Planning Search Algorithm Analysis

For this analysis we will use 3 problems in the air cargo domain defined as follow:
- 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))
```
Planning Problems

Three search strategies are evaluated for each problems and the result are provided.
Note that breath first tree search was not able to complete Problem 2 and Problem 3
as it was expending too many nodes.

Problem1:

Algorithm	**Node expansions**	**Path Length**	**Time Elapsed**	**Optimal**
Breath First Search	43	6	0.031	yes
Breath First Tree Search	1458	6	1.031	yes
Depth First Search	12	12	0.008	no

An optimal sequence for the problem is:

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

Problem2:

Algorithm	**Node expansions**	**Path Length**	**Time Elapsed**	**Optimal**
Breath First Search	3343	9	15.32	Yes
Breath First Tree Search	-	-	-	-
Depth First Search	1669	1444	15.27	No

An optimal sequence for the problem is:

```
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)
```

Problem3:

Algorithm	Node expansions	Path Length	Time Elapsed	Optimal
Breath First Search	14663	12	111.44	Yes
Breath First Tree Search	-	-	-	-
Depth First Search	592	571	4.60	No

An optimal sequence for the problem is:

```
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)
```

Analysis

Breath First search is capable of finding an optimal solution. It takes longer on average than Depth First Search and expands more nodes.
Breath First Tree Search can find an optimal solution but only for very small problems. It was not able to find a solution within 10 minutes for the more complex ones.
Depth First Search finds a solution very quickly but it is not optimal.
Those observations are in line with the description of uninformed search algorithm as described in [1]. Breath first search expand the nodes level by level until if finds the goal. The search is guaranteed to be optimal but requires maintaining a lot of expanded node in memory.

Depth first search on the other hand go as deep as possible. It is not guaranteed to be optimal but do not require as much memory.

Domain-independent heuristics

In this part of the analysis we are investigating the performance of the different
Heuristics applied to the A* algorithm.

Problem1

Heuristic	**Node expansions**	**Path Length**	**Time Elapsed**	**Optimal**
1	55	6	0.0447	Yes
Ignore precondition	41	6	0.04	Yes
levelsum	11	6	1.904	Yes

Problem2

Heuristic	**Node expansions**	**Path Length**	**Time Elapsed**	**Optimal**
1	4852	9	13.4058	Yes
Ignore precondition	1450	9	4.616	Yes
levelsum	86	9	333.63	Yes

Problem3

Heuristic	**Node expansions**	**Path Length**	**Time Elapsed**	**Optimal**
1	18235	12	53.51	Yes
Ignore precondition	5040	12	18.97	Yes
levelsum	-	-	-	-

Analysis

Of the three heuristics, “IgnorePrecondition” seems the most promising. It expends fewer nodes and run faster overall. The levelsum heuristic do lead to an optimal solution but it doesn’t perform well. Even though it does not expand a lot of nodes, it cannot finish the search in the imparted time.
The heuristic planning algorithms are capable of finding optimal solution quicker than the non-informed algorithm for larger problems. The A* algorithm with the “Ignore Precondition Heuristic” converge to an optimal solution 6 time faster than the breath first search algorithm. This advantage is not clear on smaller problems though where the non-informed algorithm seems to perform as good if not better.

References

[1] Artificial Intelligence: A Modern Approach" 3rd edition