Heuristic Analysis

In this project, we implement a planning search agent to solve an air cargo transportation system. We implement a **planning graph and automated domain-independent heuristic with A* search**. We also compare the results against several uninformed non-heuristic search methods such as **breadth first search (BFS)**, **depth first search (DFS) and uniform cost search (UCS)**.

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))
```

Initial States and Goals:

Problem 1	Problem 2	Problem 3
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))	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))	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))

Part 1: Non-heuristic search

Breadth First Search:

	p1	p2	p3
Expansion	43	3343	14663
Goal Test	56	4609	18098
New Nodes	100	30509	129631
Time Elapsed	0.034	14.36	109.36
Plan Length	6	9	12

Depth First Search:

	p1	p2	p3
Expansion	21	624	408
Goal Test	22	625	409
New Nodes	84	5602	3364
Time Elapsed	0.018	3.51	1.81
Plan Length	20	619	875

Uniform Cost Search:

	p1	p2	p3
Expansion	55	4853	18164
Goal Test	57	4855	18166
New Nodes	224	44041	159147
Time Elapsed	0.05	12.94	57.60
Plan Length	6	9	12

Optimal Plans:

P1:	P2:	P3:
Load(C1, P1, SFO)	Load(C1, P1, SFO)	Load(C1, P1, SFO)
Load(C2, P2, JFK)	Load(C2, P2, JFK)	Load(C2, P2, JFK)
Fly(P2, JFK, SFO)	Load(C3, P3, ATL)	Fly(P2, JFK, ORD)
Unload(C2, P2, SFO)	Fly(P2, JFK, SFO)	Load(C4, P2, ORD)
Fly(P1, SFO, JFK)	Unload(C2, P2, SFO)	Fly(P1, SFO, ATL)
Unload(C1, P1, JFK)	Fly(P1, SFO, JFK)	Load(C3, P1, ATL)
	Unload(C1, P1, JFK)	Fly(P1, ATL, JFK)
	Fly(P3, ATL, SFO)	Unload(C1, P1, JFK)
	Unload(C3, P3, SFO)	Unload(C3, P1, JFK)
		Fly(P2, ORD, SFO)
		Unload(C2, P2, SFO)
		Unload(C4, P2, SFO)

Analysis:

1. The minimum steps are 6, 9 and 12 for problem 1, 2 and 3 respectively, since breadth first search (BFS) guarantees path cost.
2. Depth first search (DFS) is significantly faster as it visits less nodes, however, the result is highly non-optimal as it keep going deeper until it finds a result.
3. For uniform cost search (UCS), since our cost is path cost, so it is essentially breadth first search. It also finds the optimal solution. However, breadth first search stops when it reaches the goal, while uniform cost search keeps going until it searches all nodes at the same level to minimize cost. Therefore, we can see here uniform cost search visits more nodes. In theory it should take more time than BFS, but in this case according to discussion in the forum the PriorityQueue implementation makes UCS faster.

4. As the problems get more complex, each search spends more time as it visits more nodes. The nodes visited grows exponentially.

Part 2: Planning Graph and Automated Heuristics with A* Search

Result:

Ignore Precondition Heuristic: it estimates the minimum number of actions that must be carried out from the current state in order to satisfy all of the goal conditions by ignoring the preconditions required for an action to be executed.

	p1	p2	p3
Expansion	41	1450	5038
Goal Test	43	1452	5040
New Nodes	170	13303	44924
Time Elapsed	0.036	4.80	18.50
Plan Length	6	9	12

Level Sum Heuristic: uses a planning graph representation of the problem state space to estimate the sum of all actions that must be carried out from the current state in order to satisfy each individual goal condition.

	p1	p2	p3
Expansion	44	3056	18203
Goal Test	46	3058	18205
New Nodes	178	27410	159619
Time Elapsed	0.77	496.00	3090.81
Plan Length	6	9	12

First Observation:

1. Both heuristics find an optimal plan because the both of them are admissible. Notice in general the level sum heuristic is not admissible but this problem is fine.
2. In theory the level sum heuristic should visit fewer nodes than other methods investigated here as the heuristic is a better estimate of cost. However, when calculating the heuristic itself it still runs through the tree. I ask the forum mentor and they say it may be caused by how the expansion is calculated in this case. However, I am still not sure about this explanation as i see others in the forum have results that visit few nodes. But in any case the level sum heuristic still returns an optimal plan.
3. The ignore precondition heuristic is better here as it takes less time and visit few nodes compared to level sum heuristic, as well as BFS and UCS. DFS doesn’t find the optimal

plan. So overall the ignore precondition is better as it is a relatively easy heuristic, while it is time consuming to calculate level sum.