Udacity Artificial Intelligence NanoDegree - Research Review

Fabio Quimio Pereira Fujii

May 29, 2018

1 Problem definition

This project aims to find the optimal sequence of actions through a planning graph. The problem simulates a simple air cargo logistic problem.

$$\begin{aligned} &Load(c,p,a): \\ &\qquad PRECOND: At(c,a) \land At(p,a) \land Cargo(c) \land Plane(p) \land Airport(a) \\ &\qquad EFFECT: \neg At(c,a) \land In(c,p) \end{aligned} \tag{1}$$

$$\begin{aligned} &Unload(c,p,a): \\ &\qquad PRECOND: In(c,p) \land At(p,a) \land Cargo(c) \land Plane(p) \land Airport(a) \\ &\qquad EFFECT: At(c,a) \land \neg In(c,p) \end{aligned} \tag{2}$$

$$\begin{aligned} &Fly(p,from,to): \\ &\qquad PRECOND: At(p,from) \land Plane(p) \land Airport(from) \land Airport(to) \\ &\qquad EFFECT: \neg At(p,from) \land At(p,to) \end{aligned} \tag{3}$$

2 Air Cargo Problem 1

$$\begin{aligned} Initial: &At(C1,SFO) \land At(C2,JFK) \\ &\land At(P1,SFO) \land At(P2,JFK) \\ &\land Cargo(C1) \land Cargo(C2) \\ &\land Plane(P1) \land Plane(P2) \\ &\land Airport(JFK) \land Airport(SFO) \\ Goal: &At(C1,JFK) \land At(C2,SFO) \end{aligned} \tag{4}$$

Table 1: Result table

Search Algorithm	**Plan Length**	**Time Lapsed**	**Nodes Expanded**
Breadth First Search	6	0.043	43
Depth First Search	20	0.021	21
Uniform Cost Search	6	0.053	55
A* Search Ignore Precond	6	0.054	41
A* Level Sum	6	1.584	11

The first problem is the simplest one. Considering all algorithms employed, the A* with the ignore precondition heuristic had the best performance. But, it is important to note that this

$$\begin{aligned} &Load(C1, P1, SFO)\\ &Fly(P1, SFO, JFK)\\ &Load(C2, P2, SFO)\\ &Fly(P2, JFK, SFO)\\ &Unload(C1, P1, JFK)\\ &Unload(C2, P2, SFO) \end{aligned} \tag{5}$$

1: Optimal Solution for problem 1

particular problem was simple enough to allow breadth first search to perform well, obtaining the optimal result. Naturally, Depth First Search was the fastest and lowest memory usage method, but having the worst solution. The A* with Level Sum heuristic was the slowest method among the tested ones. Almost all algorithms had the optimal result, with overall good performance both in time and memory.

3 Air Cargo Problem 2

$$\begin{aligned} Initial :&At(C1, SFO) \wedge At(C2, JFK) \wedge At(C3, CGO)\\ &\wedge At(P1, SFO) \wedge At(P2, JFK) \wedge At(P3, CGO)\\ &\wedge Cargo(C1) \wedge Cargo(C2) \wedge Cargo(C3)\\ &\wedge Plane(P1) \wedge Plane(P2) \wedge Plane(P3)\\ &\wedge Airport(JFK) \wedge Airport(SFO) \wedge Airport(CGO)\\ Goal :&At(C1, JFK) \wedge At(C2, SFO) \wedge At(C3, SFO) \end{aligned} \tag{6}$$

Table 2: Result table for problem 2

Search Algorithm	**Plan Length**	**Time Lapsed**	**Nodes Expanded**
Breadth First Search	9	12.519	3343
Depth First Search	619	4.605	624
Uniform Cost Search	9	18.389	4853
A* Search Ignore Precond	9	6.123	1450
A* Level Sum	9	279.954	86

$$\begin{aligned} &Load(C3, P3, CGO)\\ &Fly(P3, CGO, SFO)\\ &Unload(C3, P3, SFO)\\ &Load(C2, P2, JFK)\\ &Fly(P2, JFK, SFO)\\ &Unload(C2, P2, SFO)\\ &Load(C1, P1, SFO)\\ &Fly(P1, SFO, JFK)\\ &Unload(C1, P1, JFK) \end{aligned} \tag{7}$$

2: Optimal Solution for problem 2

Table 3: Result table for problem 3

Search Algorithm	Plan Length	Time Lapsed	Nodes Expanded
Breadth First Search	12	64.647	14663
Depth First Search	392	2.399	408
Uniform Cost Search	12	77.281	18223
A* Search Ignore Precond	12	24.307	5040
A* Level Sum	12	1166.292	315

In this problem, one more value for each variable was introduced: one more cargo, plane and airport, thus increasing the problem complexity exponentially . As seen in the table 2, the nodes expanded was significantly increased. We can identify a pattern within the results. Identically to the previous result table, except for the DFS algorithm, all algorithms achieved the optimal result. The slowest method was, once more, A* level sum; the most memory expensive method was UFC; and finally, the worst solution found was the DFS method.

4 Air Cargo Problem 3

$$
\begin{aligned}
\mathit{Initial} :&\mathit{At}(C1, SFO) \wedge \mathit{At}(C2, JFK) \wedge \mathit{At}(C3, CGO)\\
&\wedge \mathit{At}(C4, VCP) \wedge \mathit{At}(P1, SFO) \wedge \mathit{At}(P2, JFK)\\
&\wedge Cargo(C1) \wedge Cargo(C2) \wedge Cargo(C3)\\
&\wedge Cargo(C4) \wedge \mathit{Plane}(P1) \wedge \mathit{Plane}(P2)\\
&\wedge \mathit{Airport}(JFK) \wedge \mathit{Airport}(SFO) \wedge \mathit{Airport}(CGO)\\
&\wedge \mathit{Airport}(VCP)\\
\mathit{Goal} :&\mathit{At}(C1, JFK) \wedge \mathit{At}(C2, SFO) \wedge \mathit{At}(C3, JFK) \wedge \mathit{At}(C4, SFO)
\end{aligned}
\tag{8}
$$

$$
\begin{aligned}
&\mathit{Load}(C1, P1, SFO)\\
&\mathit{Fly}(P1, SFO, CGO)\\
&\mathit{Load}(C3, P1, CGO)\\
&\mathit{Fly}(P1, CGO, JFK)\\
&\mathit{Load}(C2, P2, JFK)\\
&\mathit{Fly}(P2, JFK, VCP)\\
&\mathit{Load}(C4, P2, VCP)\\
&\mathit{Fly}(P2, VCP, SFO)\\
&Unload(C4, P2, SFO)\\
&Unload(C3, P1, JFK)\\
&Unload(C2, P2, SFO)\\
&Unload(C1, P1, JFK)
\end{aligned}
\tag{9}
$$

3: Optimal Solution for problem 2

Finally, the third Air Cargo Problem is the most complex problem. The pattern is maintained within the tested algorithms: The fastest method is DFS, but with the worst solution; The least memory expensive algorithm is A* Level Sum, in contrast with its computing time; Curiously, the DFS algorithm had better performance in respect to the DFS used on the previous problem. This can be explained by the configuration of the problem 3.

5 Conclusion

For this heuristic analysis, 5 different algorithms were tested. A* algorithms outperformed all other algorithms tested, having the best Time Lapsed/Node Expanded values. The overall result indicates that A* Search with Ignore Precondition heuristic had the best result among all the optimal algorithms. If one want less memory usage, A* Search with Level Sum is recommended, due to its Nodes Expanded values.