# Planning Research Review – Planning Search

Author: Bruno Gonçalves de Aquino
Course: Udacity - Nanodegree in AI – Planning Search

**Optimal plan:**

**Problem 1:**

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

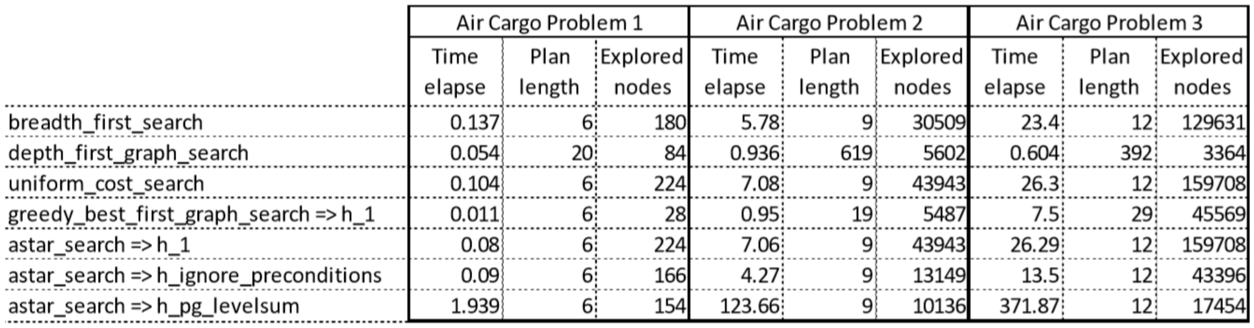
**Problem 2:**

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

**Problem 3:**

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

**Methods comparison:**



| | Air Cargo Problem 1 | | | Air Cargo Problem 2 | | | Air Cargo Problem 3 | | |
|---|---|---|---|---|---|---|---|---|---|
| | Time elapse | Plan length | Explored nodes | Time elapse | Plan length | Explored nodes | Time elapse | Plan length | Explored nodes |
| breadth_first_search | 0.137 | 6 | 180 | 5.78 | 9 | 30509 | 23.4 | 12 | 129631 |
| depth_first_graph_search | 0.054 | 20 | 84 | 0.936 | 619 | 5602 | 0.604 | 392 | 3364 |
| uniform_cost_search | 0.104 | 6 | 224 | 7.08 | 9 | 43943 | 26.3 | 12 | 159708 |
| greedy_best_first_graph_search => h_1 | 0.011 | 6 | 28 | 0.95 | 19 | 5487 | 7.5 | 29 | 45569 |
| astar_search => h_1 | 0.08 | 6 | 224 | 7.06 | 9 | 43943 | 26.29 | 12 | 159708 |
| astar_search => h_ignore_preconditions | 0.09 | 6 | 166 | 4.27 | 9 | 13149 | 13.5 | 12 | 43396 |
| astar_search => h_pg_levelsum | 1.939 | 6 | 154 | 123.66 | 9 | 10136 | 371.87 | 12 | 17454 |

**Breadth first search:** Very well method overall. Even being so simple, it delivers a good time elapse and find the optimal plan in every case, because it searches through the whole tree, ever, as stated in (Russell & Norvig, 2010, pg. 82): "…technically, breadth-first search is optimal if the path cost is a nondecreasing function of the depth of the node.", which is our case. But the good time elapse delivered is a little odd, because, in the same book, the space complexity of this method is O($b^d$), what makes it explore so many nodes.

**Depth first search:** It finds the solution very fast, no one can beat it, but it is ever a bad solution, in the problem 2 it brought a plan of 619 steps. This method can't be used for this type of problem, because it’s not optimal, and can do a mess in some problems like logical planning. (Russell & Norvig, 2010, pg. 86): “…depth-first search…both versions are nonoptimal. For example, in Figure 3.16, depth-first search will explore the entire left subtree even if node C is a goal node. If node J were also a goal node, then depth-first search would return it as a solution instead of C, which would be a better solution; hence, depth-first search is not optimal.”

**Uniform cost search:** It works practically as the A* using he heuristic 1, so it delivers the same results. And this method is optimal, as stated in the text book (Russell & Norvig, 2010, pg. 83): “When all step costs are equal, breadth-first search is optimal because it always expands the shallowest unexpanded node.”. In the case of our problem, all the steps costs equal, then this state fits.

**Greedy best first (using h_1):** It is like a depth first with a heuristic. The results aren’t optimal, but this method has some value, because it is extremally fast. So, if there is a great restriction, it can be used. But with caution, as (Russell & Norvig, 2010, pg. 93) says: “…hence, its search cost is minimal. It is not optimal, however… Greedy best-first tree search is also incomplete even in a finite state space…”.

**A* search (using h_1):** A*, most of the times is the best method to employ, but this test show that a poor, in this case, too much simple heuristic, can lost for a very simple method as the Breadth First. It’s the same method of Uniform cost search.

**A* search (using h_ignore_preconditions):** Here we have our champion, this method is my choice among the others compared here. It explores few nodes and can be computed very fast, this is the key of his victory. If we compare with the Planning Graph Levelsum, the Planning Graph explore less nodes, it shows that is more precise, but for each node, the Planning Graph has to create a Graph, and it is an expensive job. According with (Russell & Norvig, 2010, pg. 93), “A∗ search is both complete and optimal.” because it minimizes the total estimated solution cost. But it relies on a good heuristic, and (Russell & Norvig, 2010, pg. 376) says “An admissible heuristic can be derived by defining a relaxed problem that is easier to solve.” and “For example, the ignore preconditions heuristic drops all preconditions from actions.”

**A* search (using h_pg_levelsum):** Very sadly this one couldn’t beat the “Ignore Precondition” heuristic. As replenished before, it is very expensive to calculate a Planning Graph for every node explored. Maybe it can show his value with the Graphplan algorithm. (Russell & Norvig, 2010, pg. 382) “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. It is much more accurate than the numberof-unsatisfied-goals heuristic…”.