Planning Search - Heuristic Analysis

Introduction and Problem definition

For this project, we were asked to implement a planning search agent. We were directed to "define a group of problems in classical PDDL for the air cargo domain discussed in the lectures" and "setup the problems for search, experiment with automatically generated heuristics" and solve the problem. In this written analysis, we will review the approach and results of our experiments.

The project included 3 problems to solve, with the following action schema, summarized in table 1.

Table 1

Action Name	Action Schema
Load	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))
Unload	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))
Fly	Action(Fly(p, from, to), PRECOND: At(p, from) ∧ Plane(p) ∧ Airport(from) ∧ Airport(to) EFFECT: ¬ At(p, from) ∧ At(p, to))

The three problems had the following initial states and goal, summarized in table 2.

Table 2

Problem	Initial State	Goal
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))
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))
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))

Uninformed Search Algorithms

The first part of the analysis consists of running at least 3 search agents based upon uniformed search algorithms. Uninformed search algorithms are called so, because they do not differentiate between different non-goal states. In our analysis, we used 3 uniformed search algorithms for our agent:

1. Breadth-first search
2. Depth-first tree search
3. Uniform-cost search

Informed Search Algorithms

The second part of the analysis consists of running at least 2 search agents based upon A* search algorithm with different heuristic functions. In our analysis, we used 2 different heuristics along with the A* search algorithm, as follows:

1. A* with ignore preconditions heuristics
2. A* with level-sum heuristic

Results:

Table 3 contains a summary of the results for all problems and search algorithms. It includes the plan length, whether the algorithm returned an optimal plan length or not, the time elapsed and the number of Nodes expanded.

Table 3

Algorithm	Plan Length	Optimal	Time Elapsed(s)	Nodes Expanded
Problem 1				
Breadth-first	6	Yes	0.05	43
Depth-first	12	No	0.01	12
Uniform-cost	6	Yes	0.05	55
A* with ignore preconditions	6	Yes	0.08	41
A* with level-sum	6	Yes	1.23	11
Problem 2				
Breadth-first	9	Yes	19.95	3343
Depth-first	575	No	4.98	582
Uniform-cost	9	Yes	17.85	4830
A* with ignore preconditions	9	Yes	9.22	1425
A* with level-sum	9	Yes	224.80	79
Problem 3				
Breadth-first	12	Yes	156.19	14663
Depth-first	596	No	4.91	627
Uniform-cost	12	Yes	78.47	18233
A* with ignore preconditions	12	Yes	37.10	4857
A* with level-sum	12	Yes	1224.19	291

Sample Plans

Table 4 contains sample plans for the three problems. The order of the plan may change but the plan length remains the same for any other optimal plan.

Table 4

Problem	Plan Steps	Plan Length
1	Load(C2, P2, JFK) Load(C1, P1, SFO) Fly(P2, JFK, SFO) Unload(C2, P2, SFO) Fly(P1, SFO, JFK) Unload(C1, P1, JFK)	6
2	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(C2, P2, SFO) Unload(C3, P3, SFO) Unload(C1, P1, JFK)	9
3	Load(C2, P2, JFK) Load(C1, P1, SFO) Fly(P2, JFK, ORD) Load(C4, P2, ORD) Fly(P1, SFO, ATL) Load(C3, P1, ATL) Fly(P1, ATL, JFK) Unload(C1, P1, JFK) Unload(C3, P1, JFK) Fly(P2, ORD, SFO) Unload(C2, P2, SFO) Unload(C4, P2, SFO)	12

Optimality

We should expect breadth-first search and uniform-cost search to give us correct path lengths as these algorithms are optimal, i.e. they will always give the shortest path to the goal state.

Artificial Intelligence a Modern Approach (AIMA) discusses the reasons as to why these two algorithms should give us optimal results.

Firstly, breadth-first search "discards any new path to a state already in the frontier or explored state; it is easy to see that any such path must be at the least as deep as the one already found. Thus, breadth-first search always has the shallowest path to every node on the frontier" [Ch 3, Pg. 81, AIMA 3rd edition, Norvig and Russel]. This means that breadth-first search is always optimal as the path to the goal state would be the shortest path available in the search tree.

Also, "Uniform cost search is optimal in general. First, we can observe that whenever uniform cost search selects a node *n* for expansion, the optimal path to that node has been found…Then because step cost or non-negative, paths never get shorter as nodes are added" [Ch 3, Pg. 85, AIMA 3rd edition, Norvig and Russel], uniform-cost search should always give us the optimal plan length for our problem.

Depth-first search, however is not optimal as "Depth-first search expands the deepest unexpanded node first. It is neither complete nor optimal" [Ch 3, Pg. 109, AIMA 3rd edition, Norvig and Russel]. So, depth-first search may reach the goal state via a path that is much longer than the optimal plan.

A* search will be optimal depending upon the heuristic function. If the heuristic function is admissible, i.e. the estimated cost of the heuristic function is always lower than the actual cost till goal state, then the A* search algorithm will give an optimal plan as the solution. [Ch 3, Pg. 382, AIMA 3rd edition, Norvig and Russel]

For Ignore precondition, the function is inadmissible (i.e. it overestimates the cost to the goal state) when "one action may achieve multiple goals" [Ch 3, Pg. 382, AIMA 3rd edition, Norvig and Russel]. For our aircargo problems none of the actions, as defined in the action schema, achieve two goal states at once; hence for our problems the heuristic is admissible and A* search with ignore preconditions is optimal.

The level-sum heuristic assumes subgoal independence and will overestimate the cost when "subplans contain redundant actions, for instance – two actions that could be replaced by a single action in the merged plan" [Ch 3, Pg. 384, AIMA 3rd edition, Norvig and Russel]. For the air cargo problem, the optimality of this search agent would be dependent upon the initial state of the problem. Inadmissible means it does not guarantee an optimal solution, however, it is possible that the agent returns an optimal solution by chance due to the initial and goal states.

From the results, we can see that all tested algorithms, except depth first search, give us an optimal solution. While, breadth-first, uniform-cost and A* with ignore precondition searches are optimal for this problem, the level-sum search does not guarantee an optimal solution but returned on anyways.

Time and Nodes

The average time elapsed and node expansion values over all problems and for all algorithms are shown in table 5.

Table 5

Algorithm	Time elapsed (seconds)	Nodes Expanded
Breadth-first	58.73	6016.33
Depth-first	3.30	407.00
Uniform-cost	32.12	7706.00
A* with ignore preconditions	15.47	2107.67
A* with level-sum	483.41	127.00

Figure 1, shows the average time elapsed and nodes expanded values on a bar chart. We can see depth-first search was the fastest and expanded the least number of nodes but this algorithm did not return an optimal solution so we will not consider it as a recommendation. The uninformed searches expanded a lot of nodes and this makes sense, since they cannot differentiate between a good node and a bad node for further expansion. For the number of nodes expanded however, their speed compared to A* search

with level-sum heuristic was much better. This is since they do not have to calculate a heuristic function for every node expansion.

The A* with ignore precondition search gave us the best results overall. It had the lowest node expansions and time elapsed amongst the optimal solution algorithms. This is because unlike the uninformed searches it does have a guiding function that allows it to minimize the number of nodes it needs to expand. And, unlike the level-sum heuristic the ignore preconditions heuristic is very fast to calculate and hence the overall speed of the algorithm is much better than A* with level-sum.

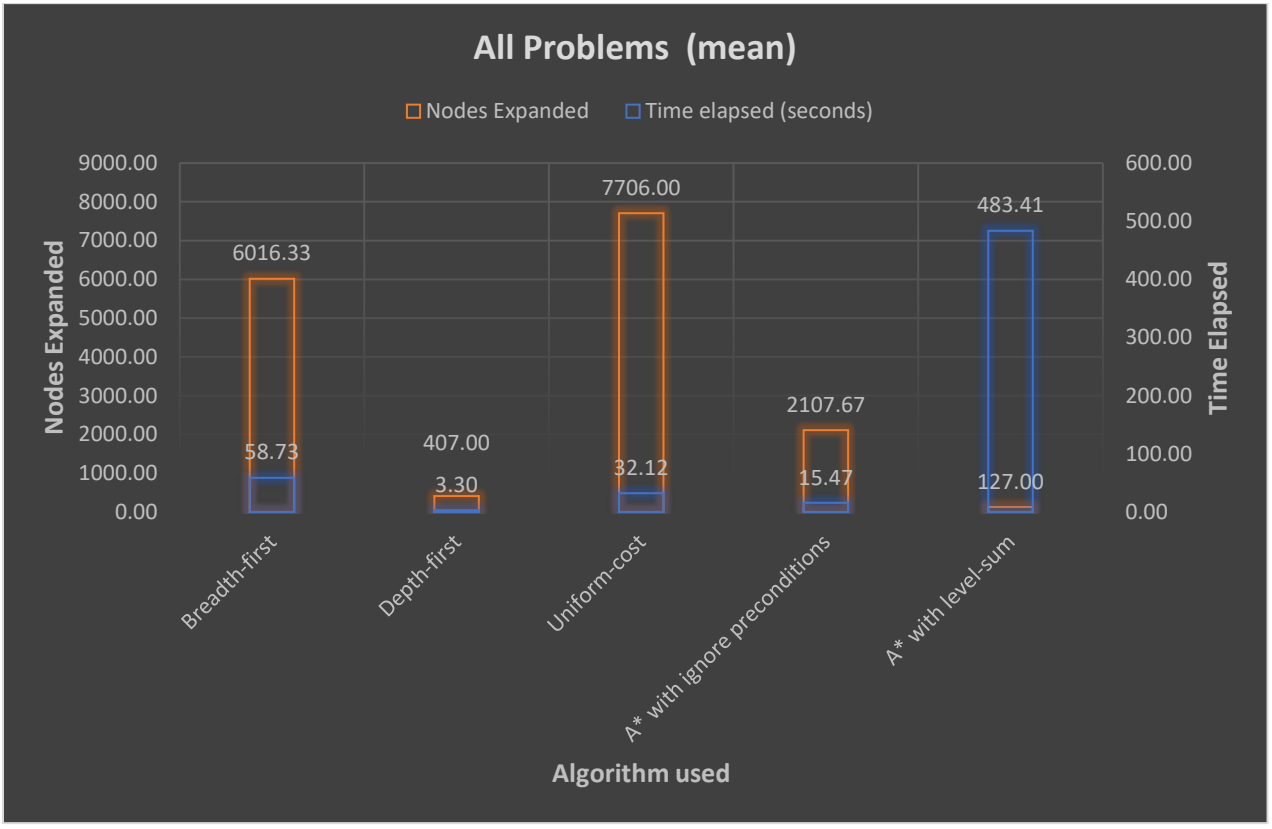

Figure 1

Conclusion

Looking at the results above it would be a good idea to recommend the **A* search with ignore preconditions heuristic** to solve our air cargo problems.

Amongst the algorithms that returned an optimal plan as the solution, the A* search with ignore preconditions heuristics is the fastest and expands the least number of nodes. We can see the advantages of informed search algorithms, they expand much fewer nodes than uninformed searches and their speeds can be lowered by using the correct heuristic.