Heuristic Analysis

For Udacity-AIND Planning Project

In this project, we solved deterministic logistics planning problems for an Air Cargo transport system using a planning search agent. In part one of the project we implement uninformed non heuristic search methods (like breadth-first search, depth-first search etc.) based on the problems defined in classical PDDL (Planning Domain Definition Language). In part 2 we implemented domain independent heuristics using a planning graph and A* search. And In this report we will compare the results of part 1 & part 2 and will try to deduce which strategy is better.

Planning Problem Overview

We were given three problems in the Air-Cargo Domain. They had the same action schema defined, but different initial states and goals.

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))

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))

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))

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))

We reached our goals using different plans of different lengths and execution time but the optimal plan lengths for problems 1, 2, 3 were 6, 9, 12 respectively. And these are the plans with these optimal lengths.

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

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(C3, P3, SFO)
Unload(C1, P1, JFK)
Unload(C2, P2, SFO)

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

Part 1 Analysis

In this section we will compare the performance of the uninformed search strategies where no extra information is given about the states. We will compare seven different algorithms on the basis of time elapsed, path length(optimality) and node expansions(memory). For problem 2 & 3 as the time elapsed is much larger, so we will compare only four algorithms.

Problem 1:

Algorithm	Time Elapsed	Path length	Node expansions	Optimality
Breadth First Search	0.061	6	43	Yes
Breadth First Tree Search	1.614	6	1458	Yes
Depth First Graph Search	0.013	12	12	No
Depth Limited Search	0.162	50	101	No
Uniform Cost Search	0.062	6	55	Yes
Recursive Best First Search	4.708	6	4229	Yes
Greedy Best First Graph Search	0.008	6	7	Yes

Problem 2:

Algorithm	Time Elapsed	Path length	Node expansions	Optimality
Breadth First Search	22.164	9	3343	Yes
Depth First Graph Search	4.766	575	582	No
Uniform Cost Search	18.622	9	4852	Yes
Greedy Best First Graph Search	3.790	21	990	No

Problem 3:

Algorithm	Time Elapsed	Path length	Node expansions	Optimality
Breadth First Search	160.859	12	14663	Yes
Depth First Graph Search	5.050	596	627	No
Uniform Cost Search	81.809	12	18235	Yes
Greedy Best First Graph Search	25.344	22	5614	No

From the results comparison here in the tables above we can have a consensus that **Breadth First Search** & **Uniform Cost Search** are the two algorithms we can use for our problem. Among these two, Breadth First Search also takes less number of Node Expansions so less memory usage and slightly higher execution time so We can say that **Breadth First Search** is the best algorithm for this problem. The only disadvantage of these two algorithms are the number of node expansions i.e. Memory Usage. So if the branching factor becomes too high then we know that Breadth First Search has space and time complexities depended on the branching factor and hence in that case it is not optimal. **This can be justified from AIMA Textbook (3rd Edition) page 91 Section 3.4.7 where the Uninformed Search algorithms are compared**. However If we have a memory and time constraint then **Depth First Graph Search** should be the one used as it is the fastest and uses least memory. But the optimal plan length of this algorithm is too high which is again an issue. So **Greedy Best First Graph Search** can act as a solution in case of high branching

factor even though its execution time and node expansions are somewhat higher but its optimal path length is way more less than the Depth First Graph Search.

Part 2 Analysis

Here we will see the comparison of the informed heuristic based search algorithms with A* search where we have extra information about the states which helps to search efficiently. We will compare three different heuristics with A* search. We will not compare the level-sum heuristic for problem 3 since the execution time is much larger.

Problem 1:

Algorithm	Time Elapsed	Path length	Node expansions	Optimality
A* Search with h1	0.068	6	55	Yes
A* Search with ignore preconditions	0.064	6	41	Yes
A* Search with level-sum	1.807	6	11	Yes

Problem 2:

Algorithm	Time Elapsed	Path length	Node expansions	Optimality
A* Search with h1	18.538	9	4852	Yes
A* Search with ignore preconditions	6.920	9	1450	Yes
A* Search with level-sum	283.143	9	86	Yes

Problem 3:

Algorithm	Time Elapsed	Path length	Node expansions	Optimality
A* Search with h1	82.684	12	18235	Yes
A* Search with ignore preconditions	26.796	12	5040	Yes

From the results comparison here in the tables above we can have a consensus that both the **A* search with h1** and **A* search with ignore preconditions** algorithms gets us the best results. In between them **A* search with ignore preconditions** gets to be the winner as it shows better trade offs between speed and memory usage. **This can also be justified from the AIMA Textbook page 97, 98 Section 3.5.2 Sub Heading Optimality of A* where A* is declared optimally efficient for any given consistent heuristic**. **A* search with level-sum** uses less memory than the other two but takes a lot of execution time.

Part 1 vs Part 2 Analysis

Here we have the three best algorithms from the uninformed search and informed search. Breadth First search and Uniform Cost Search from the uninformed search and A* search with ignore preconditions heuristic from the informed search.

Problem 1:

Algorithm	Time Elapsed	Path length	Node expansions	Optimality
Breadth First Search	0.061	6	43	Yes
Uniform Cost Search	0.062	6	55	Yes
A* Search with ignore preconditions	0.064	6	41	Yes

Problem 2:

Algorithm	Time Elapsed	Path length	Node expansions	Optimality
Breadth First Search	22.164	9	3343	Yes
Uniform Cost Search	18.622	9	4852	Yes
A* Search with ignore preconditions	6.920	9	1450	Yes

Problem 3:

Algorithm	Time Elapsed	Path length	Node expansions	Optimality
Breadth First Search	160.859	12	14663	Yes
Uniform Cost Search	81.809	12	18235	Yes
A* Search with ignore preconditions	26.796	12	5040	Yes

Now from these tables above we can conclude that **A* search with Ignore preconditions heuristic** in Part 2 (Informed search) is clearly the best algorithm for this planning problem. It has the best trade off between execution time, memory usage and optimal path length. This also states that informed search has more benefits than uninformed search.