Planning Search Heuristic Analysis

by Jordan Carson

Introduction

For this project, we implemented a planning search agent to solve deterministic logistics planning problems for an Air Cargo transport system.

Planning Problems

We were given three Air Cargo planning problems that follow the same below action schema:

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))

From this, we had to solve for three air_cargo_problems() that have the following initial states and goals. This exercise was to solve the air_cargo_problems() and build A* search heuristic of informed search and compare these to the AIMA uninformed search heuristics.

Air Cargo problem 1:
Initial State and Goal:
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)

Air Cargo problem 2:
Initial State and Goal:
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))

Air Cargo Problem 3:
Initial State and Goal:
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))

The goals above can be obtained through a different set of plans and lengths of plans. The optimal plan lengths for problems 1, 2, and 3 are 6, 9, and 12 actions. Below are the optimal sequence of actions:

Problem 1:
Load(C1, P1, SFO)
Load(C2, P2, JFK)
Fly(P2, JFK, SFO)
Fly(P1, SFO, JFK)
Unload(C2, P2, SFO)
Unload(C1, P1, JFK)

Problem 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)

Problem 3:
Load(C1, P1, SFO)
Load(C2, P2, JFK)
Fly(P1, SFO, ATL)
Load(C3, P1, ATL)
Fly(P2, JFK, ORD)
Load(C4, P2, ORD)
Fly(P2, ORD, SFO)
Fly(P1, ATL, JFK)
Unload(C4, P2, SFO)
Unload(C3, P1, JFK)
Unload(C2, P2, SFO)
Unload(C1, P1, JFK)

Comparison of Uninformed Planning Searches

The results for the uninformed deterministic searches can be found in the pdf titled "Uninformed search results.pdf" or below:

- Breath First Search (BFS): Also known as shortest first search, will always find the shortest way in terms of node searching to the goal, however it will take more time than other searches.
- Depth First Search (DFS): This method is faster than BFS, but the path to the goal takes longer as it generates a longer search path. This is not an optimal case.
- Uniform Cost Search (UCS): Also known as cheapest first search picks the path with the lowest total cost. Thus this would be the optimal method compared to BFS or DFS.

	Air Cargo Problem	Expansion Nodes	Goal Tests	New Nodes	Length	Time Elapses in Seconds	Optimality
Breath First Search	1	43	56	180	6	0.0287	Optimal
Breath First Search	2	2899	3845	25527	9	10.1856	Optimal
Breath First Search	3	14663	18098	129631	12	89.5827	Optimal
Breath First Tree Search	1	1458	1459	5960	6	0.7823	Optimal
Depth First Graph Search	1	21	22	84	20	0.0108	Bad
Depth First Graph Search	2	1524	1525	12704	557	4.7730	Bad
Depth First Graph Search	3	408	409	3364	392	1.5544	Bad
Depth Limited Search	1	101	271	414	50	0.0692	Bad
Greedy Best First Graph Search with h1	1	7	9	28	6	0.0066	Optimal
Greedy Best First Graph Search with h1	2	726	728	6334	30	1.8481	Bad
Greedy Best First Graph Search with h1	3	5579	5581	49159	22	15.9469	Good
Recursive Best First Search with h1	1	4229	4230	17023	6	2.5431	Optimal
Uniform Cost Search	1	55	57	224	6	0.0305	Optimal
Uniform Cost Search	2	4000	4002	35439	9	10.7097	Optimal
Uniform Cost Search	3	18223	18225	159618	12	47.2723	Optimal

Informed Search Results

	Air Cargo Problem	Expansions	Goal Tests	New Nodes	Plan Length	Time Elapses in Seconds	Optimal?
A* with H1	1	55	57	224	6	0.0346	Optimal
A* with H1	2	4000	4002	35439	9	9.4668	Optimal
A* with H1	3	18223	18225	159618	12	45.5419	Optimal
A* with Ignore Preconditions	1	41	42	170	6	0.0323	Best
A* with Ignore Preconditions	2	1317	1319	11820	9	3.6357	Best
A* with Ignore Preconditions	3	5040	5042	44944	12	15.2359	Best
A* with level-sum	1	56	58	228	6	1.4970	Good
A* with level-sum	2	4581	4583	40439	9	1595.2919	Good
A* with level-sum	3	20049	20051	174481	12	11439.4219	Good

Comparison of Informed Planning Searches
The A* search finds the shortest length path while it is expanding the minimum. It depends on the heuristic, which keeps the algorithm focused to reach the goal. Looking at the above results, there are far less expansions required for the A* with ignore preconditions heuristic, however this is far faster than H1 or level-sum. The level-sum then requires less goal tests but if you compare the amount of time taken the ignore preconditions heuristic is the most optimal.

According to the Norvil & Russell AIMA textbook, using A* is a good heuristic to find optimal solutions to planning problems, and an admissible heuristic can be discovered by defining a relaxed problem that is easier to solve. In our planning search, a "The ignore preconditions heuristic drops all preconditions from actions. Every action becomes applicable in every state, and any single goal fluent can be achieved in one step." This almost implies that this heuristic uses a relaxed problem in the number of steps required to solve the problem, because some actions may achieve multiple goals and some actions may undo the effects of others.

Looking at the above results, there are far less expansions required for the A* with ignore preconditions heuristic; however this is far faster than H1 or level-sum. The level-sum then requires less goal tests but if you compare the amount of time taken the ignore preconditions heuristic is the most optimal.

Informed vs Uninformed Search
The best search strategies, as the strategies that generate optimal plans, are BFS, UCS, and A* search with all heuristics.

BFS and UCS are simple strategies that expand the root node first, then their successors are expanded, then their successors and so on. The order how we expand these successors is what makes the difference between BFS and UCS. BFS will expand all nodes at a given depth in the search tree before expanding any node at the next level (this is implemented with a FIFO queue for the frontier). UCS has a different expansion strategy: Instead of going level by level, it expands the node with the lowest path cost. This lowest path cost in geographical space, means cities that are closer to the root nodes are expanded first. This is implemented with a priority queue.

The results of the A* search (informed) strategies with custom heuristics over-perform uninformed search techniques when searching for an optimal plan. According to Norvig et al., “The level-sum heuristic returns the sum of the level costs of the goals; this can be inadmissible but works well in practice for problems that are largely decomposable.” Looking at the results, our level-sum output optimality is good however the H1 heuristic provides better results. The Level-sum heuristic should provide us with the least number of expansions, while the ignore-preconditions as the next best heuristic for this metric. This means that there are true benefits of building custom-made heuristics for a particular problem, and these benefits can be seen in terms of speed, memory usage, and the plan’s length.

Cited works:

Artificial Intelligence A Modern Approach (3rd Edition), Peter Norvig & Stuart Russell