P3-Planning-heuristic_analysis

Adam Liu

In this review, the non-heuristic search functions are going to be analysed regarding to their performances on the 3 cargo problems.

Problem 1:

```
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))
```

The Performance results are as below:

Search Function	Expansions	Goal Tests	New Nodes	Plan length	Time elapsed
1. breadth_first_search	43	56	180	6	0.024
2. breadth_first_tree_search	1458	1459	5960	6	0.738
3. depth_first_graph_search	21	22	84	20	0.011
4. depth_limited_search	101	271	414	50	0.095
5. uniform_cost_search	55	57	224	6	0.031
6. recursive_best_first_search w/ h_1	4229	4230	17023	6	2.0
7. greedy_best_first_graph_search w/ h_1	7	9	28	6	0.007
8. A* search w/ h_1	55	57	224	6	0.043
9. A* search w/ h_ignore_preconditions	41	43	170	6	0.036
10. A* search w/ h_pg_levelsum	11	13	50	6	0.853

One obvious thing in this table is that all the tasks were finished in relatively short amount of time, this might be due to the reason that the problem is relatively simple.
greedy_best_first_graph_search w/ h_1 is amazingly the best one out of all the 10 algorithms used, it showed the best results in all criteria.

Problem 2:

```
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))
```

The Performance results are as below:

Search Function	Expansions	Goal Tests	New Nodes	Plan length	Time elapsed
1. breadth_first_search	3343	4609	30509	9	9.179
3. depth_first_graph_search	624	625	5602	619	2.371
5. uniform_cost_search	4853	4855	44041	9	7.644
7. greedy_best_first_graph_search w/ h_1	998	1000	8982	15	1.734
8. A* search w/ h_1	4853	4855	44041	9	8.126
9. A* search w/ h_ignore_preconditions	1450	1452	13303	9	3.036
10. A* search w/ h_pg_levelsum	86	88	841	9	158.156

In this problem, **breadth_first_tree_search**, **depth_limited_search**, **recursive_best_first_search w/ h_1**, these 3 algorithms weren't even able to return a result within a reasonable time.

The interesting aspect of the results of this problem is that the most efficient algorithm was a non-heuristic algorithm, and the least efficient algorithm among the algorithms that returned a result was a heuristic algorithm.

greedy_best_first_graph_search w/ h_1 was the most efficient one, but didn't find the most optimised plan, the most optimised plan was found by **A* search w/ h_ignore_preconditions**, which was slightly slower than greedy algorithm.

However, in this problem, the non-heuristic algorithms outperformed the heuristic ones, in efficiency as well as the simplicity of the results. But in real life, the cost of choosing non optimal algorithms to solve some problems like in this case might be very high.

Problem 3:

```
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))
```

The Performance results are as below:

Search Function	Expansions	Goal Tests	New Nodes	Plan length	Time elapsed
1. breadth_first_search	14663	18098	129631	12	76.738
3. depth_first_graph_search	408	409	3364	392	1.250
5. uniform_cost_search	18223	18225	159618	12	35.044
7. greedy_best_first_graph_search w/ h_1	5578	5580	49150	22	10.529
8. A* search w/ h_1	18223	18225	159618	12	36.085
9. A* search w/ h_ignore_preconditions	5040	5042	44944	12	10.861
10. A* search w/ h_pg_levelsum	325	327	3002	12	799.534

In this problem, the **greedy_best_first_graph_search w/ h_1** algorithm used the least execution time and the **A* search w/ h_ignore_preconditions** algorithm returned the most optimised plan length. However, regardless of this result, the non-heuristic algorithm **depth_first_graph_search** performed most efficiently, used only almost 1/10 of the second efficient algorithm.

This problem is more complicated than problem 2, and the **breadth_first_tree_search**, **depth_limited_search**, **recursive_best_first_search w/ h_1** algorithms did not give any result within the reasonable time.

Comparison:
A* Ignore Preconditions and A* Planning Graph Level Sum:

These 2 algorithms have different approaches to the problem. **A* IP** estimates the minimum number of actions needed to satisfy all the goal states by ignoring the preconditions required to execute an action, while **A* PG_LS** sums up the levels where any literal of the goal first appears.

Plan Searching: Both algorithms found the most optimal solutions in all problems.

Time Elapsed: A* PG_LS algorithm took significantly longer time to find the most optimal plans in all problems than A* IP algorithm (*23.69, 52.09, 73.61* times more accordingly in each problem).

Other Aspects: In expanded nodes, goal tests as well as new nodes, A* IP was able to explore at least 3 times more than A* PG_LS.

Generated Plans:

Problem 1	Problem 2	Problem 3
Load(C1, P1, SFO)	Load(C1, P1, SFO)	Load(C2, P2, JFK)
Load(C2, P2, JFK)	Load(C2, P2, JFK)	Fly(P2, JFK, ORD)
Fly(P1, SFO, JFK)	Load(C3, P3, ATL)	Load(C4, P2, ORD)
Fly(P2, JFK, SFO)	Fly(P1, SFO, JFK)	Fly(P2, ORD, SFO)
Unload(C1, P1, JFK)	Fly(P2, JFK, SFO)	Unload(C4, P2, SFO)
Unload(C2, P2, SFO)	Fly(P3, ATL, SFO)	Load(C1, P1, SFO)
	Unload(C3, P3, SFO)	Fly(P1, SFO, ATL)
	Unload(C2, P2, SFO)	Load(C3, P1, ATL)
	Unload(C1, P1, JFK)	Fly(P1, ATL, JFK)
		Unload(C3, P1, JFK)
		Unload(C2, P2, SFO)
		Unload(C1, P1, JFK)

Conclusion:

A* search w/ Ignore Preconditions (heuristic) algorithm is the best heuristic algorithm used in those 3 presented problems, regarding the simplicity of the returned plan as well as the execution time the algorithm used. This algorithm did not just beat all the other heuristic algorithms in all problems, it also beat all the non-heuristic algorithms in complex problems (problem 2 & 3).

In chapter 10 of the book Artificial Intelligence: A Modern Approach, Peter Norvig said in the following sentence: *"An admissible heuristic can be derived by defining a relaxed problem that is*

easier to solve. The exact cost of a solution to this easier problem then becomes the heuristic for the original problem."

The reason of the outstanding performance of A* search w/ Ignore Preconditions algorithm is that **it relaxes the problem and not overestimates the goal**, and it was able to find the right direction of finding the optimal solution, the required calculation capability and time would contribute to the significance of the algorithm as well.

Reference:

Peter, N & Stuart J. R, 2009, *Artificial Intelligence: A Modern Approach (3rd Edition)*, Pearson