Artem Arakcheev

September 6, 2017

Heuristic Analysis

Non-heuristic search result metrics

I present non-heuristic search result metrics in the following tables for Air Cargo Problems 1, 2 and 3. Used Breath First Search, Depth First Search, Uniform Cost Search algorithms.

Air Cargo Problem 1

Search Algorithm	breadth_first_search	depth_first_graph_search	uniform_cost_search
Node Expansions	43	21	55
Goal Tests	56	22	57
New Nodes	180	84	224
Plan length	6	20	6
Time elapsed, seconds	0.036	0.019	0.040

Air Cargo Problem 2

Search Algorithm	breadth_first_search	depth_first_graph_search	uniform_cost_search
Node Expansions	3343	624	4853
Goal Tests	4609	625	4855
New Nodes	30509	5602	44041
Plan length	9	619	9
Time elapsed, seconds	10.670	2.922	9.782

Air Cargo Problem 3

Search Algorithm	breadth_first_search	depth_first_graph_search	uniform_cost_search
Node Expansions	14663	408	18151
Goal Tests	18098	409	18153
New Nodes	129631	3364	159038
Plan length	12	392	12
Time elapsed, seconds	83.889	1.422	45.970

According the metrics, the most performant non-heuristic planning solution is Depth First Search algorithm. Depth First Search takes significantly shorter execution time, along with lowest node expansions, goal tests and new nodes for Air Cargo Problems. In the same time, Depth First Search doesn't give us optimal solution with minimal plan length.

Breath First Search and Uniform Cost Search algorithms give us much better results for plan length. Both algorithms gives relatively equal results on time elapsed, node expansions, goal tests and new nodes. But Uniform Cost Search took half of time on execution Air Cargo Problem 3. We can chose Uniform Cost Search as the best non-heuristic solution for Air Cargo Problems.

Breadth First Search is always expanding shortest paths first. It's going to find it by examining no longer paths. Breadth First Search is optimal [1]. As we see, Breadth First Search algorithm found an optimal paths for problems with lengths 6, 9, 12 accordingly.

Depth First Search tries to go as deep as it can first. It doesn't necessarily find the shortest path of all. It would find the longer path and find the goal there. And would not find the goal at the shortest path. Depth First Search is not optimal. Depth First Search is used in those cases of having storage requirements [1]. As possible to see, it didn't find an optimal paths (lengths are 20, 619 and 392 for problems 1, 2, 3 accordingly)

Uniform Cost Search firstly expands the paths according minimal costs. It's guaranteed to find the cheapest path of all assuming that all the individual step costs are non-negative. It's also optimal [1]. It also found shortest lengths of paths (6, 9, 12)

In state space of very large or even infinite binary tree, we can go to level one, two, three down to level n, the tree gets larger and larger. Let's consider the frontier of search algorithms: for Breadth First Search, when we get down to level n, we'll require storage space of 2^n paths; for Uniform Cost Search, the storage space is going to have similar total number of node; for Depth First Search, when we go down to tree branch to the deepest level, and the frontier is going to have only n notes. That's substantial savings for Depth First Search. When we keep track of explored set, we don't get that much savings. But without the explored set, Depth First Search has a huge advantage in terms of space saved [1].

One more property of the algorithm to consider is the property of completeness. If there is a goal somewhere, will the algorithm find it? Let's consider moving from very large trees to infinite trees, and let's say there is some goal hidden somewhere deep down in that tree.

Breadth First Search is complete, so even if the tree is infinite, if the goal is placed in any finite level, eventually, we're going to march down and find that goal. Same with Uniform Cost Search, if path has finite cost, the search algorithm goes down and find it. Depth First Search goes down and down to infinite level, but never get the path that the goal consists of. Depth First Search is not complete [1].

Heuristic search result metrics

Tables below contain result metrics for A* Search algorithm with constant number, ignore preconditions, level sum heuristics.

Air Cargo Problem 1

Search Algorithm	astar_search with h_1	astar_search with h_ignore_preconditions	astar_search with h_pg_levelsum
Node Expansions	55	41	11
Goal Tests	57	43	13
New Nodes	224	170	50
Plan length	6	6	6
Time elapsed, seconds	0.034	0.030	0.570

Air Cargo Problem 2

Search Algorithm	astar_search with h_1	astar_search with h_ignore_preconditions	astar_search with h_pg_levelsum
Node Expansions	4853	1450	86
Goal Tests	4855	1452	88
New Nodes	44041	13303	841
Plan length	9	9	9
Time elapsed, seconds	9.226	2.731	50.103

Air Cargo Problem 3

Search Algorithm	astar_search with h_1	astar_search with h_ignore_preconditions	astar_search with h_pg_levelsum
Node Expansions	18151	5038	314
Goal Tests	18153	5040	316
New Nodes	159038	44926	2894
Plan length	12	12	12
Time elapsed, seconds	45.361	12.862	270.045

The best heuristic

As we see in tables, A* Search algorithm with all presented heuristic functions gave out the same plan length in frame of solving Air Cargo Problem 1, 2 or 3. A* Search with ignore preconditions solved the problem most quickly. Ignore preconditions heuristic estimates the minimum number of actions that must be carried out from the current state in order to satisfy all of the goal conditions by ignoring the preconditions required for an action to be executed.

Constant number heuristic is not a true heuristic, and it has not good results with comparison with other heuristics.

Level sum heuristic uses a planning graph representation of the problem state space to estimate the sum of all actions that must be carried out from the current state in order to satisfy each individual goal condition. Level sum has longest execution time, but the best results for node expansions, goal tests, new nodes. So level sum heuristics is a good fit for those cases when the size of RAM matters.

Heuristic approach is not guaranteed to be optimal or perfect, but sufficient for the immediate goals. Heuristic methods can be used to speed up the process of finding a satisfactory solution.

According all result metrics (heuristic and non-heuristic), Breath First Search, Uniform Cost Search algorithms, A* Search with ignore preconditions and A* Search with level sum solved the Air Cargo problems with equal length of plan. Choosing from these 4 algorithms, A* Search with ignore preconditions has the best execution time and A* Search with level sum is optimized for memory usage.

So heuristic search algorithms are the better planners than non-heuristics for Air Cargo Problem on deterministic, fully observable, static environment.

A* Search algorithm works by always expending the path that has a minimum value of the function f, which is defined f = g + h, where function g (path) = path cost, and h(path) = h(state) estimated distance to the goal. Minimizing g helps us keep the path short, and minimizing h helps us keep focused on finding the goal. The result is a search strategy that is the best possible, in the sense that it finds the shortest length path, while expanding the minimum number of path possible [1].

A* is able to find the lowest cost path, depending on the heuristic estimate function h. A* finds the lowest cost path if h(state) < (less than) true cost of the path to the goal through the state.

Conditions when A* find the lowest cost path:

1) h should never overestimate the distance to the goal (h is optimistic), and
2) h is admissible to use it to find the lowest cost path [1].

A* Search with ignore-preconditions heuristics estimates the minimum number of actions that must be carried out from the current state in order to satisfy all of the goal conditions by ignoring the preconditions required for an action to be executed.

The ignore preconditions heuristic drops all preconditions from actions. Every action becomes applicable in every state, and any single goal fluent can be achieved in one step. This almost implies that the number of steps required to solve the relaxed problem is the number of unsatisfied goals—almost but not quite, because some action may achieve multiple goals and some actions may undo the effects of others. For many problems an accurate heuristic is obtained by considering and ignoring. First, we relax the actions by removing all preconditions and all effects except those that are literals in the goal. Then, we count the minimum number of actions required such that the union of those actions effects satisfies the goal. This is an instance of the set-cover problem. There is one minor irritation: the set-cover problem is NP-hard [2].

A* Search with level sum heuristic uses a planning graph representation of the problem state space to estimate the sum of all actions that must be carried out from the current state in order to satisfy each individual goal condition.

The level sum heuristic, following the subgoal independence assumption, returns the sum of the level costs of the goals; this can be inadmissible but works well in practice for problems that are largely decomposable [3].

A planning graph is polynomial in the size of the planning problem. For a planning problem with l literals and a actions, each S has no more than l nodes and l^2 mutex links, and each Ai has no more than a + l nodes (including the no-ops), $(a + l)^2$ mutex links, and 2(al + l) precondition and effect links. Thus, an entire graph with n levels has a size of $O(n(a + l)^2)$. The time to build the graph has the same complexity [4].

In this way, complexity of A* Search algorithm acquires additional complexity for level sum heuristic on building a planning graph. It gives us answer on why ignore preconditions heuristic is significantly faster than the level sum heuristic.

Optimal plan for Problems 1, 2, and 3

Multiple optimal solutions are possible for Air Cargo Problem domain. I have identified an optimal plan length for each problem (6, 9 and 12 actions accordingly).

Air Cargo Problem 1

For the Problem 1, Breath First Search found the optimal sequence of actions below:

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

Plan length: 6

Air Cargo Problem 2

A* Search with ignore preconditions solved problem with the following optimal actions sequence for the Problem 2:

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

Plan length: 9

Air Cargo Problem 3

A* Search with level sum gave us the following optimal sequence of actions 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(C1, P1, JFK)
Unload(C2, P2, SFO)
Unload(C3, P1, JFK)
Unload(C4, P2, SFO)
```

Plan length: 12

References

1. AIND video lessons, lesson 9 “Search”, Search Comparison 1, 2, 3;
2. Russell & Norvig, Artificial Intelligence: A Modern Approach 3ed., 10.2.3, Heuristics for planning;
3. Russell & Norvig, Artificial Intelligence: A Modern Approach 3ed., 10.3.1, Planning graphs for heuristic estimation.
4. Russell & Norvig, Artificial Intelligence: A Modern Approach 3ed., 10.3 PLANNING GRAPHS