Heuristic Analysis

Problem 1

#	Search Method	*Processing Time*	*Plan Length*	*Expansions*	*Goal Test*	*New Nodes*
1	Breadth-First	0.054	6	43	56	180
2	Breadth-First Tree	1.528	6	1,458	1,459	5,960
3	Depth-First Graph	0.013	12	12	13	48
4	Depth Limited	0.133	50	101	271	414
5	Uniform Cost	0.060	6	55	57	224
6	Recursive Best First with *h_1*	4.446	6	4,229	4,230	17,029
7	Greedy Best First with *h_1*	0.009	6	7	9	28
8	A* with *h_1*	0.062	6	55	57	224
9	A* with *h_ignore_preconditions*	0.047	6	41	43	170
10	A* with *h_peg_levelsum*	1.940	6	41	43	167

Methods

Non-heuristic

Depth-first methods are fast but very ineffective. According to Norvig and Russel, "depth-first is neither complete not optimal, but has linear space complexity" [1]. That explains these results.

Breadth-First and the Uniform Cost searches are capable of finding optimal solutions, with good timing. However, the good performance on speed is probably because the search space if very small, as according to Norvig and Russel, "Breadth-first is complete and optimal for unit step costs, but has exponential space complexity." [1] So for larger spaces, this method will probably take a lot more time to process.

Heuristic

All methods deliver optimal results, with a significant speed advantage for Greedy Best First graph search. However, according to Patel, "Greedy Best-First-Search is *not* guaranteed to find a shortest path. However, it runs much quicker (…) because it uses the heuristic function to guide its way towards the goal very quickly." [2] So, like in the Depth-First case above, the success of this algorithm is probably to the small search space.

Recursive Best First search was the most inefficient of them, with lots of expansions and long processing time. According to Berry, "one major flaw of this algorithm is that it can visit the same node several times. This occurs due to the algorithm changing paths only to come back to the same path again. It is understandable how this can happen as once a node is expanded there is a good chance its current f-value will become worse." [3]

Optimal Solution

The best solution is the ***Greedy Best First Graph Search with h_1***.

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

Problem 2

#	Search Method	Processing Time	Plan Length	Expansions	Goal Test	New Nodes
1	Breadth-First	17.506	9	3,343	4,609	30,509
2	Breadth-First Tree	*too long*	-	-	-	-
3	Depth-First Graph	3.847	575	582	583	5,211
4	Depth Limited	*too long*	-	-	-	-
5	Uniform Cost	18.543	9	4,852	4,854	44,030
6	Recursive Best First with *h_1*	*too long*	-	-	-	-
7	Greedy Best First with *h_1*	3.629	21	990	992	8,910
8	A* with *h1*	17.903	9	4,852	4,854	44,030
9	A* with *h_ignore_preconditions*	5.956	9	1,450	1,452	13,303
10	A* with *h_peg_levelsum*	201.928	9	86	88	841

Methods

Non-heuristic

Again, Breadth-First and the Uniform Cost yield the best results, although not the best timing. As pointed before, the exponential increase of the search space of Breadth-First will consume more and more time.

As for Uniform Cost search, according to Norvig and Russel, this algorithm is a modification of breadth-first search that *"instead of expanding the shallowest node, [it] expands the node with the lowest path cost."* [1] So it is not a surprise they have similar performances.

Heuristic

Among the Heuristic methods, only the ones based in A* were effective. According to Norvig and Russel, "The algorithm is identical to Uniform Cost search except that A* uses g + [heuristic] instead of g."

Hence, it is expected that will perform equal or better than Uniform Search, depending on the heuristic being used. And in fact, the heuristic *h1* - which is not an heuristic at all - yield exactly the same results of Uniform Cost search. On the other hand, as demonstred by case #10 with "A* with h_peg_levelsum", the heuristic might increase processing time.

Greedy Best First had a bad plan output, which is consistent with our observation above that the increase of the search space will make this algorithm more ineffective.

Optimal Solution

The best solution is the ***A* with h_ignore_preconditions*** heuristics. The algorithm was fast with an efficient plan. The A* with the heuristics *h_peg_levelsum* was efficient but took too long.

- 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

#	Search Method	***Processing Time***	***Plan Length***	***Expansions***	***Goal Test***	***New Nodes***
1	Breadth-First	130.050	12	14,663	18,098	129,631
2	Breadth-First Tree	*too long*	-	-	-	-
3	Depth-First Graph	4.464	596	627	628	5,176
4	Depth Limited	*too long*	-	-	-	-
5	Uniform Cost	87.572	12	18,223	18,225	159,618
6	Recursive Best First with *h_1*	*too long*	-	-	-	-
7	Greedy Best First with *h_1*	25.894	22	5,578	5,580	49,150
8	A* with *h1*	91.697	12	18,223	18,225	159,618
9	A* with *h_ignore_preconditions*	24.554	12	5,040	5,042	44.944
10	A* with *h_peg_levelsum*	1,097.362	12	325	327	3,002

Methods

Non-heuristic

As before, Breadth-First and Uniform Cost found a good plan, but they took significantly more time than most Heuristics methods.

Heuristic

As before, the A* methods outperformed. According to Perter Norvig, on A*, on one hand minimizing the cost G helps to keep the path to the goal short, and on the other hand minimizing the heuristic H keeps the search focused on the goal itself. *"The result is a search strategy that is the best possible, in the sense that it finds the shortest length path while expanding a minimum number of paths possible".* [4]

Optimal Solution

The best solution is again the ***A* with h_ignore_preconditions*** heuristics. Again, the A* with the heuristics *h_peg_levelsum* was efficient too but took a lot more time to calculate.

According to Poole and Mackworth, *"Typically a trade-off exists between the amount of work it takes to derive a heuristic value for a node and how accurately the heuristic value of a node measures the actual path cost from the node to a goal."* [5] This is exactly the issue with *h_peg_levelsum:* although accuracy is great, the amount of work on its calculation is quite high.

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

References

[1] Norvig, Russel. “Artificial Intelligence A Modern Approach” 3rd edition. Prentice Hall Series. Uppper Saddle River, New Jersey, 2010.

[2] Amit Patel, http://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html

[3] Chris Berry, http://aicat.inf.ed.ac.uk/entry.php?id=666

[4] Peter Norvig, Udacity’s AIND classes, Lesson 11: Search and Lesson 15: Planning

[5] Poole and Mackworth, Artificial Intelligence: Foundations of Computational Agents, 2nd Edition, http://artint.info/html/ArtInt_56.html