Implementation of Planning Solvers – Heuristic Analysis*

Konrad Burnik

1 Background

The purpose of this project was to implement in Python several planning problems in the Air Cargo domain described in PDDL (Problem Domain Description Lanugage) as well as several search heuristics and then apply different search algorithms to solve them and compare the results.

2 Planning Problems

We are given three instances of the Air Cargo planning problems. Each problem is given as a pair of start and goal state. In general, to solve a planning problem means to find a sequence of actions which will start from the initial state and reach the goal state or otherwise indicate that such a sequence can not be obtained.

Initial state and goal of Air Cargo Problem 1:

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

*submitted as part of the Udacity AI Engineer Nanodegree

Initial state and goal of Air Cargo Problem 2:

$$\begin{aligned} Init(&At(C1, SFO) \land At(C2, JFK) \land At(C3, ATL) \\ &\land At(P1, SFO) \land At(P2, JFK) \land At(P3, ATL) \\ &\land Cargo(C1) \land Cargo(C2) \land Cargo(C3) \\ &\land Plane(P1) \land Plane(P2) \land Plane(P3) \\ &\land Airport(JFK) \land Airport(SFO) \land Airport(ATL)) \\ Goal(&At(C1, JFK) \land At(C2, SFO) \land At(C3, SFO)) \end{aligned}$$

Initial state and goal of Air Cargo Problem 3:

$$\begin{aligned} Init(&At(C1, SFO) \land At(C2, JFK) \land At(C3, ATL) \land At(C4, ORD) \\ &\land At(P1, SFO) \land At(P2, JFK) \\ &\land Cargo(C1) \land Cargo(C2) \land Cargo(C3) \land Cargo(C4) \\ &\land Plane(P1) \land Plane(P2) \\ &\land Airport(JFK) \land Airport(SFO) \\ &\land Airport(ATL) \land Airport(ORD)) \\ Goal(&At(C1, JFK) \land At(C3, JFK) \land At(C2, SFO) \land At(C4, SFO)) \end{aligned}$$

Available Air Cargo Actions that can be performed are given in PDDL as follows.

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

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

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

In the following, I present the gathered results by using `run_search.py` script provided by the Udacity staff. The script uses my own implemented descriptions of the Air Cargo problems from `my_air_cargo_problems.py` in an already provided implementation of PDDL as well as the search heuristics I implemented in `my_planning_graph.py` to execute the already provided search strategies and solve the Air Cargo problems. As per instruction by Udacity staff, the execution of search algorithms which took longer time were not included in these results.

Table 1: A list of provided search strategies

Search Strategy Name	Implemented method in `run_search.py`
BFS	breadth_first_search
BFTS	breadth_first_tree_search
DFS	depth_first_graph_search
DLS	depth_limited_search
UCS	uniform_cost_search
RBFS h_1	recursive_best_first_search h_1
GBFS h_1	greedy_best_first_graph_search h_1
$A^\star$ h_1	astar_search h_1
$A^\star$ h_ignore_preconditions	astar_search h_ignore_preconditions
$A^\star$ h_pg_levelsum	astar_search h_pg_levelsum

3 Results for uninformed search strategies

Here we present the data collected by running the `run_search.py` script. In the following tables lower values are better. Here, we present the results of running the uninformed search strategies for all three problems.

Table 2: Results for Air Cargo Problem 1

Search Strategy	Plan length	Is optimal?	Expansions	Goal Tests	New Nodes	Time (sec)
BFS	6	Yes	86	169	422	0.17
BFSTS	6	Yes	1894	1895	9232	1.26
DFS	12	No	12	13	48	0.03
DLS	50	No	101	271	414	0.22
UCS	6	Yes	55	57	224	0.10
RBFS with h1	6	Yes	4229	4230	17029	2.17
GBFS with h1	6	Yes	7	9	28	0.03

Table 3: Results for Air Cargo Problem 2

Search Strategy	Plan length	Is optimal?	Expansions	Goal Tests	New Nodes	Time (sec)
BFS	9	Yes	3343	4609	30509	3.14
BFSTS	–	–	–	–	–	–
DFS	1444	No	1669	1670	14863	2.31
DLS	50	No	222719	2053741	2054119	841.06
UCS	9	Yes	4852	4854	44030	4.60
RBFS with h1	–	–	–	–	–	–
GBFS with h1	21	No	990	992	8910	1.09

Table 4: Results for Air Cargo Problem 3

Search Strategy	Plan length	Is optimal?	Expansions	Goal Tests	New Nodes	Time (sec)
BFS	12	Yes	14663	18098	128605	15.14
BFSTS	–	–	–	–	–	–
DFS	195	No	3664	3665	29381	4.67
DLS	–	–	–	–	–	–
UCS	12	Yes	18235	18237	158272	17.02
RBFS with h1	–	–	–	–	–	–
GBFS with h1	26	No	5673	5675	49221	4.44

4 Results for informed search strategies

Here we present the results of running `run_search.py` informed search strategies (heuristics). The *h_ignore_preconditions* and *h_pg_levelsum* heuristics were implemented as part of this assignment. Lower values presented in the tables are better.

Table 5: Results for Air Cargo Problem 1

Search Strategy	Plan length	Is optimal?	Expansions	Goal Tests	New Nodes	Time (sec)
$A^\star$ with h1	6	Yes	55	57	224	0.10
$A^\star$ h_ignore_preconditions	6	Yes	41	43	170	0.14
$A^\star$ h_pg_levelsum	6	Yes	11	13	50	0.64

Table 6: Results for Air Cargo Problem 2

Search Strategy	Plan length	Is optimal?	Expansions	Goal Tests	New Nodes	Time (sec)
$A^\star$ with h1	9	Yes	4852	4854	44030	4.03
$A^\star$ h_ignore_preconditions	9	Yes	1450	1452	13303	2.44
$A^\star$ h_pg_levelsum	9	Yes	86	88	841	11.23

Table 7: Results for Air Cargo Problem 3

Search Strategy	Plan length	Is optimal?	Expansions	Goal Tests	New Nodes	Time (sec)
$A^\star$ with h1	12	Yes	18235	18237	158272	14.97
$A^\star$ h_ignore_preconditions	12	Yes	5040	5042	44769	9.40
$A^\star$ h_pg_levelsum	12	Yes	387	389	3550	95.47

5 Optimal plans obtained

5.1 Optimal plan for Air Cargo Problem 1

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

5.2 Optimal plan for Air Cargo Problem 2

$Load(C2, P2, JFK)$
$Load(C1, P1, SFO)$
$Load(C3, P3, ATL)$
$Fly(P2, JFK, SFO)$
$Unload(C2, P2, SFO)$
$Fly(P1, SFO, JFK)$
$Unload(C1, P1, JFK)$
$Fly(P3, ATL, SFO)$
$Unload(C3, P3, SFO)$

5.3 Optimal plan for Air Cargo Problem 3

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

6 Preprocessing and analysis of the data

The methods were compared in terms of optimality, speed and amount of memory. To analyze the gathered results I first preprocessed them for easier analysis.

First, only the search strategies for which I had complete measurements for *all three* problems were compared.

Second, for measuring the elapsed time I have summed up for each method all the times I've gathered on all three problems. The amount of memory can be measured in *Node Expansions* and also in *New Nodes* but since these two are highly positively correlated (when one grows then the other grows as well) then we can pick any one of them. In the following, we shall simply add them up into a new metric called *overall_exp_nn*. In this way, we can now rank the methods based on their *overall elapsed time* and *overall memory usage*.

Finally, I defined the new *always_optimal* as being 'Yes' if the method finds the plan of optimal length for *all three problems* and 'No' otherwise.

Below we present the results for both, the uninformed and informed search strategies, ranked by *overall elapsed time* and *overall memory usage*.

Table 8: Ranking by total time usage for uninformed search strategies

rank	method	always_optimal	overall_time_elapsed	overall_expansions_and_new
1	GBFS with h1	No	5.56	64829
2	DFS	No	7.01	49637
3	BFS	Yes	18.44	177628
4	UCS	Yes	21.72	225668

Table 9: Ranking by total memory usage for uninformed search strategies

rank	method	always_optimal	overall_time_elapsed	overall_expansions_and_new
1	DFS	No	7.01	49637
2	GBFS with h1	No	5.55	64829
3	BFS	Yes	18.44	177628
4	UCS	Yes	21.72	225668

Table 10: Ranking by total time elapsed for informed strategies

rank	method	always_optimal	overall_time_elapsed	overall_exp_nn
1	$A^\star$ h_ignore_preconditions	Yes	11.99	64773
2	$A^\star$ with h1	Yes	19.09	225668
3	$A^\star$ h_pg_levelsum	Yes	107.35	4925

Table 11: Ranking by total memory for informed search strategies

rank	method	always_optimal	overall_time_elapsed	overall_exp_nn
1	$A^\star$ h_pg_levelsum	Yes	107.35	4925
2	$A^\star$ h_ignore_preconditions	Yes	11.99	64773
3	$A^\star$ with h1	Yes	19.09	225668

7 Discussion

Based on the analysis of these gathered results, for the uninformed search strategies we first see that *Depth First Search* uses the least memory and is the fastest. However, the downside of DFS is that it does not always produce an optimal plan as we can see from the original gathered data. The best in terms of time and that was also consistently optimal is *Breadth First Search*, but it does use more memory. It is well known that BFS is a search strategy which finds all the shortest paths in terms of number of edges, between source and destination. This is somewhat intuitively clear from the definition of BFS, however, a rigorous proof of that fact can be found in any extensive algorithms textbook (see for example Theorem 22.5 in [1]). Therefore, the plans we obtained with BFS are indeed the shortest possible (optimal).

For the uninformed search strategies we see that $A^\star$ with *all three* heuristics find the optimal plan. In terms of time $A^\star$ *with h_ignore_preconditions* performs the best. If time is not critical, then $A^\star$ *with h_pg_level_sum* can be considered as well.

In the following table we show a comparison between the based informed strategy $A^\star$ *with h_ignore_preconditions* and the best uninformed search strategy we have, namely BFS.

Table 12: Final ranking of the best selected search strategies

rank	method	always_optimal	overall_time_elapsed	overall_exp_nn
1	$A^\star$ h_ignore_preconditions	Yes	11.99	64773
2	BFS	Yes	18.44	177628

We see that the informed heuristic is better. This is justified for example in the AIMA text ([2]) which argues that a search strategy which is more informed will likely reach the goal state faster as well as expand on the more useful states.

8 Conclusion

We conclude that for the Air Cargo problems, the $A^\star$ *with ignore preconditions heuristic* performs the best, both in terms of time, memory and optimality of obtained solution.

References

[1] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2009. Introduction to Algorithms, Third Edition (3rd ed.). The MIT Press.

[2] Russell, Stuart J. and Norvig, Peter, Artificial Intelligence: A Modern Approach, 2nd Edition, 2003.