Algorithm Analysis

Problem

All problems are in the Air Cargo domain. They have the same action schema defined, but different initial states and goals.

Air Cargo Action Schema:

```
Action(Load(c, p, a),
        PRECOND: At(c, a) ∧ At(p, a) ∧ Cargo(c) ∧ Plane(p) ∧ Airport(a)
        EFFECT: ¬ At(c, a) ∧ In(c, p))
Action(Unload(c, p, a),
        PRECOND: In(c, p) ∧ At(p, a) ∧ Cargo(c) ∧ Plane(p) ∧ Airport(a)
        EFFECT: At(c, a) ∧ ¬ In(c, p))
Action(Fly(p, from, to),
        PRECOND: At(p, from) ∧ Plane(p) ∧ Airport(from) ∧ Airport(to)
        EFFECT: ¬ At(p, from) ∧ At(p, to))
```

Problem 1 initial state and goal:

```
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))
```

Problem 2 initial state and goal:

```
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))
```

Problem 3 initial state and goal:

```
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))
```

Algorithm Analysis

Algorithm Comparison

Algorithm	*Problem*	**Time**	**Expansions**	**Goal Tests**	**New Nodes**	**Plan length**
greedy_best_first_graph_search	*P1*	0.006	7	9	28	6
depth_first_graph_search	*P1*	0.009	12	13	48	12
astar_search_ignore_preconditions	*P1*	0.037	41	43	170	6
breadth_first_search	*P1*	0.047	43	56	180	6
uniform_cost_search	*P1*	0.500	55	57	224	6
astar_search_pg_levelsum	*P1*	1.446	11	13	50	6
greedy_best_first_graph_search	*P2*	2.276	385	387	3512	19
depth_first_graph_search	*P2*	3.500	582	583	5211	575
astar_search_ignore_preconditions	*P2*	14.600	1494	1496	13708	9
breadth_first_search	*P2*	15.560	3343	4609	30509	9
uniform_cost_search	*P2*	48.620	4823	4825	43774	9
astar_search_pg_levelsum	*P2*	142.180	86	88	841	9
depth_first_graph_search	*P3*	4.050	627	628	5176	596
greedy_best_first_graph_search	*P3*	92.998	4198	4200	37094	27
astar_search_ignore_preconditions	*P3*	101.920	5118	5120	45650	12
breadth_first_search	*P3*	114.440	14663	18098	129631	12
uniform_cost_search	*P3*	434.170	18235	18237	159716	12
astar_search_pg_levelsum	*P3*	997.174	405	407	3724	12

Summary:

1. Greedy best first graph search:

   GBFS has lowest time in all most all case but plan length is not always optimum as GBFS doesn't allow for revisit of decisions. Hence path may seems short at start but may have obstacle later which increase the plan length. While solving problem 3 greedy best first graph search takes a longer than DFS as search tree is wider.

2. Depth first graph search

   DFS has lowest time but largest plan length as DFS takes path further from start point. DFS has lower memory and time requirement as it's not necessary to store all of the child pointers at each level while running DFS.

3. Breath first Search

   Best-first search explores a graph by expanding the most promising node chosen according to a specified rule (shallowest nodes first). BFS always for revisit of decisions to find optimum path, hence it have higher search space than GBFS but is guaranteed to have optimum path.

Algorithm Analysis

4. Uniform cost search

   Uniform Cost Search does not involve the use of heuristics. It can solve any general graph for optimal cost. Uniform-cost search expands the node with lowest path cost and is optimal for general step costs.

5. A* with ignore preconditions

   Overall A* with ignore preconditions have lower time required, lowest expansions and optimum plan length among four remaining algorithm as heuristic function reduces the search space but time can increase depending on time required to run heuristic function.

6. A* with gp and level sum

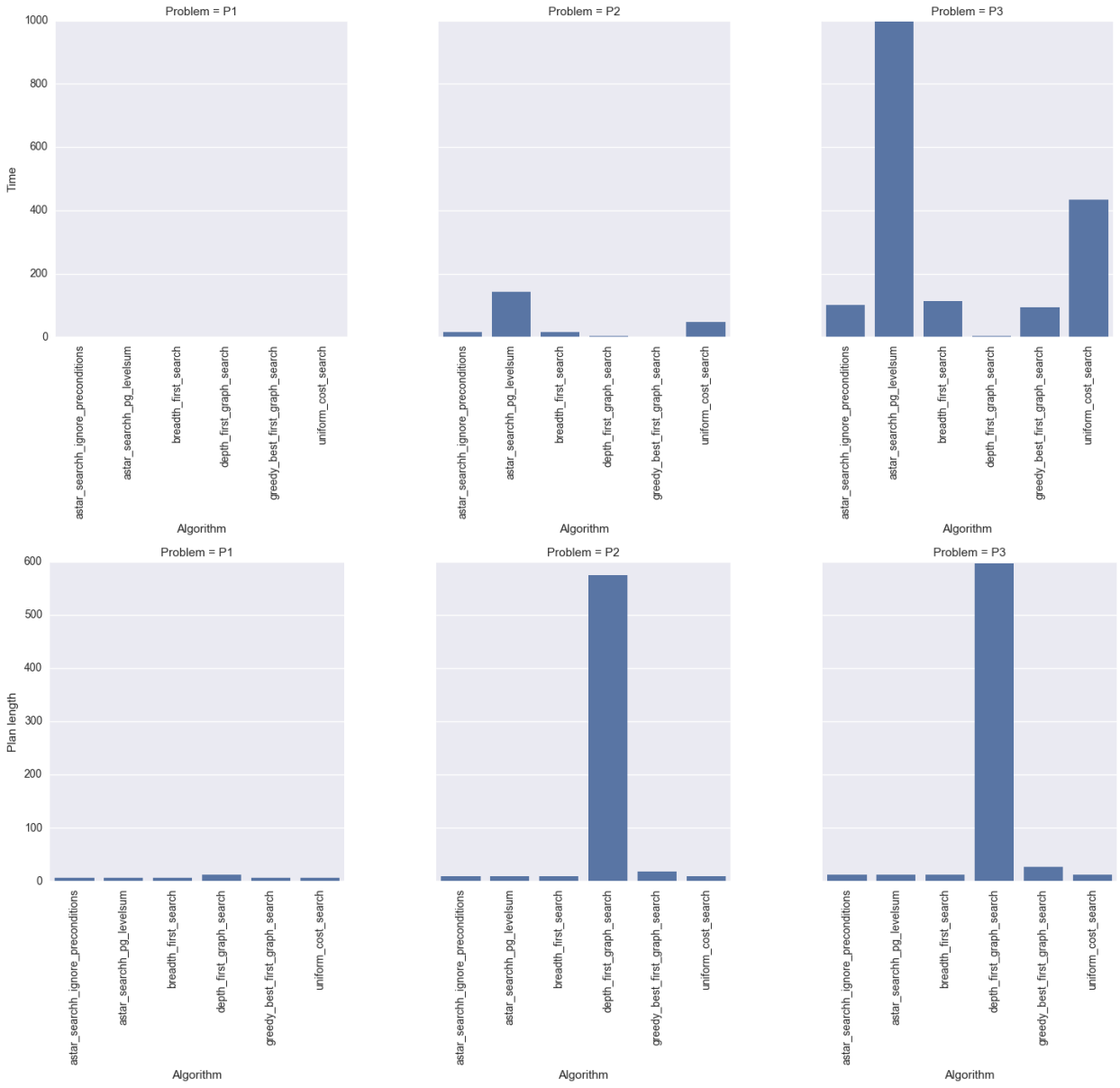
   A* with gp and level sum has lowest expansions and highest cost as gp and level sum significantly reduce search space but time required is high as algorithm built GRAPHPLAN at each heuristic estimated.

Conclusion:

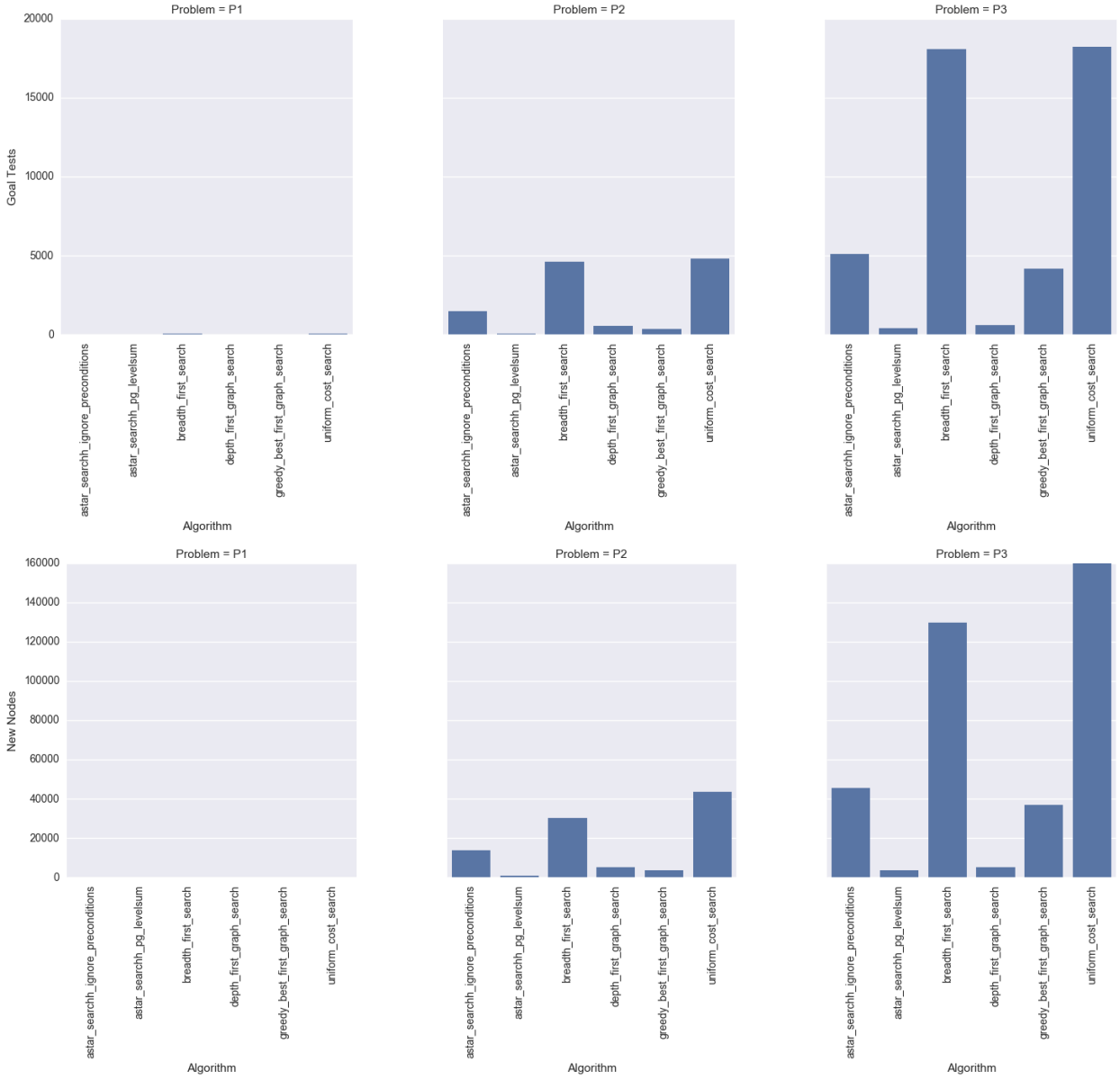
- If we need solution at fastest time and there is no cost function (optimum path is not required) than we can used GBFS or DFS depending on depth or width of tree. If you know a solution is not far from the root of the tree, a breadth first search (BFS) might be better. If the tree is very deep and solutions are rare, depth first search (DFS) might take an extremely long time, but BFS could be faster. If the tree is very wide, a BFS might need too much memory, so it might be completely impractical. If solutions are frequent but located deep in the tree, BFS could be impractical. If the search tree is very deep we will need to restrict the search depth for depth first search (DFS)
- If optimum path is need and we have good heuristic function A* search can be used, as it has lowest expansions, optimum path function and lowest time.
- If search space is too complex and time is not the constant than we can use A* with pg and level sum, as it drastically reduce the search space but requires a lot of time.

Algorithm Analysis

Algorithm Comparison



Algorithm Analysis



Algorithm Analysis