AIND-Planning project: heuristic analysis

Fernando Martinelli Ramacciotti

***Abstract*— This work is a comparison of informed and uninformed search planning methods for three different logistic problems. Some uninformed methods fail to find an optimal solution while others don't - but this is due to the very nature of themselves. Informed searches, however, use heuristics to guide its search and were able to find the optimal solutions, through different ways, though. We conclude that the choice of planning method and heuristics play an essential role regarding algorithm performance together with the business problem at hand.**

I. INTRODUCTION

We investigated the performance of multiple planning algorithms on three different logistic problems. The so-called *air cargo* problems, share the same schema, but differ on initial states and goals. The solutions are part of the Udacity's Aritifial Intelligence Engineer Nanodegree (AIND) and have been developed in Python [1].

The methods studied ranges from uninformed and informed and belong to the class of breadth-first, depth-first and A* search. The heuristics used for informed planning are: constant cost, ignore precondition and level sum.

Some uninformed methods fail to find an optimal solution while others don't - but this is due to the very nature of themselves. Informed searches, however, use heuristics to guide its search and were able to find the optimal solutions, through different ways, though. We conclude that the choice of planning method and heuristics play an essential role regarding algorithm performance.

II. PROBLEM SET UP

A. Problem schema

The air cargo problem schema is set up 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))

[1]Source code can be found on https://github.com/fernandoramacciotti/nanodegree-ai-engineer/tree/master/projects/planning

B. Problem 1

The first air cargo problem is set up as follows:
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))

C. Problem 2

The second air cargo problem is set up as follows:
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))

D. Problem 3

The third air cargo problem is set up as follows:
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))

III. OPTIMAL SOLUTIONS

Optimal solutions for all three problems are described in Subsections III-A, III-B and III-C. There are algorithms that always find the optimal path towards the goal (provided an appropriate cost function, but this is not the scope of this study), such as breadth-first search [1], and, therefore, we use the results of them to provide here the optimal solutions.

A. Optimal solution - problem 1

An example of optimal solution, with path length 6, is as follows:
Load(C2, P2, JFK)
Load(C1, P1, SFO)
Fly(P2, JFK, SFO)
Unload(C2, P2, SFO)
Fly(P1, SFO, JFK)
Unload(C1, P1, JFK)

TABLE I

ALGORITHMS COMPARISON FOR PROBLEM 1

Algorithm	Heuristic	Path length	Time elapsed (s)	Optimal	Expansions	Goal tests	New nodes
Breadth First Search	n/a	6	.092	Yes	43	56	180
Depth First Search	n/a	12	.011	No	12	13	48
Uniform Cost Search	n/a	6	.050	Yes	55	57	224
A* Search	H1	6	.037	Yes	55	57	224
A* Search	Ignore Precond	6	.035	Yes	41	43	170
A* Search	Level Sum	6	.85	Yes	11	13	50

TABLE II

ALGORITHMS COMPARISON FOR PROBLEM 2

Algorithm	Heuristic	Path length	Time elapsed (s)	Optimal	Expansions	Goal tests	New nodes
Breadth First Search	n/a	9	12.99	Yes	3,346	4,612	30.534
Depth First Search	n/a	1085	14.74	No	1,124	1,125	10,017
Uniform Cost Search	n/a	9	28.75	Yes	4,853	4,855	44,041
A* Search	H1	9	11.84	Yes	4,853	4,855	44,041
A* Search	Ignore Precond	9	4.23	Yes	1,450	1,452	13,303
A* Search	Level Sum	9	150.1	Yes	86	88	841

TABLE III

ALGORITHMS COMPARISON FOR PROBLEM 3

Algorithm	Heuristic	Path length	Time elapsed (s)	Optimal	Expansions	Goal tests	New nodes
Breadth First Search	n/a	12	88.32	Yes	14,120	17,673	124,926
Depth First Search	n/a	660	7.45	No	677	678	5,608
Uniform Cost Search	n/a	12	112.7	Yes	18,223	18,225	156,618
A* Search	H1	12	51.58	Yes	18,223	18,225	156,618
A* Search	Ignore Precond	12	16.55	Yes	5,040	5,042	44,944
A* Search	Level Sum	12	845.3	Yes	316	318	2,912

B. Optimal solution - problem 2

An example of optimal solution, with path length 9, is as follows:

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)

C. Optimal solution - problem 3

An example of optimal solution, with path length 12, is as follows:

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

IV. PERFORMANCE COMPARISON

A. Uninformed searches

The uninformed searches were performed using breadth-first, depth-first and uniform cost algorithms and no heuristics. It is interesting to note that only depth-first could not find an optimal solution, although its performance is faster (except for problem 2) and expand much less nodes than the others. For memory constraints situation, this could be an advantage, but for a business perspective, it might be unfeasible. For instance, for problem 3, the optimal path length is 12 and the depth-first algorithm found a path of length 660, but 12 times faster. However, do 55 more actions than the optimal might be unfeasible. Also, in an algorithm perspective, if the branch factor grows toward infinite, this algorithm may never finds a solution. A summary of results can be found on Table I, Table II and Table III, for problems 1, 2 and 3, respectively.

B. Informed searches

The informed searches were performed using the A* search with three different heuristics: constant cost, ignore preconditions and level sum. All attempts found an optimal solution, but they differ in performance metrics. The constant cost heuristic is gives constant cost to all steps of the search,

therefore it may not be viewed as a real heuristic. Still, when used with A* search, the optimal solution was found for all three problems. The ignore preconditions heuristic, ignore the preconditions of all actions and gives a good estimate of the optimal number of actions. It also expanded more nodes than constant cost A*. The last heuristic, level sum, actually builds a planning graph and the cost is the graph level where all goals are satisfied (each goal can be satisfied at independently at different levels). Although the solution is slower, it expands fewer nodes than all the other approaches, informed and uninformed. A summary of results can be found on Table I, Table II and Table III, for problems 1, 2 and 3, respectively

V. Conclusion

It is interesting to notice that simple heuristics lead to optimal solutions and, among all, the planning graph approach with level sum heuristic gives the least nodes expansion at the expense of the slowest performance. The comparison made in this study helps to decide which approach to use according to the problem at hand. The tradeoff between optimal solution, time elapsed and memory requisites must be accounted for.

References

[1] Russell, Stuart J., and Peter Norvig. Artificial intelligence: a modern approach. Malaysia; Pearson Education Limited,, 2016.