Analysis of Different Search Algorithms on the Air Cargo Problem

The Problem

The Air Cargo Schema in PDDL is given as

```
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))
```

For each problem, we compare the following search technique:

1. Breadth first graph search: see Section 3.3 of [1] for detailed descriptions of this search
2. Breadth first tree search: see Section 3.3 of [1] for detailed descriptions of this search
3. Depth first graph search: see Section 3.3 of [1] for detailed descriptions of this search
4. A* search with h1: A* search with naive heuristic function. For each state, h=1.
5. A* search with h_ignore_preconditions: The heuristic function counts the differences between the number of literals in the goal state and the current state. See Section 10.3.1 of [1] for more details.
6. A* search withh_pg_levelsum: this heuristic function adds up the level in in the planning graph that a goal literal is first seen. See Section 10.3.1 of [1] for more details.

Problem 1

PDDL representation:

```
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))
```

Result

Search	Expansions	Goal Tests	New nodes	Plan Lengh	Time elapsed
Breadth first graph search	43	56	180	6	0.036
Breadth first tree search	1458	1459	5960	6	1.310
Depth first graph search	12	13	48	12	0.010627
A* with h_1	55	57	224	6	0.045
A* with h_ignore preconditions	41	43	170	6	0.05
A* with h_pg_levelsum	41	43	170	6	3.2977

Optimal solution:

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

Analysis:

This problem is relatively simple. It only has two cargos, two planes, and two airports. Straightforward breadth first graph search (BFS) and depth first graph search (DFS) work best in this case. A* search with heuristic function does not help reducing the search space because search tree is shallow.

Problem 2

PDDL representation

```
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))
```

Search	Expansions	Goal Tests	New nodes	Plan Length	Time elapsed
Breadth first graph search	3346	4612	30534	9	16.257
Breadth first tree search	Inf	Inf	Inf	Inf	Inf

Search	Expansions	Goal Tests	New nodes	Plan Length	Time elapsed
Depth first graph search	1124	1125	10017	1085	9.088
A* with h_1	4605	4607	41839	9	85.72
A* with h_ignore preconditions	1399	1401	12816	9	18.08
A* with h_pg_levelsum	1138	1140	10303	9	2259.92

Optimal solution:

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

Analysis

This problem has three cargos, three planes, and three airports. The results show that `A* with h_ignore_preconditions` and `A* with h_pg_levelsum` effectively reduced the number of searched node. However, `A* with h_pg_levelsum` takes more time to calculate because computing the level sum is computationally expensive in our implementation. It is worth mentioning that although DFS takes less time to find a solution, it is not optimal.

Problem 3

PDDL representation

```
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))
```

Results

Search	Expansions	Goal Tests	New nodes	Plan Length	Time elapsed
Breadth first graph search	14120	17673	124926	12	133.493
Breadth first tree search	Inf	Inf	Inf	Inf	Inf
Depth first graph search	677	678	5608	660	18.099

Search	Expansions	Goal Tests	New nodes	Plan Length	Time elapsed
A* with h_1	16961	16963	149117	12	818.56
A* with h_ignore preconditions	4723	4725	41835	12	112.39
A* with h_pg_levelsum	2622	2624	23148	12	7023.16

Analysis

This problem has four cargos, two planes, and four airports. The search space is then much more larger than the previous two problems. DFS and BFS then becomes less effective. BFS found an optimal solution but explored too many nodes, whereas DFS expanded less nodes but returned a non-optimal plan. `A* search with h_pv_levelsum` expanded the least number of nodes but it takes large amount of time with our implementation. `A* with h_ignore_preconditions` seems to be the most efficient one in this case.

Optimal Solution

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

Reference

[1] S. Russell and P. Norvig, “Artificial Intelligence: A Modern Approach” 3rd ed., Pearson, 2010