by leading you through path's that have successfully delivered on some goals.

- **A_Star_Level_Sum:** This heuristic compared with A_Star_H1_Ignore_Preconditions provides an excellent comparison of the relative benefits of a complex heuristic with eliminating search paths. The Level Sum Heuristic is a very complex heuristic to calculate – because it is evaluating the depth of each goal state. Because of this, as you can see, while the number of expansions, goal tests, and new nodes is dramatically lower than h_1_precondition, the time it takes to calculate is still far higher. Further, A_Star_Level_Sum does not guarantee an optimal solution if the problem is not subgoal independent. In Problem 3, we see this, where we have 4 cargoes and 2 planes, so we actually produce a suboptimal result.
- *Overall, I think the best heuristic is A_Star_H1_Ignore_Preconditions.* It calculates an optimal solution, very efficiently in terms of time and number of nodes. The heuristic is relatively quick to calculate, but still moves us in the right direction in the problem.

Heuristic Analysis – Jon Katzur







Problem	Search	Expansions	Goal Tests	New Nodes	Plan Length	Time
1	1 - BFS	43	56	180	6	0.22
1	3 - DFS	12	13	48	12	0.04
1	5 - UCS	55	57	224	6	0.17
1	8 - A_Star_H1	55	57	224	6	0.17
1	9 - A_Star_H1_Ignore	41	43	170	6	0.193
1	10 - A_Star_Level_Sum	11	13	50	6	0.571
2	1 - BFS	3,343	4,609	30,509	9	46.32
2	3 - DFS	582	583	5,211	575	9.22
2	5 - UCS	4,605	4,607	41,839	9	50.67
2	8 - A_Star_H1	4,605	4,607	41,839	9	50.57
2	9 - A_Star_H1_Ignore	1311	1313	11989	9	19.77
2	10 - A_Star_Level_Sum	74	76	720	9	39.53
3	1 - BFS	14,663	18,098	129,631	12	313.16
3	3 - DFS	627	628	5,176	596	10.40
3	5 - UCS	16,961	16,963	149,117	12	239.72
3	8 - A_Star_H1	16,961	16,963	149,117	12	250.10
3	9 - A_Star_H1_Ignore	4444	4446	39227	12	66.08
3	10 - A_Star_Level_Sum	229	231	2081	13	173.47