HEURISTIC ANALYSIS

For Deterministic Logistics Planning Problems using a Planning Search Agent

Vatsal Srivastava

The project aims to understand and solve deterministic logistics planning problems for an Air Cargo transport system using a planning search agent. With progression search algorithms optimal plans for each problem will be computed. Since there is no simple distance heuristic to aid the agent, domain-independent heuristics will be utilized. The problem is to be first set up for search, followed by experimental analysis of automatically generated heuristics, including planning graph heuristic.

- 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))
```

- 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))
```

<u>Uninformed Non-Heuristic Search:</u>

The uninformed non-heuristic planning was experimented with for *Breadth First Search* (BFS), *Depth First Search* (DFS) and *Uniform Cost Search* (UCS). The results for the same can be summarized below.

Problem	Search Type	Plan Length	Time Elapsed (sec)	Number of Node Expansions	Number of Goal Tests	New Nodes
P1	BFS	6	0.0337	43	56	180
P1	DFS	12	0.00898	12	13	48
P1	UCS	6	0.0423	55	57	224
P2	BFS	9	16.669	3346	4612	30534
P2	DFS	105	0.3635	107	108	959
P2	UCS	9	14.47	4853	4855	44041
P3	BFS	12	124.805	14663	18098	129631
P3	DFS	3955	81.163	4189	4190	35475
P3	UCS	12	60.412	17882	17884	156769

From the above Table we can observe that DFS takes lesser time compared to the two other search techniques. However, it seems to always have a much higher Plan Length. This behavior can be explained easily by the fact that DFS always explores a possible path till the end before trying an alternate path. This approach causes the algorithm to find a solution faster but leads to a sub-optimal solution.

Both BFS and UCS take longer times to find a solution as they explore all the immediate alternatives before exploring the next level of the data. This approach, although slow, produces a much more optimal solution.

One other thing to observe is that since the memory required by the search is directly dependent on the number of nodes expanded, the DFS is the most efficient in terms of space constraint.

Heuristic Search:

The heuristic planning was experimented with following heuristics:

- h_1 – Technically same as UCS as the heuristic always returns 1
- h_ignore_preconditions – simply counts the number of outstanding goals
- h_pg_levelsum – uses a planning graph to estimate the number of actions required to meet the goals

The results can be summarized in the table below.

Problem	A* Heuristic for Search	Plan Length	Time Elapsed (sec)	Number of Node Expansions	Number of Goal Tests	New Nodes
P1	h_1	6	0.0442	55	57	224
P1	h_ignore_preconditions	6	0.0443	41	43	170
P1	h_pg_levelsum	6	12.454	11	13	50
P2	h_1	9	14.3289	4853	4855	44041
P2	h_ignore_preconditions	9	5.168	1450	1452	13303
P2	h_pg_levelsum	9	6528.71	86	88	841
P3	h_1	12	57.84	17882	17884	156769
P3	h_ignore_preconditions	12	18.7	5034	5036	44886
P3	h_pg_levelsum	12	>10min	-	-	-

As can be observed from the table above, all three of the heuristics with A^* produced the optimal solution for each problem, albeit with a big tradeoff between time and space requirements.

The **h_ignore_precondition** heuristic is always the fastest by a significant margin while the **h_pg_levelsum** algorithm failed to lead to a solution in under 10 minutes for Problem 3 and was terminated.

In terms of space constraints, the **h_pg_levelsum** outperforms all other algorithms by a huge margin. At its best, it required less than 2% of the space of h_1 (for Problem 2). This can be because Planning Graphs reduce the exponential complexity of the search to polynomial size (AIMA Chapter 10.3).

Thus, it seems that simpler heuristics seem to be the best when it comes to time efficiency, while complex heuristics seem to be the best when space complexity is a consideration.

Conclusion:

Considering all the above factors into account, **h_ignore_preconditions** seem to be the preferred algorithm. This is because, it provides a nice balance between the time taken, memory required and the guarantee of a solution. The fact that the algorithm guarantees completeness is also a huge advantage.

Appendix

1. Optimal path identified for Problem 1 with A^* and h_ignore_preconditions

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

2. Optimal path identified for Problem 2 with A^* and 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)
```

3. Optimal path identified for Problem 3 with A^* and h_ignore_preconditions

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