AIND Planner Analysis

Bertrand Louargant

I. Optimal Plans

As stated in the project enunciation, all problems are in the Air Cargo domain. They have the same action schema defined, but different initial states and goals.

- 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))
```

In the following tables the initial state and goal is displayed in the left column and its associated optimal plan is on the right.

- **Problem 1**:

Initial state and goal	Optimal plan contains **6 actions**
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))	**Load(C1, P1, SFO)** **Load(C2, P2, JFK)** **Fly(P1, SFO, JFK)** **Fly(P2, JFK, SFO)** **Unload(C1, P1, JFK)** **Unload(C2, P2, SFO)**

- **Problem 2** :

Initial state and goal	Optimal plan contains **9 actions**
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))	**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(C2, P2, SFO)** **Unload(C1, P1, JFK)**

- **Problem 3**:

Initial state and goal	Optimal plan contains **12 actions**
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))	**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)** **Fly(P2, ORD, SFO)** **Unload(C4, P2, SFO)** **Unload(C3, P1, JFK)** **Unload(C2, P2, SFO)** **Unload(C1, P1, JFK)**

II. Non-Heuristic Results

For commodity, all tests were executed overnight and a timeout of 10 minutes was used as an early stopping mechanism:

```
for P in 1 2 3; do
        for T in 1 2 3 4 5 6 7; do
                echo "# TEST $T" >> results.txt
                timeout 600s python run_search.py -p $P -s $T >> results.txt
        done
done
```

For each approach we note if the solution is either optimal or not as well as the path length, we also note the number of expansions needed (this give us an hint on the memory consumption of the search algorithm), and finally the execution time to measure its performance.

If the execution time exceed 10 minutes then the search is stopped and its results are discaded.

For a clearer and more synthetic view, results are displayed in 3 tables:

- **Problem 1**

Search Strategy	Is Optimal	Path Length	Expansions	Time Elapsed (s)
Breadth First Search	**Yes**	**6**	43	0.03
Breadth First Tree Search	**Yes**	**6**	1458	0.89
Depth First Graph Search	NO	20	21	0.01
Depth Limited Search	NO	50	101	0.09
Uniform Cost Search	**Yes**	**6**	55	0.03
Recursive Best First Search	**Yes**	**6**	4229	2.59
Greedy Best First Graph Search	**Yes**	**6**	**7**	**0.005**

- **Problem 2**

Search Strategy	Is Optimal	Path Length	Expansions	Time Elapsed (s)
Breadth First Search	**Yes**	**9**	3343	7.74
Breadth First Tree Search	*NA*	*NA*	*NA*	**Over 10 minutes**
Depth First Graph Search	NO	619	624	3.13
Depth Limited Search	*NA*	*NA*	*NA*	**Over 10 minutes**
Uniform Cost Search	**Yes**	**9**	4853	11.06
Recursive Best First Search	*NA*	*NA*	*NA*	**Over 10 minutes**
Greedy Best First Graph Search	NO	21	998	2.31

- **Problem 3**

Search Strategy	Is Optimal	Path Length	Expansions	Time Elapsed (s)
Breadth First Search	**Yes**	**12**	14663	39.05
Breadth First Tree Search	*NA*	*NA*	*NA*	**Over 10 minutes**
Depth First Graph Search	NO	392	408	1.68
Depth Limited Search	*NA*	*NA*	*NA*	**Over 10 minutes**
Uniform Cost Search	**Yes**	**12**	18223	49.38
Recursive Best First Search	*NA*	*NA*	*NA*	**Over 10 minutes**
Greedy Best First Graph Search	NO	22	5578	15.03

In the Air Cargo problem, as well as in many other planning problems, failing to find an optimal path drives cost upwards and sometime exponentially, thus disqualifying any search algorithm that cannot satisfy this condition.

In this regard, **Depth First Graph** Search and **Depth Limited Search** has to be discarded as they fail to meet this requirement even on a simple problem.

This also applies to **Greedy Best First Graph Search** that fails to find an optimal plan once the problem becomes slightly more difficult.

Conclusion: In this series of tests, the **Breadth First Search** seems to be the best algorithm, achieving the goal of finding an optimal plan in a time frame that is quite acceptable.
Two algorithms have been discarded (**Breadth First Tree Search** and **Recursive Best First Search**) because they were not able to find a path in less than 10 minutes but that doesn't make them worst candidates than algorithms that were not able to find an optimum. Actually it's quite the contrary, if they were able to find an optimal, then they would have been more suitable candidates, that's because computer resources are much cheaper than airplane transports' resources.

III. Heuristic Results

This time we will study the impact of using different heuristics to solve this planning problem.

First, let's summarize the results in the same way as we did before:

- **Problem 1**

Search Strategy	Is Optimal	Path Length	Expansions	Time Elapsed (s)
A* - H1	**Yes**	**6**	55	0.036
A* - Ignore Preconditions	**Yes**	**6**	41	0.028
A* - Level Sum	**Yes**	**6**	11	0.86

- **Problem 2**

Search Strategy	Is Optimal	Path Length	Expansions	Time Elapsed (s)
A* - H1	**Yes**	**9**	4853	11.19
A* - Ignore Preconditions	**Yes**	**9**	*1450*	3.46
A* - Level Sum	**Yes**	**9**	86	159.99

- **Problem 3**

Search Strategy	Is Optimal	Path Length	Expansions	Time Elapsed (s)
A* - H1	**Yes**	**12**	18223	49.45
A* - Ignore Preconditions	**Yes**	**12**	5040	14.11
A* - Level Sum	*NA*	*NA*	*NA*	**Over 10 minutes**

Of the three search heuristics, the **A* Search with Level Sum** is clearly behind in term of performances so much that it isn't even capable of finding a solution in less than 10 minutes.

Otherwise, all heuristics seems to be able to find an optimal path, with an advantage for the **A* Search with Ignore Preconditions** that is faster and consume less memory.

IV. Non-Heuristic Vs Heuristic search:

In this section we compare what emerges as the best solution of both approaches, **Breadth First Search** and **A* Search with Ignore Preconditions**.

We restrict the comparison to the third and most complex problem as both strategies seems to behave in the same way proportionally to the problem's complexity.

- **Problem 3**

Search Strategy	Is Optimal	Path Length	Expansions	Time Elapsed (s)
A* - Ignore Preconditions	**Yes**	**12**	5040	14.11
Breadth First Search	**Yes**	**12**	14663	39.05

In this head to head comparison we can clearly see the advantage of using a good heuristic as it will be able to find a solution much faster and at the same time consuming only one third of the memory used by the best non-heuristic algorithm.

Yet, finding the right heuristic for a given problem is a time consuming process and one that must be repeated for each new problem.

In this case, the Artificial Intelligence is located in the relevance of the algorithm written by the developper.