AI Planning and Search - Air Cargo Planning Analysis¶

This project solves the Air Cargo Transport problem using PDDL (Planning Domain Definition Language). An air cargo transport problem involves loading and unloading cargo and flying it from place to place. This is a classical AI planning problem with AI Planner or Agent has to come-up with cargo delivery plan under problem description and constraint.

Problem Description¶

Problem-1 (P-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)

Problem-2 (P-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)

Problem-3 (P-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)

Problem Complexity¶

Table below shows the state space complexity of three problems.

In [14]:

```
import pandas as pd
from IPython.display import display

state_space = pd.read_csv("problem_state.csv")
uninformed  = pd.read_csv("results_uninformed.csv")
informed    = pd.read_csv("results_informed.csv")
```

In [15]:

```
display(state_space)
```

	Problem	Fluents	State Space
0	P-1	12	2^12
1	P-2	27	2^27
2	P-3	32	2^32

Optimal Plans¶

Search agents using Breath First Search, Uniform Cost Search and A* under admissible heuristic have found the following plans for P-1, P-2 and P-3 which is guaranteed to be an optimal plan.

P-1¶

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

P-2¶

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)

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

Uninformed Search Performance Analysis¶

To search the state space of the problem under uninformed search category, Breath-First-Search (BFS), Depth-First-Search (DFS) and Uniform-Cost-Search (UCS) algorithms were chosen for all three problems. Table below shows the performance comparison of these three algorithms.

In [16]:

```
display(uninformed)
```

	Problem	Algorithm	Node Expansions	Goal Tests	New Nodes	Plan Length	Search Time (Seconds)	Optimal
0	P-1	Breath-First-Search	43	56	180	6	0.05537	Yes
1	P-1	Depth-First-Search	21	22	84	20	0.02622	No
2	P-1	Uniform-Cost-Search	55	57	224	6	0.06824	Yes
3	P-2	Breath-First-Search	3346	4612	30534	9	14.71962	Yes
4	P-2	Depth-First-Search	107	108	959	105	0.55493	No
5	P-2	Uniform-Cost-Search	4853	4855	44041	9	20.65752	Yes
6	P-3	Breath-First-Search	14120	17673	124926	12	72.21721	Yes
7	P-3	Depth-First-Search	292	293	2388	288	1.95240	No
8	P-3	Uniform-Cost-Search	18223	18225	159618	12	90.61325	Yes

As shown above in results, BFS and UCS are guaranteed to find the optimal path while DFS may or may not report an optimal path. As the space complexity increases, BFS and UCS need to explore exponentially more nodes than DFS. At the same time, DFS is reporting poor execution path which seems exponentially costlier than the optimal path as state space complexity is increased.
BFS and UCS are showing similar performance results, with UCS slightly higher cost than BFS. This is due to the node expansion in UCS for guaranteed optimality even when goal state has been reached but frontier has unexplored nodes of lower cost.

Informed Search Performance Analysis¶

To search the state space of the problem under informed search category, A* (A-Star) algorithm was chosen under two domain independent heuristics. These two heuristics were derived automatically by the problem state and not specific to the problem domain.

1. Ignore Preconditions Heuristic
   this heuristic was derived using relaxed constrained strategy. It estimated the goal distance from a given state by counting number of ground actions required to reach the goal, irrespective of pre-conditions required by those actions.

2. Graph Plan Level Sum Heuristic
   'Planning Graph' data structure is used to derive this heuristic. A planning graph is a polynomial size approximation to the tree that directs the path from initial state to goal state. Under the assumption of sub goal independence and decomposable nature of the problem, level sum heuristic returns the sum of the level costs of the goals. This heuristic has much higher accuracy than the above heuristic.

Table below shows the performance comparison of A* under these two heuristics for all three problems.

```
In [17]:
```

```
display(informed)
```

	Problem	Algorithm	Node Expansions	Goal Tests	New Nodes	Plan Length	Search Time (Seconds)	Optimal
0	P-1	A*-Relaxed-Heuristic	41	43	170	6	0.070370	Yes
1	P-1	A*-Graph-Plan	11	13	50	6	1.822440	Yes
2	P-2	A*-Relaxed-Heuristic	1450	1452	13303	9	7.830350	Yes
3	P-2	A*-Graph-Plan	86	88	841	9	318.069101	Yes
4	P-3	A*-Relaxed-Heuristic	5040	5042	44944	12	30.005930	Yes
5	P-3	A*-Graph-Plan	325	327	3002	12	1824.008730	Yes

A** algorithm under both heuristics returns the optimal execution plan for all three problems. As explained above, Level Sum Graph Plan heuristic shows better performance in terms of state exploration and is performing 'better informed' search than the relaxed heuristic. Although the search algorithm is suffering from time complexity of the heuristics, it is performing better search with small state exploration than uninformed search algorithms.

Summary¶

For informed search, Graph Plan Level Sum heuristic is showing it's search more accurately by controlling the numbers of nodes expansion. Although, it is doing the right thing, information required for this heuristic is computationally expensive (O(n(a + l)^2) for n levels, a actions and l literals). As compared to this, heuristic with relaxed problem (by ignoring pre-conditions on actions), is much cheaper to compute and performs better in larger state space.

Since informed searches have an extra cost of computing the heuristic, cost is worth computing for larger state space problems. For smaller state space problems (like P-1), uninformed searches would yield similar or better results.