Heuristic Analysis

Optimal Plans

Problem 1:

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))
Load(C1, P1, SFO)
Load(C2, P2, JFK)
Fly(P2, JFK, SFO)
Unload(C2, P2, SFO)
Fly(P1, SFO, JFK)
Unload(C1, P1, JFK)
Goal(At(C1, JFK) ∧ At(C2, SFO))

Problem 2:

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))
Load(C1, P1, SFO)
Load(C2, P2, JFK)
Load(C3, P3, ATL)
Fly(P1, SFO, JFK)
Fly(P2, JFK, SFO)
Fly(P3, ATL, SFO)
Unload(C3, P3, SFO)
Unload(C2, P2, SFO)
Unload(C1, P1, JFK)
Goal(At(C1, JFK) ∧ At(C2, SFO) ∧ At(C3, SFO))

Problem 3:

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))
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)
Goal(At(C1, JFK) ∧ At(C3, JFK) ∧ At(C2, SFO) ∧ At(C4, SFO))

Non-heuristic search result metrics

The three non-heuristic search types we'll be comparing are:

- Breadth First Search
- Depth First Search
- Uniform Cost Search

Problem 1:

Algorithm	Plan Length	Expansions	Goal Tests	New Nodes	Time(in seconds)
Breadth First Search	6	43	56	180	0.10999
Depth First Search	20	21	22	84	0.0520127
Uniform Cost Search	6	55	57	224	0.1303401

Problem 2:

Algorithm	Plan Length	Expansions	Goal Tests	New Nodes	Time(in seconds)
Breadth First Search	9	3343	4609	30509	26.979027
Depth First Search	619	624	625	5602	10.5665068
Uniform Cost Search	9	4852	4854	44030	34.77713539

Problem 3:

Algorithm	Plan Length	Expansions	Goal Tests	New Nodes	Time(in seconds)
Breadth First Search	12	14663	18098	129631	125.037976
Depth First Search	392	408	409	3364	4.46308432
Uniform Cost Search	12	18223	18225	159618	77.88214778

Depth First Search had the highest plan length compared to Breadth First Search and Uniform Cost Search, but used the lowest expansions, goal tests and new nodes which resulted in less execution time. On comparing Breadth First Search to Uniform Cost Search there is a relatively small difference between the expansions and goal tests, but there is a sizable difference in the number of new nodes.

So it can be clearly seen that while Breadth First Search and Uniform Cost Search take more time than Depth First Search, they find the more optimal plans.

Heuristic search result metrics

The three non-heuristic search types we'll be comparing are:

- A* with h_1
- A* with h_ignore_preconditions
- A* with h_pg_levelsem

Problem 1:

Algorithm	Plan Length	Expansions	Goal Tests	New Nodes	Time(in seconds)
A* with h_1	6	55	57	224	0.13373031
A* with h_ignore	6	41	43	170	0.13619042
A* with h_pg_levelsum	6	11	13	50	1.42397087

Problem 2:

Algorithm	Plan Length	Expansions	Goal Tests	New Nodes	Time(in seconds)
A* with h_1	9	4852	4854	44030	34.0807011
A* with h_ignore	9	1450	1452	13303	11.410158199
A* with h_pg_levelsum	9	86	88	841	132.31125526

Problem 3:

Algorithm	Plan Length	Expansions	Goal Tests	New Nodes	Time(in seconds)
A* with h_1	12	18223	18225	159618	142.2129989
A* with h_ignore	12	5040	5042	44944	57.0367131992
A* with h_pg_levelsum	12	315	317	2902	626.2212973

Of the three searches, A* with h_1 uses significantly more(relatively) expansions, goal tests and new nodes, while on average taking the second most time for execution. The A* with h_ignore_preconditions proves what was said in the class lectures, that **relaxing the constraints leads to less execution time** while also being a good search algorithm. And finally, A* with h_pg_levelsem finds the optimal solution with significantly lesser number of expansions, goal tests and new nodes, but has significantly **more** time overhead.

Best Heuristic

The best heuristic of the three is **A* with h_ignore_preconditions** as it reaches the optimal solution the quickest compared to the three while not doing as much expansion as 'A* with h1'**.** This happens because of the **relaxation of constraints** (explained in the Search section of the AIND videos) that allows 'A* with h_ignore_preconditions' to solve a simpler problem than the one 'A* with h_pg_levelsum' is trying to solve (which makes 'A* with h_pg_levelsum' the slowest search). Compared to the non heuristic searches, 'A* with h_ignore_preconditions' performs better than 'Breadth First Search' and 'Uniform Cost Search', but takes slightly more time than the fastest uninformed search, the 'Depth First Search'. This is because the given problem is simple, hence 'Depth First Search' can simply apply actions till it reaches a suboptimal plan in a very short amount of time.