Planning Search Heuristic Analysis

By Krishna Kant

In the project, we implemented a planning search to solve the logistics planning problem for air cargo system. Initially we have done non-heuristic search with breadth-first, depth-first and many other techniques. Later we have applied heuristic search to the three problems given.
We have been given three problems and all of them use the same schema. The Action schema is given below:

Action(Load(c, p, a),
    PRECOND: At(c, a) ∧ At(p, a) ∧ Cargo(c) ∧ Plane(p) ∧ Airport(a)
    EFFECT: ¬ At(c, a) ∧ In(c, p))
Action(Unload(c, p, a),
    PRECOND: In(c, p) ∧ At(p, a) ∧ Cargo(c) ∧ Plane(p) ∧ Airport(a)
    EFFECT: At(c, a) ∧ ¬ In(c, p))
Action(Fly(p, from, to),
    PRECOND: At(p, from) ∧ Plane(p) ∧ Airport(from) ∧ Airport(to)
    EFFECT: ¬ At(p, from) ∧ At(p, to))
```

The initial states and goals of the three problems are given below:

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

RESULTS

The analysis was done using both informed and uninformed searches on the three problems. Uninformed searches are based on goal state and non-goal state. It creates successor nodes and checks whether it is the goal state or not. Uninformed searches have no additional information apart from that provided by the problem definition.
Informed search uses information in addition to what is provided in the problem definition. In the results below I have captured the time for execution which would define speed, whether the solution is optimal or not, the plan length, search node expansion and goal test.
The execution time for breadth first tree search, depth limited search and recursive best first search for problem 2 and problem 3 took more than 10 minutes hence were abandoned

Problem 1

Search Type	Expansions	Goal Test	New Nodes	Plan Length	Time(s)	Optimal
breadth_first_search	43	56	180	6	0.025522799	Yes
breadth_first_tree_search	1458	1459	5960	6	0.58673806	Yes
depth_first_graph_search	21	22	84	20	0.008763247	No
depth_limited_search	101	271	414	50	0.055739641	No
uniform_cost_search	55	57	224	6	0.041126119	Yes
recursive_best_first_search h_1	4229	4230	17023	6	1.665087881	Yes
greedy_best_first_graph_search h_1	7	9	28	6	0.003963067	Yes
astar_search h_1	55	57	224	6	0.028279915	Yes
astar_search h_ignore_preconditions	41	43	170	6	0.023562599	Yes
astar_search h_pg_levelsum	11	13	50	6	0.518591928	Yes

Problem 2

Search Type	Expansions	Goal Test	New Nodes	Plan Length	Time(s)	Optimal
breadth_first_search	3343	4609	30509	9	5.006444083	Yes
breadth_first_tree_search	Time more than 10 mins					No
depth_first_graph_search	624	625	5602	619	2.080571663	No

depth_limited_search	Time more than 10 mins					No
uniform_cost_search	4852	4854	44030	9	7.30239868	Yes
recursive_best_first_search h_1	Time more than 10 mins					No
greedy_best_first_graph_search h_1	990	992	8910	17	2.036230379	No
astar_search h_1	4852	4854	44030	9	11.36483558	Yes
astar_search h_ignore_preconditions	1450	1452	13303	9	3.144775885	Yes
astar_search h_pg_levelsum	86	88	841	9	73.0172786	Yes

Problem 3						
Search Type	**Expansions**	**Goal Test**	**New Nodes**	**Plan Length**	**Time(s)**	**Optimal**
breadth_first_search	14663	18098	129631	12	28.93727994	Yes
breadth_first_tree_search	Time more than 10 mins					No
depth_first_graph_search	408	409	3364	392	1.350056331	No
depth_limited_search	Time more than 10 mins					No
uniform_cost_search	18235	18237	159716	12	37.51163197	Yes
recursive_best_first_search h_1	Time more than 10 mins					No
greedy_best_first_graph_search h_1	5614	5616	49429	22	11.0749855	No
astar_search h_1	18235	18237	159716	12	48.24930296	Yes
astar_search h_ignore_preconditions	5040	5042	44944	12	16.74450719	Yes
astar_search h_pg_levelsum	318	320	2934	12	370.0835869	Yes

The optimal path for the three problems is as follows.
For **Problem 1 the optimal plan length is 6**. Below is the plan for the optimal plan length
Load(C1, P1, SFO)
Load(C2, P2, JFK)
Fly(P2, JFK, SFO)
Unload(C2, P2, SFO)
Fly(P1, SFO, JFK)
Unload(C1, P1, JFK)

For **Problem 2 the optimal plan length is 9**. Below is the plan for the optimal plan length
Load(C3, P3, ATL)
Fly(P3, ATL, SFO)
Unload(C3, P3, SFO)
Load(C1, P1, SFO)
Fly(P1, SFO, JFK)

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

For **Problem 3 the optimal plan length is 12**. Below is the plan for the optimal plan length
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(C1, P1, JFK)
Unload(C2, P2, SFO)

Analysis

Amongst the uninformed search, breadth first search and uniform cost search are the only two which finds an optimal plan in all the three problems, whereas the three A* searches with heurisitcs find an optimal plan for all the three problems. In terms of memory and speed greedy_best_first_graph_search is the best for problem 1. It has the minimum expansion nodes and also takes the least amount of time. For Problem 2 & 3 greedy_best_first_graph_search fails to find the optimal path. In terms of speed with a search which was able to find an optimal path A*_ignore_preconditions is the best for problem 2 and 3. From a memory usage standpoint A* search_h_pg_levelsum is the best for problem 2 & 3

Informed Search Vs Uninformed Search

Informed searches definitely score over Uninformed searches. The Greedy best first graph search definitely scores better in problem one but this could be attributed to the smaller problem size. Uninformed searches were able to find the optimal plan for all the three heuristics for all the three problems. Only breadthfirst search and uniform cost search were able to find optimal plan in all the three problems, however breadth first search performed better in memory (Expansions) and time. From the uninformed search heuristics though A*_search_h_pg_levelsum is better than A*_search_h_ignore_precondition in terms of memory however A*_search_h_ignore_precondition far leads in terms of time. For informed and uninformed search comparison lets compare Breadthfirst search and A*_search_h_ignore_precondition. Ignore precconditioin returns a better search as it’s a more relaxed heuristic compared to pg level sum. Depth first search takes a dip at depth to find the optimal path. if the optimal path is not at the beginning of the tree, the depth first tree is bound to take longer time.

Problem	Search Type	Expansions	Goal Test	New Nodes	Path Length	Time(s)	Optimal
Problem 1	**breadth_first_search**	43	56	180	6	0.025522799	Yes
	astar_search h_ignore_preconditions	**41**	43	170	6	**0.023562599**	Yes
Problem 2	**breadth_first_search**	3343	4609	30509	9	5.006444083	Yes
	astar_search h_ignore_preconditions	**1450**	1452	13303	9	**3.144775885**	Yes
Problem 3	**breadth_first_search**	14663	18098	129631	12	28.93727994	Yes
	astar_search h_ignore_preconditions	**5040**	5042	44944	12	**16.74450719**	Yes

The analysis from the table above clearly shows that A*search with the heuristic which ignores preconditions scores better than breadth first search both in terms of memory (expansions are less) and speed (Time is less). Also the performance of A* becomes better as the problem grows in size.