Heuristic Analysis for Deterministic Logistic Planning Problem

1 Overview

The aim of this project is to solve three deterministic logistic planning problem for the air cargo transport system and compare the performance of different planning search algorithms. We use several uninformed non-heuristic search algorithms (Breadth First Search, Depth First Graph Search, Uniform Cost Search etc.) as well as A* Search algorithm with some automatic heuristics (Ignore Preconditions and Level Sum).

2 Air Cargo Problem

Here we briefly summarize the three air cargo problems that we consider in this project (the descriptions by STRIPS given below are from *README.md* provided by Udacity). First of all, the three actions we consider in these problems are summarized in Figure 1. The three problems to solve are described in Figures 2, 3, 4. Examples of the optimal action plans for these problems are summarized in Appendix. The length for the optimal action plan(s) is **6** for Problem 1, **9** for Problem 2 and **12** for Problem 3.

```
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))
```

Figure 1: Description of 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))
```

Figure 2: Description of Problem 1

```
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))
```

Figure 3: Description of Problem 2

```
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))
```

Figure 4: Description of Problem 3

3 Performance

In this part, we summarize the performance of several search algorithms. For solving the problems, we have run the codes on iMac with 3.2 GHz Intel Core i5 and 16 GB memory.

3.1 Uninformed Non-heuristic Search

We first summarize the results of the uninformed non-heuristic searches. For the search algorithms, we have considered the seven algorithms as given in the tables below (we note that for Problem 2 and 3, we have run the code but we stopped in the middle of search for some algorithms since the algorithms seem to require a lot of time to find a solution).

Problem 1

The result is summarized in the following table:

Search Algorithm	Plan Length	Expansions	Goal Tests	New Nodes	Time (Sec.)	Optimality
Breadth First Search	6	43	56	180	3.7×10^{-2}	Yes
Breadth First Tree Search	6	1458	1459	5960	1.2	Yes
Depth First Graph Search	12	12	13	48	9.6×10^{-3}	No
Depth Limited Search	50	101	271	414	1.0×10^{-1}	No
Uniform Cost Search	6	55	57	224	4.4×10^{-2}	Yes
Recursive Best First Search	6	4229	4230	17029	3.3	Yes
Greedy Best First Graph Search	6	7	9	28	5.8×10^{-3}	Yes

Problem 2

The result is summarized in the following table (we note that for the algorithms with "-" inserted in the tables, we run the code but stopped in the middle of the search):

Search Algorithm	Plan Length	Expansions	Goal Tests	New Nodes	Time (Sec.)	Optimality
Breadth First Search	9	3346	4612	30534	1.0×10^{1}	Yes
Breadth First Tree Search	-	-	-	-	-	-
Depth First Graph Search	105	107	108	959	4.0×10^{-1}	No
Depth Limited Search	-	-	-	-	-	-
Uniform Cost Search	9	4853	4855	44041	1.5×10^{1}	Yes
Recursive Best First Search	-	-	-	-	-	-
Greedy Best First Graph Search	21	998	1000	8982	3.1	No

Problem 3

The result is summarized in the following table:

Search Algorithm	Plan Length	Expansions	Goal Tests	New Nodes	Time (Sec.)	Optimality
Breadth First Search	12	14120	17673	124926	5.2×10^1	Yes
Breadth First Tree Search	-	-	-	-	-	-
Depth First Search	660	677	668	5608	4.7	No
Depth Limited Search	-	-	-	-	-	-
Uniform Cost Search	12	18223	18225	159618	1.0×10^2	Yes
Recursive Best First Search	-	-	-	-	-	-
Greedy Best First Graph Search	22	5578	5580	49150	2.1×10^1	No

3.2 A* Search with Automatic Heuristic

We next summarize the results of A* searches with some automatic heuristics. For the heuristics, we have used the Ignore Preconditions Heuristics and Level Sum Heuristic (for the description of these heuristics, please refer to chapter 11.2 of AIMA book by Russel and Norvig). We note that A* Search with a constant heuristic is essentially the same as the Uniformed Cost Search analyzed above.

Problem 1

The result is summarized in the following table:

Search Algorithm	Plan Length	Expansions	Goal Tests	New Nodes	Time (Sec.)	Optimality
A* w/ Ignore Preconditions	6	41	43	170	4.3×10^{-2}	Yes
A* w/ Level Sum	6	11	13	50	1.5×10^1	Yes

Problem 2

The result is summarized in the following table:

Search Algorithm	Plan Length	Expansions	Goal Tests	New Nodes	Time (Sec.)	Optimality
A* w/ Ignore Preconditions	9	1450	1452	13303	5.3	Yes
A* w/ Level Sum	9	86	88	841	8.1×10^3	Yes

Problem 3

The result is summarized in the following table:

Search Algorithm	Plan Length	Expansions	Goal Tests	New Nodes	Time (Sec.)	Optimality
A* w/ Ignore Preconditions	12	5040	5042	44944	2.1×10^1	Yes
A* w/ Level Sum	12	325	327	3002	5.7×10^4	Yes

4 Comparison

In this part, we compare the performance of the search algorithms summarized above. We also discuss some reasoning of the obtained performance.

4.1 Among Uninformed Non-heuristic Search Algorithms

Some comparisons are in order:

- Among the algorithms we used, only Breadth First Search and Uniform Cost Search lead to the optimal action plan for all the three problems. This is what is expected since among the four algorithms with which we could complete the search for all the three problems, only these two algorithms are optimal. Breadth First Search requires a little bit less memories than Uniform Cost Search.
- From the point of view of time and memory consumption, the best algorithm is Depth First Graph Search, but the plan obtained by this algorithm is much longer than the optimal one.
- Greedy Best First Graph Search does not derive the optimal solution for Problem 2 and 3, but the plan length is not very long compared to the optimal one. In addition, the required memory is not very large and required time is somehow short.

Judging from the above, if one wants to give priority to optimality and time, then the best choice is **Breadth First Search**. If one wants to use an algorithm with a good balance among the memory consumption, time and optimality, then the best choice is **Greedy Best First Graph Search**.

4.2 Among A* Search Algorithms with Automatic Heuristics

Some comparisons are in order:

- The A* Search algorithms with the two different automatic heuristics both lead to the optimal action plans for all the three problems.
- The best algorithm from the point of view of time is the one with Ignore Preconditions Heuristic. The algorithm with Level Sum Heuristic is by far slower than the other.
- From the point of view of the use of memory, the best algorithm is the one with Level Sum Heuristic.

Judging from these, especially taking into account the fact that A* Search with Level Sum Heuristic consumes too much time, the best choice among these two is **A* Search with Ignore Preconditions Heuristic**.

4.3 Among All Search Algorithms

Among the three algorithms we highlighted above (Breadth First Search, Greedy Best First Graph Search, and A* Search with Ignore Preconditions Heuristic), the best choice is **A* Search with Ignore Preconditions Heuristic**. Among the three, this algorithm is the best from the point of view of memory consumption, time consumption as well as optimality.

As mentioned above, Uniform Cost Search is essentially the same as A* Search with a constant heuristic. We can see in the above tables that when Uniform Cost Search algorithm is compared with the Breadth First Search, the former is slower and consumes more memories. Thus, our result shows that combining Uniform Cost Search Algorithm with an appropriate non-trivial heuristic (i.e. Ignoring Preconditions Heuristics for the current case) reduces the consumption of both time and memory. This explains why A* Search with Ignore Preconditions Heuristic works well for the current problems.

5 Summary

In this project, we have compared the performance of several planning search algorithms for the air cargo problems. Our analysis shows that A* Search with Ignore Preconditions Heuristic is the best algorithm for this purpose. It would be interesting to apply this algorithm (as well al those highlighted in the previous section) to more complicated air cargo problems and analyze the performance further.

Appendix: Optimal Plans

In this Appendix, we provide an example of the optimal action plan(s) for Problem 1, 2, 3.

Problem 1

The optimal action plans for Problem 1 are of length 6 and an example is given as follows:

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

Problem 2

The optimal action plans for Problem 2 are of length 9 and an example is given as follows:

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)

Problem 3

The optimal action plans for Problem 2 are of length 12 and an example is given as follows:

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