Heuristic Analysis

1. Problem 1

```
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))
```

Branching factor:
Minimum actions (all cargo and plan are at different airport): 2 (best case)
Maximum actions (all cargo and plan are at same airport): 4 (worst case)
Optimal Plan length: 6
Search space to the optimal plan in best case: 2^6=64
Search space to the optimal plan in worst case: 4^6=4096

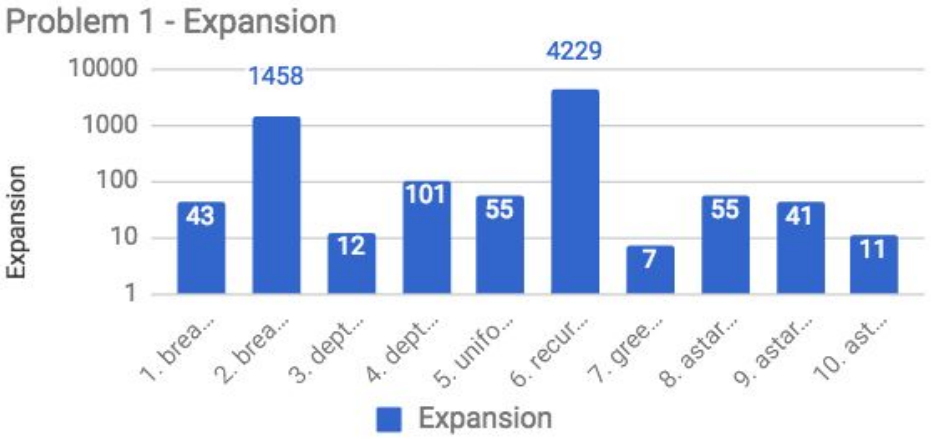
1.1 Optimal Plan

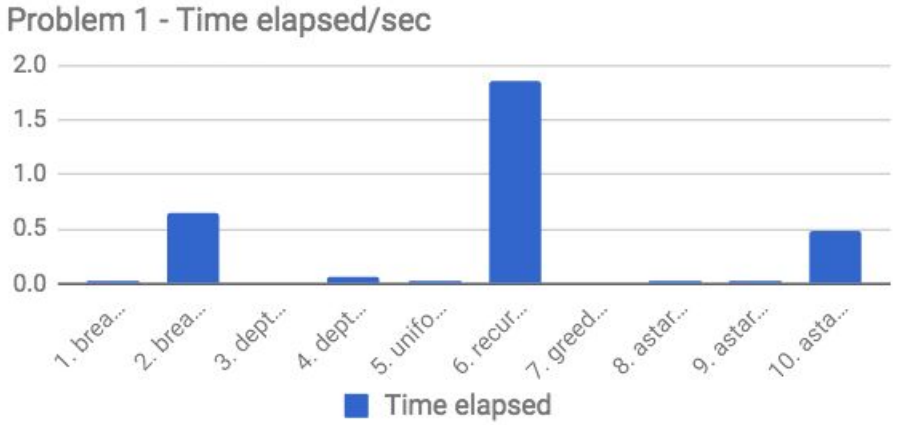
Load(C2, P2, JFK)
Load(C1, P1, SFO)
Fly(P2, JFK, SFO)
Unload(C2, P2, SFO)
Fly(P1, SFO, JFK)
Unload(C1, P1, JFK)

1.2 Experiment Result

Problem 1	Expansion	Goal Tests	New Nodes	Plan length	Time elapsed
1. breadth_first_search	43	56	180	6	0.02229524899
2. breadth_first_tree_search	1458	1459	5960	6	0.6499755
3. depth_first_graph_search	12	13	48	12	0.006858031004
4. depth_limited_search	101	271	414	50	0.06864740301
5. uniform_cost_search	55	57	224	6	0.02359542699
6. recursive_best_first_search h_1	4229	4230	17029	6	1.853603182
7. greedy_best_first_graph_search h_1	7	9	28	6	0.004320325999
8. astar_search h_1	55	57	224	6	0.026016294
9. astar_search h_ignore_preconditions	41	43	170	6	0.021457528

10. astar_search h_pg_levelsum	11	13	50	6	0.475881344





1.3 Non-heuristic Search Comparison

Among the non-heuristic search algorithms (1-5),

- depth_first_graph_search performs the best in terms of node expansion and time elapsed. However the plan is not optimal (longer plan length).
- Breadth_first_search, breadth_first_tree_search and uniform_cost_search find the optimal plan, but have more expansion and take longer to find the plan.
- Breadth_first_tree_search has the largest expansion and takes longest to find the the plan

1.4 Heuristic Comparison for A* Search

- Both heuristics find the optimal plan
- h_ignore_preconditions takes significantly less time than h_pg_levelsum to find the plan
- h_pg_levelsum expands much fewer nodes than h_ignore_preconditions

1.5 Best Algorithm for Problem1

In Problem 1, h_ignore_preconditions is a the best heuristic for A* algorithm. It outperforms h_ignore_preconditions in time elapsed while both heuristic help to find the optimal plan.

However in this problem, greedy_best_first_graph_search with pseudo heuristic h_1 performs the best in terms of time elapsed. It also finds the optimal plan. Second place algorithm in terms of time elapsed is depth_first_graph_search. However, the plan found using this algorithm is not optimal. Both algorithms performs faster than A* search in terms of time elapsed.

2. Problem 2

```
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))
```

Branching factor:
Minimum actions (all cargo and plan are at different airport): 6 (best case)
Maximum actions (all cargo and plan are at same airport): 15(worst case)
Optimal Plan length: 9
Search space to the optimal plan in best case: 6^9=10077696
Search space to the optimal plan in worst case: 15^9=38443359375

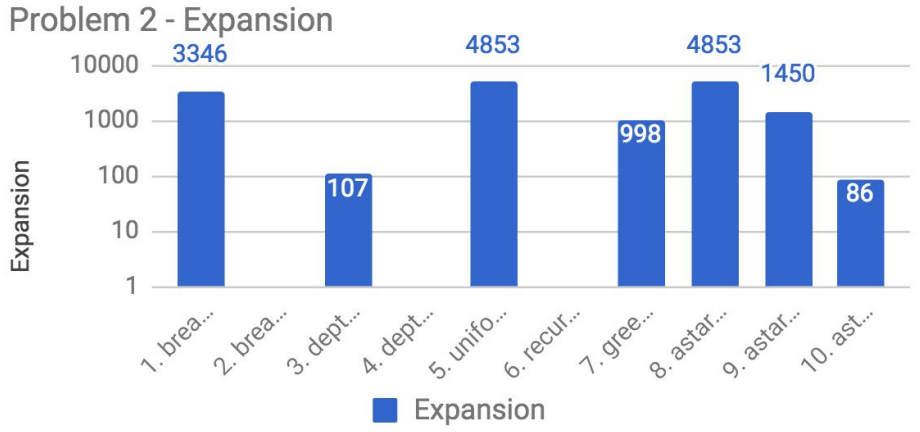
2.1 Optimal Plan

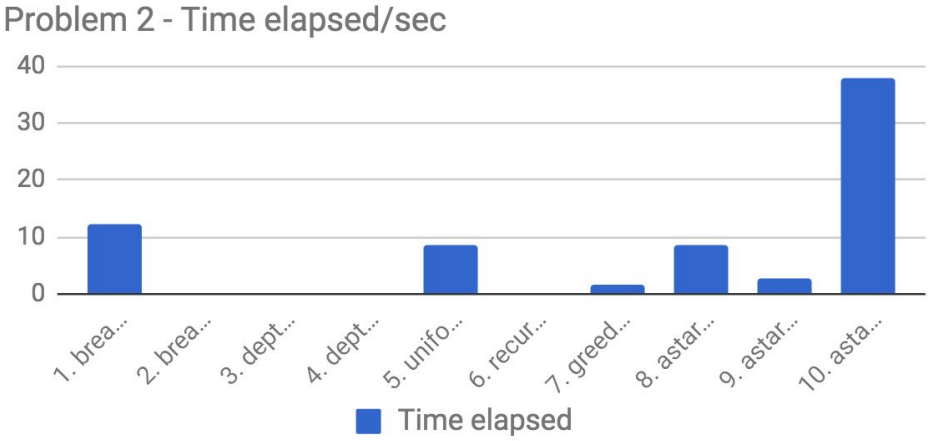
Load(C1, P1, SFO)
Load(C2, P2, JFK)
Load(C3, P3, ATL)
Fly(P1, SFO, JFK)
Fly(P2, JFK, SFO)
Fly(P3, ATL, SFO)
Unload(C3, P3, SFO)
Unload(C2, P2, SFO)
Unload(C1, P1, JFK)

2.2 Experiment Result

Problem 2	**Expansion**	**Goal Tests**	**New Nodes**	**Plan length**	**Time elapsed**
1. breadth_first_search	3346	4612	30534	9	12.1186639
2. breadth_first_tree_search	Not Complete	Not Complete	Not Complete	Not Complete	Not Complete
3. depth_first_graph_search	107	108	959	105	0.26402748
4. depth_limited_search	Not Complete	Not Complete	Not Complete	Not Complete	Not Complete
5. uniform_cost_search	4853	4855	44041	9	8.554315549
6. recursive_best_first_search h_1	Not Complete	Not Complete	Not Complete	Not Complete	Not Complete
7. greedy_best_first_graph_search h_1	998	1000	8982	21	1.735306931
8. astar_search h_1	4853	4855	44041	9	8.412159063
9. astar_search	1450	1452	13303	9	2.747029181

h_ignore_preconditions					
10. astar_search h_pg_levelsum	86	88	841	9	38.00680989





2.3 Non-heuristic Search Comparison

Among the non-heuristic search algorithms (1-5),

- Depth_first_graph_search explored least nodes and takes least time to find a plan. However, the plan found by this algorithm is not optimal. It is significantly worse than the optimal plan.
- Breadth_first_tree_search and depth_limited_search could not finish search within 10 minutes.
- Breadth_first_search and uniform_cost_search both find the optimal plan. They both takes significantly longer than Depth_first_graph_search.

2.4 Heuristic Comparison for A* Search

- Both heuristics find the optimal plan
- h_ignore_preconditions takes less time than h_pg_levelsum to find the plan
- h_pg_levelsum expands significantly fewer nodes than h_ignore_preconditions

2.5 Best Algorithm for Problem2

H_ignore_preconditions is a better heuristics for problem 2 as it takes less time than h_pg_levelsum to find the optimal plan.

Depth_first_graph_search is the fastest algorithm in finding a plan. But the plan found in this algorithm is significantly worse than the optimal plan. A* with h_ignore_preconditions is the second fastest in finding a plan and the plan found in this algorithm is optimal. Therefore A* with h_ignore_preconditions is the best algorithm for problem 2.

3. Problem 3

```
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))
```

Branching factor:
Minimum actions (all cargo and plan are at different airport): 6 (best case)
Maximum actions (all cargo and plan are at same airport): 14 (worst case)
Optimal Plan length: 12
Search space to the optimal plan in best case: 6^{12}=2176782336
Search space to the optimal plan in worst case: 14^{12}=56693912375296

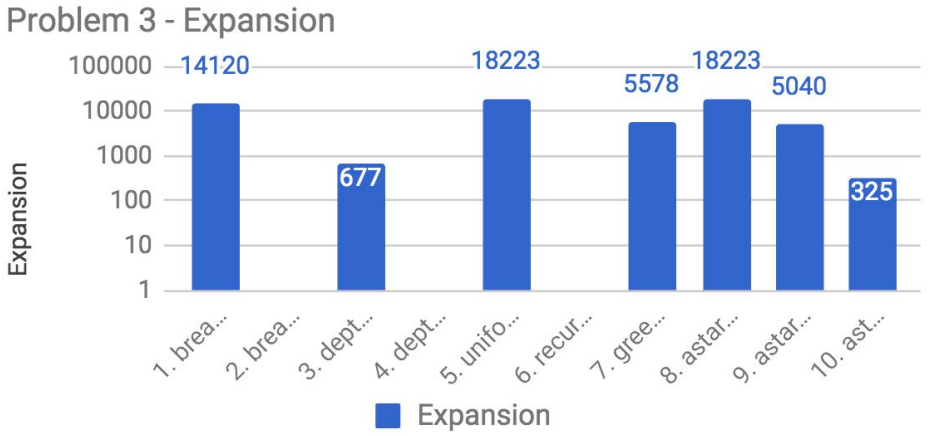
3.1 Optimal Plan

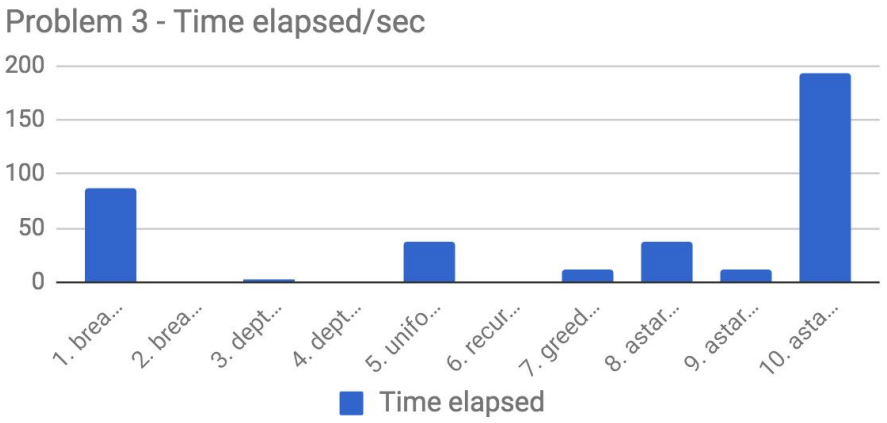
Load(C2, P2, JFK)
Fly(P2, JFK, ORD)
Load(C4, P2, ORD)
Fly(P2, ORD, SFO)
Load(C1, P1, SFO)
Fly(P1, SFO, ATL)
Load(C3, P1, ATL)
Fly(P1, ATL, JFK)
Unload(C4, P2, SFO)
Unload(C3, P1, JFK)
Unload(C2, P2, SFO)
Unload(C1, P1, JFK)

3.2 Experiment Result

Problem 3	**Expansion**	**Goal Tests**	**New Nodes**	**Plan length**	**Time elapsed**
1. breadth_first_search	14120	17673	124926	12	86.25933043
2. breadth_first_tree_search	Not Complete	Not Complete	Not Complete	Not Complete	Not Complete
3. depth_first_graph_search	677	678	5608	660	3.175898423
4. depth_limited_search	Not Complete	Not Complete	Not Complete	Not Complete	Not Complete
5. uniform_cost_search	18223	18225	159618	12	36.60460925

6. recursive_best_first_search h_1	Not Complete	Not Complete	Not Complete	Not Complete	Not Complete
7. greedy_best_first_graph_search h_1	5578	5580	49150	22	11.4355987
8. astar_search h_1	18223	18225	159618	12	37.28125859
9. astar_search h_ignore_preconditions	5040	5042	44944	12	11.00968205
10. astar_search h_pg_levelsum	325	327	3002	12	192.4254284





3.3 Non-heuristic Search Comparison

Among the non-heuristic search algorithms (1-5),

- Depth_first_graph_search explored least nodes and takes least time to find a plan. However, the plan found by this algorithm is not optimal. It is significantly worse than the optimal plan.
- Breadth_first_tree_search and depth_limited_search could not finish search within 10 minutes.
- Breadth_first_search and uniform_cost_search both find the optimal plan. They both takes significantly longer than Depth_first_graph_search.

3.4 Heuristic Comparison for A* Search

- Both heuristics find the optimal plan
- h_ignore_preconditions takes less time than h_pg_levelsum to find the plan
- h_pg_levelsum expands much significantly fewer nodes than h_ignore_preconditions

3.5 Best Algorithm for Problem 3

H_ignore_preconditions is a better heuristics for problem 3 as it takes less time than h_pg_levelsum to find the optimal plan.

Depth_first_graph_search is the fastest algorithm in finding a plan. But the plan found in this algorithm is significantly worse than the optimal plan. A* with h_ignore_preconditions is the second fastest in finding a plan and the plan found in this algorithm is optimal. Therefore A* with h_ignore_preconditions is the best algorithm for problem 3.

4. Algorithm Analysis for All Problems

Breadth_first_search in general finds optimal solution but explores more nodes and takes longer.

Breadth_first_tree_search finds optimal solution in small problem. As problem gets larger, it fails to find a solution within reasonable time (10 min in this experiment). The reason is that in a large problem like problem 2 or 3, the branching factor gets bigger and Breadth_first_tree_search wastes more time searching for all possible actions which take longer to converge to a solution.

Depth_first_graph_search performs fastest in all problems tested. However, it does not find optimal solution. The reason is Depth_first_graph_search the branch the algorithm choose in early depth are not likely to produce an optimal solution.

While in very small problem like problem 1, non-heuristic algorithms can perform better than heuristic algorithms. As problem getting more complex, such as problem 2 or 3, A* with good heuristics can produce more optimal plan within reasonable time. Non-heuristic search is prone to explore irrelevant branches.

H_ignore_preconditions performs well in problem 2 and 3 in terms of time elapsed and finds optimal solutions. H_pg_levelsum expands less node to find optimal solution but take significantly longer than H_ignore_preconditions. The reason is that H_pg_levelsum is more complex to compute.