Role of Heuristic in Search, Analysis

July 9, 2017

This analysis compares six search methods. Four are uninformed planning algorithms and other two A* search utilizing automatic heuristic: “ignore-preconditions” and “level-sum” from the Planning Graph. Comparison considers time, space efficiency and optimality of solution found. All search methods operates in graph of states of fully deterministic, fully observed environment. The evaluation is performed on four problems p1, p2, p3, p4 where p1 has smallest complexity and p4 is most complex. In order to eliminate random noise from our measurements caused by CPU utilization fluctuation on PC, each test was repeated 6 times. The mean value and standard error were calculated for each test and used in analysis. See table 1 in appendix A. Note, on p4 problem some tests where terminated by timeout. The values in table that represent timeout instead of actual measurement are highlighted in red. Such measurements will be highlighted on graph visualizations as well to avoid confusion with actual measurements. The source code used to collect data is available at *https://github.com/smile-on/aind-planning* for review and test of reproducibility.

Performance comparison.

Visual analysis of time and space complexity graphs (see pictures below) all search methods display exponential grows in time and space complexity.

By looking at solution quality Table 1 and time complexity pic 2, all search methods in the analysis can be separated into 3 groups:

- group **uninformed breadth-type**: breadth_first_search, uniform_cost_search;
  always find optimal solution but slow in most problems.
- group **uninformed depth-type**: depth_first_graph_search;
  always low quality solution but fast in most problems.
- group **informed heuristic**: greedy_best_first_graph_search_h_1, astar_search_h_ignore_preconditions, astar_search_h_pg_levelsum;
  good quality solutions most of the time.

Note 1. There is noticeable difference is time tendency among algorithms. At very simple problems, uninformative search methods finds solution much faster than A* with automatic heuristic. The only informed method that can compete in speed with uninformed searches on easy problems – greedy with h_1 heuristic. This is perfectly align with complexity of heuristic function. h_1 is not real heuristic and very fast. A* searches uses real heuristic that takes time to compute and limit practical applicability of A* search at low end of complexity. As complexity of problem rises, the breadth-type search time grows much faster than other groups and at p4 terminate by timeout that makes them not usable in most practical problems. The depth-type and A* searches display similar time efficiency grows. However depth-type searches returns solutions of low quality that deteriorates quickly with complexity that make A* searches are the only option for problems with significant complexity. Complex problems make trade-off speed vs quality very important (see p4 results in table 1). Solution quality of A* search depends on quality of heuristic. In our test ignore_precondition showed approximately same quality as pg_levelsum but little faster. At most complex problem p4 A* became absolute leader in quality and speed (see pic 2). Such efficiency of A* search can be explained by the core property of algorithm that explores only small portion of nodes at the search. That property is shown in section 3.5.2 in AIMA book which also sketches the proof that quality of solution indeed limited just by quality of heuristic and with admissible and monotonic heuristic A* would guaranty to find optimal solution. See quality of optimal solutions found by A* search with ignore-preconditions heuristic in table 2.

Note 2. Comparison of time and space complexity graphs shows that time of search is closely follows number of node expansions. At first order the time of search is proportional to number of node expansions. However each method has different penalty for node expansion. See pic 1 and pic 2.

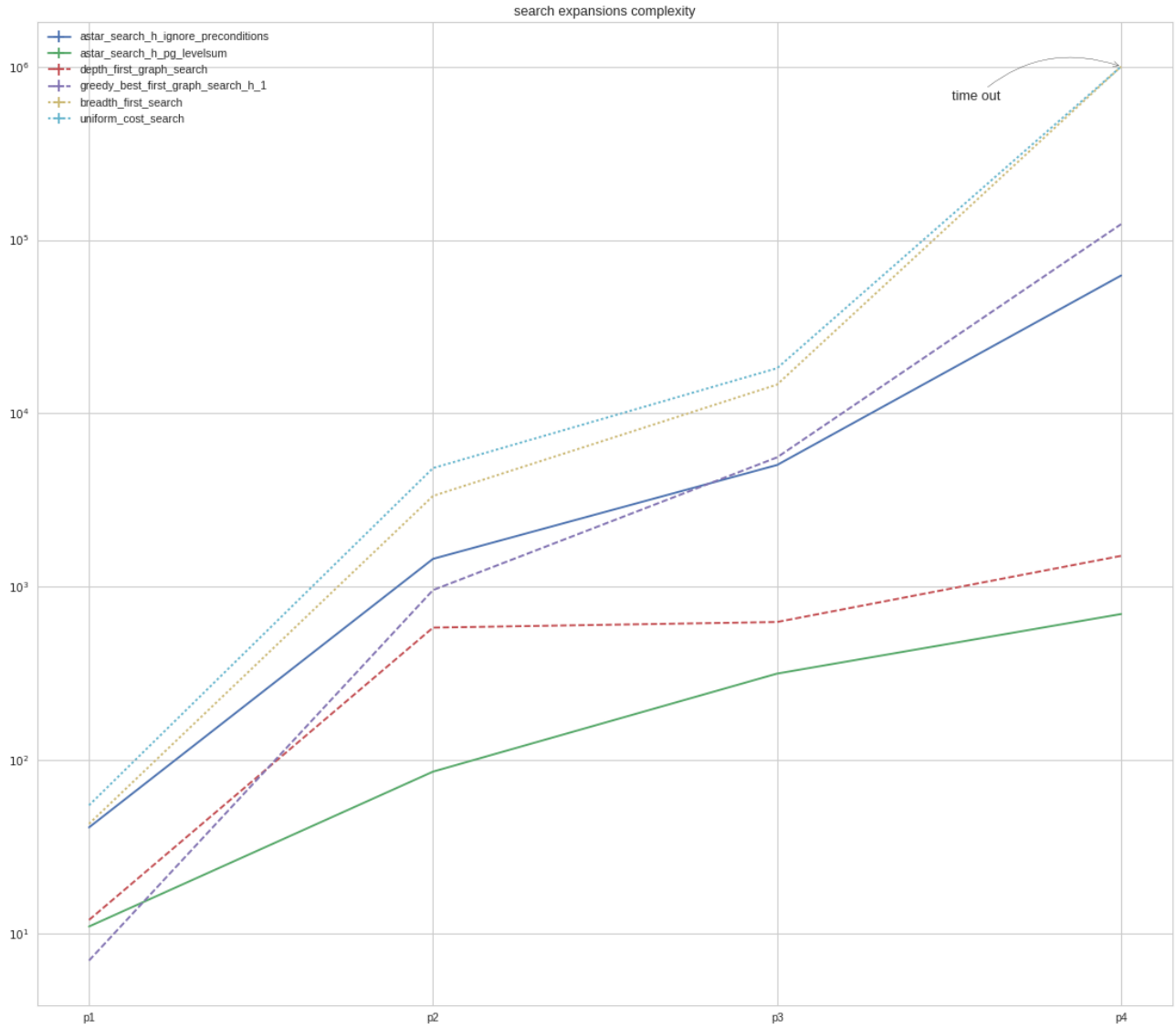
Note 3. A* search that uses planning graph heuristic demonstrates the lowest grows of space complexity and may be the best option for practical problems that bounded by memory space required to complete the search. See section 10.3.1 in AIMA book for examples of different automated heuristics available on planning graph.

		aggregate	mean	std
measure	**method**	**problem**		
Path length	**astar_search_h_ignore_preconditions**	**p1**	6*	0.0
		p2	9*	0.0
		p3	12*	0.0
		p4	15*	0.0
	astar_search_h_pg_levelsum	**p1**	6*	0.0
		p2	9*	0.0
		p3	12*	0.0
		p4	16	0.0
	breadth_first_search	**p1**	6*	0.0
		p2	9*	0.0
		p3	12*	0.0
		p4	NA	NA
	depth_first_graph_search	**p1**	12	0.0
		p2	575	0.0
		p3	596	0.0
		p4	1452	0.0
	greedy_best_first_graph_search_h_1	**p1**	6*	0.0
		p2	19	0.0
		p3	26	0.0
		p4	29	0.0
	uniform_cost_search	**p1**	6*	0.0
		p2	9*	0.0
		p3	12*	0.0
		p4	NA	NA

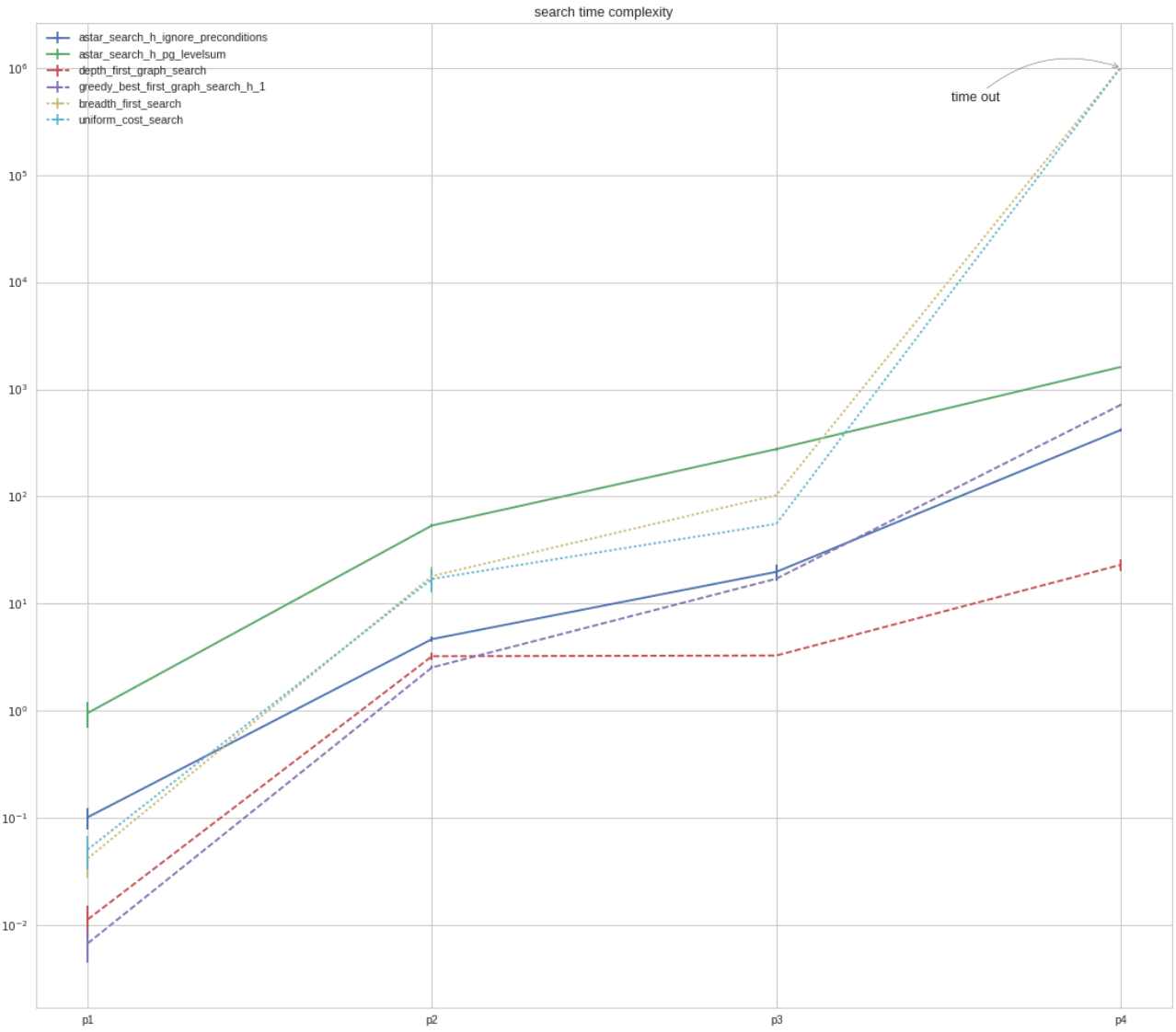
Table 1 Length of solution path found. * denotes solution of optimal size.

Table 2. Optimal plan samples. Found by A* search with ignore-preconditions heuristic.

Problem **p1**, Optimal length 6 steps: Load(C2, P2, JFK) Fly(P2, JFK, SFO) Unload(C2, P2, SFO) Load(C1, P1, SFO) Fly(P1, SFO, JFK) Unload(C1, P1, JFK)	Problem **p2**, Optimal length 9 steps: Load(C3, P3, ATL) Fly(P3, ATL, 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)
Problem **p4**, Optimal length 15 steps: Load(C2, P2, JFK) Fly(P2, JFK, ORD) Load(C4, P2, ORD) Fly(P2, ORD, SVO) Load(C5, P2, SVO) Fly(P2, SVO, SFO) Unload(C5, P2, SFO) Unload(C4, P2, SFO) Unload(C2, P2, SFO) Load(C1, P1, SFO) Fly(P1, SFO, ATL) Load(C3, P1, ATL) Fly(P1, ATL, JFK) Unload(C3, P1, JFK) Unload(C1, P1, JFK)	Problem **p3**, Optimal length 12 steps: Load(C2, P2, JFK) Fly(P2, JFK, ORD) Load(C4, P2, ORD) Fly(P2, ORD, SFO) Unload(C4, P2, SFO) Unload(C2, P2, SFO) Load(C1, P1, SFO) Fly(P1, SFO, ATL) Load(C3, P1, ATL) Fly(P1, ATL, JFK) Unload(C3, P1, JFK) Unload(C1, P1, JFK)



pic 1. Space complexity.



Pic 2 Time complexity.

Appendix A. Table 3 Raw data collected in each test.

		aggregate	mean	std
measure	**method**	**problem**		
time_elapsed	**astar_search_h_ignore_preconditions**	**p1**	0.10	0.02
		p2	4.67	0.27
		p3	19.81	3.50
		p4	416.38	17.44
	astar_search_h_pg_levelsum	**p1**	0.96	0.25
		p2	53.69	1.67
		p3	276.74	10.70
		p4	1613.92	18.44
	breadth_first_search	**p1**	0.04	0.01
		p2	18.10	3.96
		p3	102.77	0.69
		p4	1000000.00	0.00
	depth_first_graph_search	**p1**	0.01	0.00
		p2	3.24	0.26
		p3	3.29	0.01
		p4	23.03	2.83
	greedy_best_first_graph_search_h_1	**p1**	0.01	0.00
		p2	2.54	0.08
		p3	17.11	0.06
		p4	714.62	16.80
	uniform_cost_search	**p1**	0.05	0.02
		p2	17.02	4.22
		p3	55.57	0.15
		p4	1000000.00	0.00
expansions	**astar_search_h_ignore_preconditions**	**p1**	41.00	0.00
		p2	1450.00	0.00
		p3	5040.00	0.00
		p4	62385.00	0.00
	astar_search_h_pg_levelsum	**p1**	11.00	0.00
		p2	86.00	0.00
		p3	316.00	0.00
		p4	697.000	0.00
	breadth_first_search	**p1**	43.00	0.00
		p2	3343.00	0.00
		p3	14663.00	0.00
		p4	1000000.00	0.00
	depth_first_graph_search	**p1**	12.00	0.00
		p2	582.00	0.00
		p3	627.00	0.00
		p4	1509.00	0.00
	greedy_best_first_graph_search_h_1	**p1**	7.00	0.00
		p2	958.00	0.00
		p3	5578.00	0.00
		p4	123356.00	0.00
	uniform_cost_search	**p1**	55.00	0.00
		p2	4839.00	0.00
		p3	18221.00	0.00
		p4	1000000.00	0.00