Planning Project Result Analyses

Solution and Optimal Plans

Most of search algorithms capable to find optimal plans for Air Cargo Problem. Due to nature of search, plans might have some variations of Load-Fly-Unload sequence, but in general plan is optimal (number of steps in plan is minimal).

Optimal plan for Problem 1:

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

Optimal plan for Problem 2:

```
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(C1, P1, JFK)
Unload(C3, P3, SFO)
Unload(C2, P2, SFO)
```

Optimal plan for Problem 3:

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

Depth First Search Algorithm in my research found not optimal plan for the solution (see Table 3). There is multiple solutions for current problem (even optimal solutions). Depth First Search Algorithm very efficient if solution is located very far from root tree, which is not the case in current domain and problem set.

In case of the other domain, where solution located far in tree DFS will be more practical to use. BFS algorithm might spend too much time expanding close located nodes, which could take extremely long time.

Depth First Search algorithm is not suitable to finding optimal plan for current set of problems and it is suggested to use Breadth First Search or A*-Search algorithms for searching optimal plan.

Non-heuristic search comparison

We going to compare three of non-heuristic algorithms: breadth first search (BFS), depth first graph search (DFGS) and uniform cost search (UCS).

Results showing that with problem size change searching time growing exponentially. For example, problem 2 has one more cargo, airport and plane, but still simple as P1, but search time grows from sub seconds to 4 seconds in case of BFS and UCS algorithms.

Depth first search time cost growing more linear, however solution that is found is not optimal. For P1 optimal solution contains 6 steps, but DFS found solution with 20 steps. As we discussed early, this is caused by algorithm expanding only one action, till it find solution or switches to search for another if no one was found in previous run. That fact that algorithm running on graph makes it finite, compare to search on infinite expanding tree, which might never find goal and stuck in loop of actions that interfere.

	breadth_first_search			**depth_first_graph_search**			**uniform_cost_search**		
Metric	*P1*	*P2*	*P3*	*P1*	*P2*	*P3*	*P1*	*P2*	*P3*
Time, s	0.118	4.32	20.39	0.044	0.86	0.60	0.101	5.09	18.97
Plan size	6	9	12	20	619	392	6	9	12
Expansions	43	3343	14663	21	624	408	55	4853	18223
Goal tests	56	4609	18098	22	625	409	57	4855	18225
New nodes	180	30509	129631	84	5602	3364	224	44041	159618

Table 1

Expansion and goal test and number of new nodes is relevant to problem complexity for all non-heuristic algorithms. As problem grows (minimal amount of steps required to find solution plan) algorithm have to expand more to find optimal solution.

Summary of non-heuristic search comparisons

In terms of research we can conclude that *Depth First Search* very efficient algorithm in finding solution of the problem in short time, however it doesn't guarantee solution to be optimal.

Breadth First Search and Uniform Cost Search good algorithms that doesn't require heuristics analyses to find solution of the problem, however running time is significantly more comparing to Depth First Search.

As domain becoming more complex, more actions have to be expanded and algorithm spend exponentially more time to find optimal solution.

A* Search heuristic search comparison

Next we are going to compare *A*-Search Algorithm* with BFS (non-heuristic) search algorithms. First to mention that all proposed algorithms find optimal solution for problem and have same plan size with order variation of actions (which could be technically executed in parallel).

As expected running time grows with complexity of the problem and number of actions to explore to find solution. *A*-Search* proves to be very efficient in solving problem, with heuristics that require minimum calculation, but yet gives overall good orientation to goal.

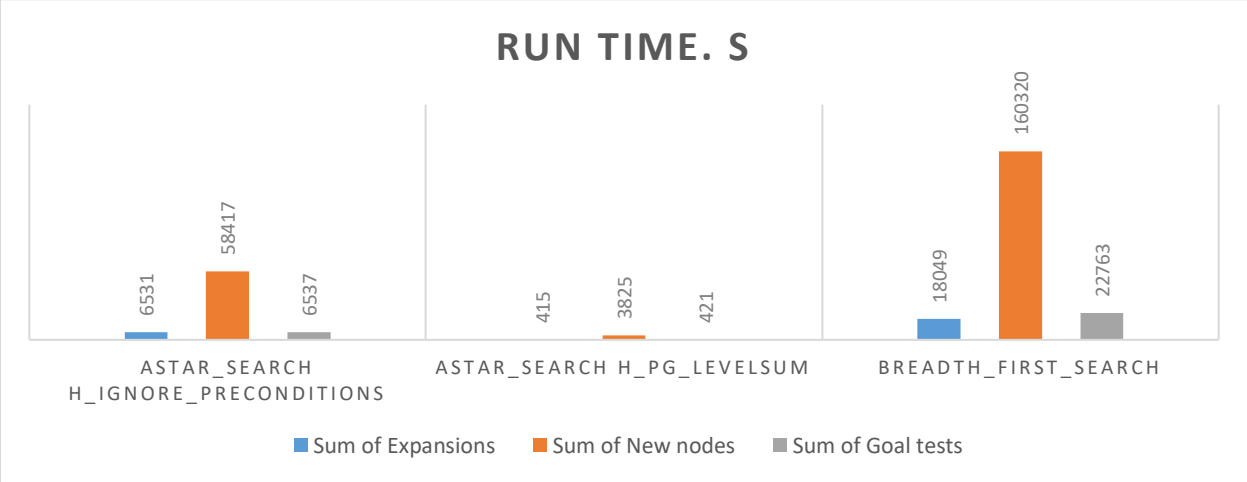

Figure 1

Counting number of unmet goal condition is very simple and efficient heuristics.

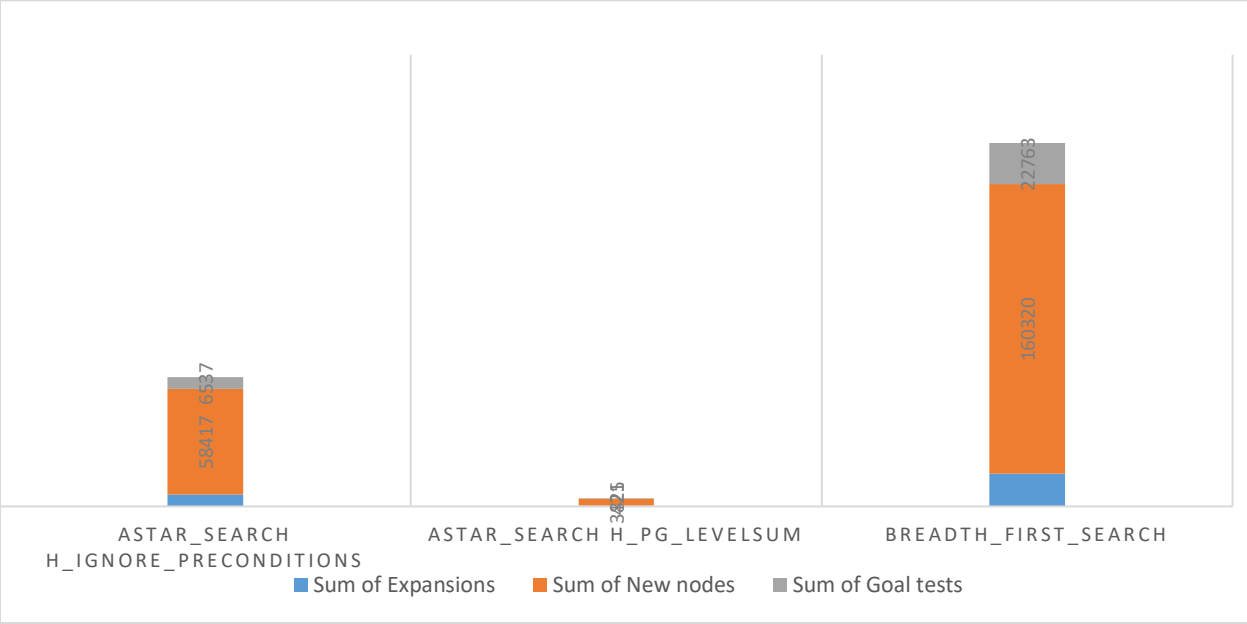

Figure 2

Results shows PlanGraph level sum is very efficient heuristic in terms of graph expansion and new nodes creation. Which means that heuristic can give better estimate *h(f).* Implementation of algorithm builds PlanGraph for every node, which leads to very poor performance comparing to other search algorithms. This heuristics will might be better for A* Search algorithm, in case of optimal run of PlanGraph.

Metric	breadth_first_search			A* h_pg_levelsum			A* h_ignore_preconditions		
	P1	*P2*	*P3*	*P1*	*P2*	*P3*	*P1*	*P2*	*P3*
Time, s	0.205	3.4	9.62	1.125	24.94	120.9	0.118	4.32	20.39
Plan size	6	9	12	6	9	12	6	9	12
Expansions	41	1450	5040	11	86	318	43	3343	14663
Goal tests	43	1452	5042	13	88	320	56	4609	18098
New nodes	170	13303	44944	50	841	2934	180	30509	129631

Table 2

Summary of A* Search heuristics algorithm comparison

A Search* Algorithm with *h_ignore_preconditions* heuristics shows best results in current review. Running complexity of grows more linear comparing to other algorithms and it is return optimal results plan for the problem.

Search run results

Following results was captured by running different search algorithms for 3 problems defined in class.

Results of running search for problem 1

Search Algorithm	Running time, s	Plan size	Expansions	Goal tests	New nodes
breadth_first_search	0.118	6	43	56	180
breadth_first_tree_search	1.181	6	1458	1459	5960
depth_first_graph_search	0.044	20	21	22	84
depth_limited_search	0.245	50	101	271	414
uniform_cost_search	0.101	6	55	57	224
recursive_best_first_search h_1	3.027	6	4229	4230	17023
greedy_best_first_graph_search h_1	0.021	6	7	9	28
astar_search h_1	0.109	6	55	57	224
astar_search h_ignore_preconditions	0.205	6	41	43	170
astar_search h_pg_levelsum	1.125	6	11	13	50

Table 3

Results of running search for problem 2

Search Algorithm	Running time, s	Plan size	Expansions	Goal tests	New nodes
breadth_first_search	4.32	9	3343	4609	30509
breadth_first_tree_search	running too long				
depth_first_graph_search	0.86	619	624	625	5602
depth_limited_search	918.89	50	222719	2053741	2054119
uniform_cost_search	5.09	9	4853	4855	44041
recursive_best_first_search h_1	running too long				
greedy_best_first_graph_search h_1	1.31	17	998	1000	8982
astar_search h_1	5.29	9	4853	4855	44041
astar_search h_ignore_preconditions	3.40	9	1450	1452	13303
astar_search h_pg_levelsum	24.94	9	86	88	841

Table 4

Results of running search for problem 3

Search Algorithm	Running time, s	Plan size	Expansions	Goal tests	New nodes
breadth_first_search	20.39	12	14663	18098	129631
breadth_first_tree_search	running too long				
depth_first_graph_search	0.60	392	408	409	3364
depth_limited_search	running too long				
uniform_cost_search	18.97	12	18223	18225	159618
recursive_best_first_search h_1	running too long				
greedy_best_first_graph_search h_1	5.92	22	5578	5580	49150
astar_search h_1	18.6	12	18223	18225	159618
astar_search h_ignore_preconditions	9.62	12	5040	5042	44944
astar_search h_pg_levelsum	120.9	12	318	320	2934

Table 5

References

[1] Stuart Russell and Peter Norvig, "Artificial Intelligence: A Modern Approach", 3rd Edition, Prentice Hall, 2009, ISBN 978-0136042594

[2] Udacity Artificial Intelligence Nanodegree Program, Fundamentals of AI by Peter Norvig and Thad Starner, 2017