HEURISTIC ANALYSIS

OPTIMAL SOLUTION FOR PROBLEMS

Problem 1:

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

Problem 2:

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

Problem 3:

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

SEARCH RESULTS

Problem 1:

Method	Expansions	Goal Tests	New Nodes	Time	Plan length
breadth_first_search	43	56	180	0.041	6
breadth_first_tree_search	1458	1459	5960	1.132	6
depth_first_graph_search	21	22	84	0.018	20
depth_limited_search	101	271	414	0.115	50
uniform_cost_search	55	57	224	0.061	6
recursive_best_first_search with h_1	4229	4230	17023	3.750	6
greedy_best_first_graph_search with h_1	7	9	28	0.007	6
astar_search with h_1	55	57	224	0.057	6
astar_search with h_ignore_preconditions	41	43	170	0.050	6
astar_search with h_pg_levelsum	11	13	50	0.637	6

For problem 1, the best method is “greedy_best_first_graph_search with h_1” as is spend the least time and use the least expansion, goal test and new nodes. Besides this method, “breadth_first_search” and “astar_search” with heuristics also did a good job to find solutions. “astar_search with h_pg_levelsum” save the number of new nodes efficiently, but takes longer time as it takes longer time to get the level sum. “depth_first_graph_search” also saves time and number of new nodes, but it didn’t find the optimal result.

Problem 2:

Method	Expansions	Goal Tests	New Nodes	Time	Plan length
breadth_first_search	3346	4612	30534	18.049	9
breadth_first_tree_search	10 mins +				
depth_first_graph_search	107	108	959	0.389	105
depth_limited_search	10 mins +				
uniform_cost_search	4853	4855	44041	16.140	9
recursive_best_first_search with h_1	10 mins+				
greedy_best_first_graph_search with h_1	998	1000	8982	3.071	21
astar_search with h_1	4853	4855	44041	16.330	9
astar_search with h_ignore_preconditions	1450	1452	13303	4.655	9
astar_search with h_pg_levelsum	86	88	841	58.762	9

For Problem 2, the best method is “astar_search with h_ignore_preconditions” as it found the optimal result at with the least time. Although it has more expansions, goal tests and new nodes than “astar_search with h_pg_levelsum”, but it saves the time very efficiently by more than 1/10. Same as problem 1, “depth_first_graph_search” saves expansions, goal test, new nodes and time, but it didn’t find the optimal result.

Problem 3:

Method	Expansions	Goal Tests	New Nodes	Time	Plan length
breadth_first_search	14663	18098	129631	144.748	12
breadth_first_tree_search	10 mins+				
depth_first_graph_search	408	409	3364	2.150	392
depth_limited_search	10 mins+				
uniform_cost_search	18235	18237	159716	71.388	12
recursive_best_first_search with h_1	10 mins+				
greedy_best_first_graph_search with h_1	5614	5616	49429	21.258	22
astar_search with h_1	18235	18237	159716	70.467	12
astar_search with h_ignore_preconditions	5040	5042	44944	19.122	12
astar_search with h_pg_levelsum	318	320	2934	171.532	12

For problem 3: The best method is astar_search with h_ignore_preconditions" as it found the optimal result at with the least time. Although it has more expansions, goal tests and new nodes than "astar_search with h_pg_levelsum", but it saves the time very efficiently by more than 1/10. Same as problem 1 & 2, "depth_first_graph_search" saves expansions, goal test, new nodes and time, but it didn't find the optimal

SEARCHING METHODS PERFORMANCE ANALYSIS

Breadth First Searches & Depth First Searches:

BFS will always find the optimal answer while take longer time and have more expansions, goal tests and number of new nodes compared with DFS. DFS generally get an answer very fast but rarely find the optimal plan as the result of the search method.

BFS's searching time will increase sharply as the problem becomes more complex.

Breadth First Searches & uniform_cost_search:

Uniform_cost_search has slightly more expansions, goal tests and number of new nodes compared with DFS. But the time used will increase slower compared with BFS as problem became more complex.

Astar Searches:

Astar searches with difference heuristics will always find the optimal results, but the performance depends on the heuristics a lot. "h_1" heuristic value is easy to get but not powerful enough to guide the search to get optimal result faster. It works just like the "uniform_cost_search". "h_ignore_preconditions" heuristics is also easy to get and won't expanse every node of the space as the cost can be estimated. In this way, the search space and time is reduced, which means less expansions, goal tests and number of new nodes compared with "h_1" heuristic. "h_pg_levelsum" is the more efficient in guiding the search in the right direction, that is to narrow down the search space. However, to get the heuristic value will take more time.

CONCLUSION

To sum up. If not using heuristics, "uniform_cost_search" will be the best choice as it will get the optimal results with less time. And it is actually working like "astar_search with h_1". If use heuristics, among all the choices above, "astar_search with h_ignore_preconditions" give the optimal results with less time, which should be the best choice. However, if we value less search space more, instead of less time more, "astar_search with h_pg_levelsum" will be a better choice.