Planning Search Heuristic Analysis

Safak Ozkan
shafaksan@gmail.com

1. Problem Definiton

We are given 3 classical PDDL problems in the Air Cargo domain. They have the same action schema defined as below.

Air Cargo 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))
```

The initial states and goals of the problems are defined in PDDL as follows

Air Cargo 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))
```

Air Cargo 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))
```

Air Cargo 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))
```

2. Search Methods and Heuristic Functions

Three uninformed (no heuristic) search methods have been used are **Depth First Search**, **Breadth First Search** and **Uniform Cost Search**. They all find a solution although of varying plan lengths and with varying efficiencies [1,2].

1. **Depth First Search** requires only a small amount of memory, however it explores each branch until its fully exhausted. Hence, when the algorithm is at the wrong branch, it wastes an undue amount of time before it hops on to a new branch. The trade off for little memory requirement is that DFS is not optimal. It often finds a solution that's much longer than the possible minimum length plan.
2. **Breadth First Search**, however advances down the search tree one level at a time spanning all the branches. Hence, it requires large memory usage. As opposed to DFS, BFS is optimal as long as the actions are unweighted (i.e. they all have equal cost).
3. **Uniform Cost Search** is the best algorithm for a search problem, which does not involve the use of heuristics. It is similar to BFS but it takes into account the cost of each action and picks the one with the minim cost first. Uniform Cost Search searches in branches which are more or less the same in cost.

For informed search method **A* Search** is used with 3 different heuristic functions. In increasing level of complexity these are listed as below:

1. **A* h_1** is a constant function non-heuristic heuristic. It's used to gauge the efficiency of A* in the absence of the computational burden a heuristic function would imposes. In essence, it yields the same algorithm as UCS.
2. **A* h_ignore_preconditions** ignores the preconditions. This heuristic estimates the minimum number of actions that must be carried out from the current state in order to satisfy all of the goal conditions.
3. **A* h_pg_level_sum** uses a planning graph representation of the problem state space and estimates the sum of all actions to satisfy the goal condition

3. Results and Evaluation

The search result metrics for the Air Cargo problems are given in Tables 1-3. For each problem, the search algorithm that yielded the shortest Plan Length and the shortest amount of runtime has been regarded as the most optimal search algorithm and has been duly highlighted.

For Problem 1, BFS has found the optimal plan and also performed better in terms of run time. This is because Problem 1 is simple enough to not require a computationally expensive heuristic. However, Problem 2 and 3 benefits from heuristic functions. In particular **h_ignore_preconditions** for Problem 2 and **h_pg_level_sum** for Problem 3 helps find the optimal plan in the shortest amount of time. Having said that, for Problem 1, A* search algorithms with heuristics, even though are able to find the optimal plans, do so in an exorbitant amount of time since they burden the algorithm with an unnecessary computation load of the heuristic functions.

DFS lacks optimality and hence finds a sub-optimal plan in each of the problems.

BFS search algorithm provides a good first choice for simple planning problems. However, as the problems get more complex, incorporating A* search with a heuristic function pays off in finding the optimal path in a shorter run time than BFS.

Table 1. Air Cargo Problem 1 search results

	Node Expansions	Plan Length	Time (sec)	Optimality
Breadth First Search	43	6	0.04	Yes
Depth First Graph Search	21	20	0.017	No
Uniform Cost Search	55	6	0.043	Yes
A^* h_1	55	6	0.048	Yes
A^* h_ignore_preconditions	41	6	0.036	Yes
A^* h_pg_levelsum	11	6	0.83	Yes

Table 2. Air Cargo Problem 2 search results

	Node Expansions	Plan Length	Time (sec)	Optimality
Breadth First Search	3353	9	16.3	Yes
Depth First Graph Search	1160	1109	9.2	No
Uniform Cost Search	4853	4855	14.5	Yes
A^* h_1	4853	4855	14.6	Yes
A^* h_ignore_preconditions	1450	9	4.7	Yes
A^* h_pg_levelsum	86	9	61.2	Yes

Table 3. Air Cargo Problem 3 search results

	Node Expansions	Plan Length	Time (sec)	Optimality
Breadth First Search	14120	12	119.1	Yes
Depth First Graph Search	292	288	1.4	No
Uniform Cost Search	18223	12	64.7	Yes
A^* h_1	18223	12	64.8	Yes
A^* h_ignore_preconditions	5040	12	19.3	Yes
A^* h_pg_levelsum	316	12	298.8	Yes

Table 4. Air Cargo Problem 1 – Optimum Sequence of actions, Breadth First Search

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

Table 5. Air Cargo Problem 2 – Optimum Sequence of actions, A* h_ignore_preconditions

`Load(C3, P3, ATL)`
`Fly(P3, ATL, SFO)`
`Unload(C3, P3, SFO)`
`Load(C2, P2, JFK)`
`Fly(P2, JFK, SFO)`
`Unload(C2, P2, SFO)`
`Load(C1, P1, SFO)`
`Fly(P1, SFO, JFK)`
`Unload(C1, P1, JFK)`

Table 6. Air Cargo Problem 3 – Optimum Sequence of actions, A* h_pg_levelsum

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

REFERENCES

1. Russell, S. and Norvig, P. (1995). *Chapter 11: Planning*, Artificial Intelligence: A Modern Approach,

2. Russell, S. and Norvig, P. (1995). *Chapter 3: Search Strategies*, Artificial Intelligence: A Modern Approach,