Planning Search Implementation Report

In this project, our goal was to analyse the performance of different search algorithms on classical PDDL problems in the Air Cargo domain.

Air Cargo schema is defined as follows:

```
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))
```

We analysed the following three problems:

Problem 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:

```
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:

```
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))
```

Non-heuristic Search

At first, we experimented with uninformed algorithms, like depth-first search, breadth-first search and uniform cost search.

	Problem 1			Problem 2			Problem 3		
	Optimal	Time (s)	Nodes	Optimal	Time (s)	Nodes	Optimal	Time (s)	Nodes
Breadth-first search	Yes	0.03	43	Yes	11.41	3,346	Yes	80.77	14,120
Depth-first graph search	No	0.01	12	No	6.76	1,124	No	2.96	677
Uniform cost search	Yes	0.03	55	Yes	9.35	4,853	Yes	41.40	18,233

The more nodes an algorithm expands, the more memory it requires, but at the same time, there are more chances to find an optimal solution. Thus, uniform cost search and breadth-first search return the most logical solutions. Another disadvantage of such algorithms is their execution time. Breadth-first search requires the most time because it expands the paths uniformly in all directions. Uniform cost search performs better because it expands only those paths that have the same costs.

Depth-first search, on the other hand, is the quickest of the three and requires the least memory. It prioritises the paths that have the largest cost. However, the solutions it provides are far from being optimal. In particular, in our Air Cargo domain, it can send planes flying from one airport to another without any cargo simply because this is also an option that leads to fulfilling a goal.

Among the three, uniform cost search, provides a nice balance between memory usage, run time, and optimality. Here are the solutions that it found for the three problems.

Problem 1	Problem 2	Problem 3

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

Heuristic Search

A* search uses a heuristic function to estimate the distance to the goal. We experimented with two heuristics: “ignore preconditions” and “level-sum”.

	Problem 1			Problem 2			Problem 3		
	Optimal	Time (s)	Nodes	Optimal	Time (s)	Nodes	Optimal	Time (s)	Nodes
A* ignore preconditions	Yes	0.03	41	Yes	3.53	1,428	Yes	13.08	4,859
A* level-sum	Yes	1.43	11	Yes	172.90	114	Yes	700.37	306

Informed searches definitely have an advantage over uniformed searches because of an admissible heuristic – a heuristic that never overestimates the cost of the path from the current position to the goal. However, a heuristic plays an important role in the algorithm’s performance.

As we can see in our case, A* search with “ignore preconditions” uses more memory but less time compared with A* search with “level sum” heuristic. The solutions they provide have only slight variations to the order of Unload and Fly operations, which in our case don’t matter since we don’t take into account the time it takes to fly from on airport to another.

A* with “ignore preconditions” uses less memory and time than uniform cost search algorithm. While A* with “level-sum” provides the most optimal solution when the memory is a main concern.

Below, we provide the optimal plans provided by the A* with “ignore preconditions” algorithm for the three problems.

Problem 1	Problem 2	Problem 3

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