Heuristic Analysis
Project: Implement a Planning Search

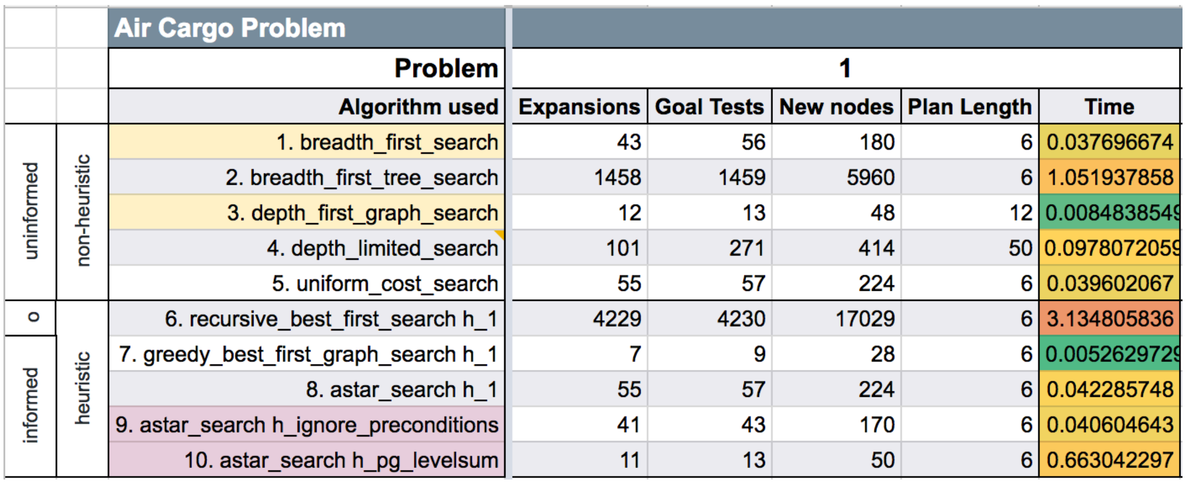
Performance Comparison

- Provide an optimal plan for Problems 1, 2, and 3.
- Compare and contrast non-heuristic search result metrics (optimality, time elapsed, number of node expansions) for Problems 1,2, and 3. Include breadth-first, depth-first, and at least one other uninformed non-heuristic search in your comparison; Your third choice of non-heuristic search may be skipped for Problem 3 if it takes longer than 10 minutes to run, but a note in this case should be included.
- Compare and contrast heuristic search result metrics using A* with the "ignore preconditions" and "level-sum" heuristics for Problems 1, 2, and 3.
- What was the best heuristic used in these problems? Was it better than non-heuristic search planning methods for all problems? Why or why not?
- Provide tables or other visual aids as needed for clarity in your discussion.

Comparing P1, P2 and P3

1 - Optimal plans

– Results for Problem 1:



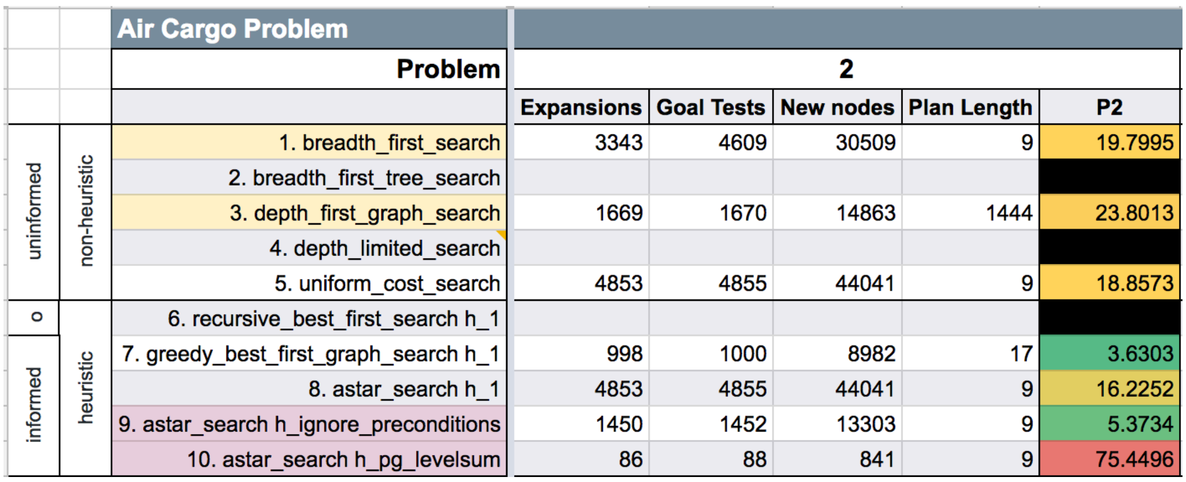
		Air Cargo Problem					
		Problem	1				
		Algorithm used	Expansions	Goal Tests	New nodes	Plan Length	Time
uninformed	non-heuristic	1. breadth_first_search	43	56	180	6	0.037696674
		2. breadth_first_tree_search	1458	1459	5960	6	1.051937858
		3. depth_first_graph_search	12	13	48	12	0.0084838549
		4. depth_limited_search	101	271	414	50	0.0978072059
		5. uniform_cost_search	55	57	224	6	0.039602067
o	heuristic	6. recursive_best_first_search h_1	4229	4230	17029	6	3.134805836
informed		7. greedy_best_first_graph_search h_1	7	9	28	6	0.0052629729
		8. astar_search h_1	55	57	224	6	0.042285748
		9. astar_search h_ignore_preconditions	41	43	170	6	0.040604643
		10. astar_search h_pg_levelsum	11	13	50	6	0.663042297

For problem 1, best execution time (0,005 seconds) uses **Greedy best first graph search** (an Informed / Heuristic search), even better than the execution time (0,008 seconds) of **Depth first graph search, DFS,** (a non-heuristic search). Our Greedy heuristic search finds the goal with a plan length of 6, expands less (only 7) and creates fewer nodes (only 28) than our best choice for non-heuristic DFS finds the goal with a plan length of 12, expands 12 and creates 48 new nodes.
So, our **optimal solution for Problem 1** is Greedy best first search.

Optimal plan:

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

- Results for Problem 2:



		Air Cargo Problem					
		Problem	2				
			Expansions	Goal Tests	New nodes	Plan Length	P2
uninformed	non-heuristic	1. breadth_first_search	3343	4609	30509	9	19.7995
		2. breadth_first_tree_search					
		3. depth_first_graph_search	1669	1670	14863	1444	23.8013
		4. depth_limited_search					
		5. uniform_cost_search	4853	4855	44041	9	18.8573
o	heuristic	6. recursive_best_first_search h_1					
informed		7. greedy_best_first_graph_search h_1	998	1000	8982	17	3.6303
		8. astar_search h_1	4853	4855	44041	9	16.2252
		9. astar_search h_ignore_preconditions	1450	1452	13303	9	5.3734
		10. astar_search h_pg_levelsum	86	88	841	9	75.4496

For problem 2, best execution time (3,630 seconds) uses **Greedy best first graph search**, even better than the execution time (5,373 seconds) of **A* search with "ignore preconditions" heuristic,** both heuristic search algorithms. Our Greedy heuristic search finds the goal with a plan length of 17, expands less (only 998) and creates fewer nodes (only 8982) than A* finds the goal with a plan length of 9 (more optimal than Greedy), expands 1450 and creates 13303 new nodes.
White squares rows (Breadth First Tree Search, Depth Limited Search and Recursive Best First Search h_1) are algorithms that were skipped for Problem 2 because it took longer than 10 minutes to run.
Taking into account number of expansions and new nodes, the one that expands less and creates fewer nodes is **A* with "levelsum" heuristic**, although this option is the one that takes more time to reach the goal, because the implementation for this heuristic may be not optimal.
Based on time execution, the optimal solution for **Problem 2** should be Greedy best first search, but its length path is 17 steps. Not so good.

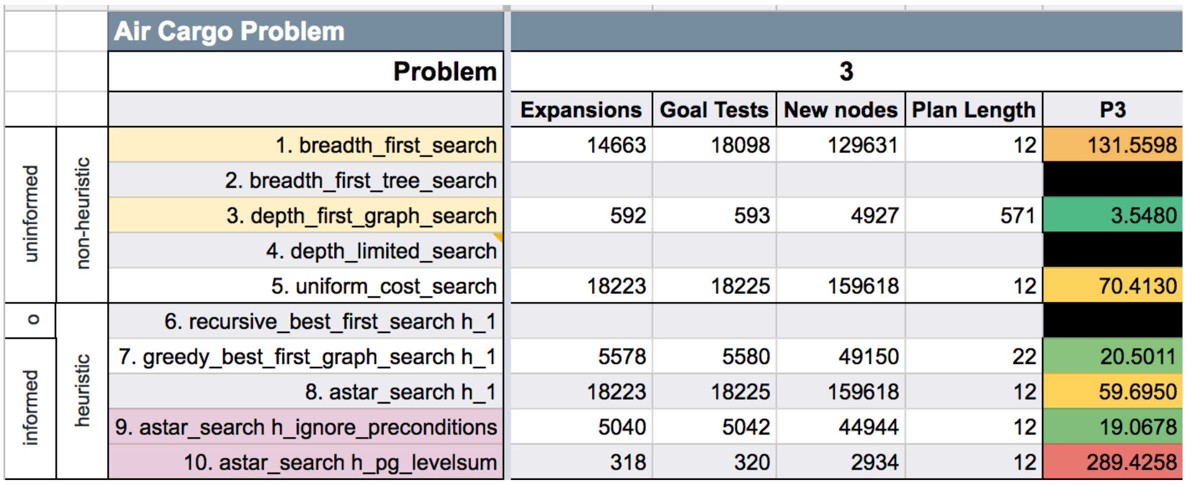
```
Solving Air Cargo Problem 2 using
greedy_best_first_graph_search with h_1...

Expansions   Goal Tests   New Nodes
   998          1000         8982

Plan length: 17  Time elapsed in seconds: 5.206292458999087
Load(C1, P1, SFO)
Load(C2, P2, JFK)
Load(C3, P3, ATL)
Fly(P1, SFO, ATL)
Fly(P2, JFK, ATL)
Fly(P3, ATL, JFK)
Fly(P1, ATL, JFK)
Unload(C1, P1, JFK)
Load(C1, P3, JFK)
Fly(P1, JFK, ATL)
Fly(P2, ATL, SFO)
Unload(C2, P2, SFO)
Fly(P2, SFO, ATL)
Fly(P3, JFK, SFO)
Unload(C3, P3, SFO)
Fly(P3, SFO, JFK)
Unload(C1, P3, JFK)
```

In terms of finding the **optimal plan** length in less time, the best is **A* search with "ignore preconditions"**.

Optimal plan:

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

- Results for Problem 3:



		Air Cargo Problem					
		Problem			**3**		
			Expansions	**Goal Tests**	**New nodes**	**Plan Length**	**P3**
uninformed	non-heuristic	1. breadth_first_search	14663	18098	129631	12	131.5598
		2. breadth_first_tree_search					⬛
		3. depth_first_graph_search	592	593	4927	571	3.5480
		4. depth_limited_search					⬛
		5. uniform_cost_search	18223	18225	159618	12	70.4130
o		6. recursive_best_first_search h_1					⬛
informed	heuristic	7. greedy_best_first_graph_search h_1	5578	5580	49150	22	20.5011
		8. astar_search h_1	18223	18225	159618	12	59.6950
		9. astar_search h_ignore_preconditions	5040	5042	44944	12	19.0678
		10. astar_search h_pg_levelsum	318	320	2934	12	289.4258

For problem 3, best execution time (3,548 seconds) uses the non-heuristic **Depth First graph search, DFS**, even better than the execution time (5,373 seconds) of **A* search with "ignore preconditions" heuristic**. Our DFS finds

the goal with a plan length of 571, expands 592 nodes and creates 4927 nodes. In this case, A* finds the goal with a plan length of 12 (more optimal than DFS), expands 5040 nodes and creates 44944 new nodes.
White squares rows (Breadth First Tree Search, Depth Limited Search and Recursive Best First Search h_1) are algorithms that were skipped for Problem 3 because it took longer than 10 minutes to run.
Taking into account number of expansions and new nodes, the one that expands less and creates fewer nodes is **A* with "levelsum" heuristic**, although this option is the one that takes more time to reach the goal, because the implementation for this heuristic may be not optimal.
Based on time execution, the optimal solution for **Problem 3** should be Depth First graph search, DFS, but its length path is 571 steps. It is not an optimal plan at all.

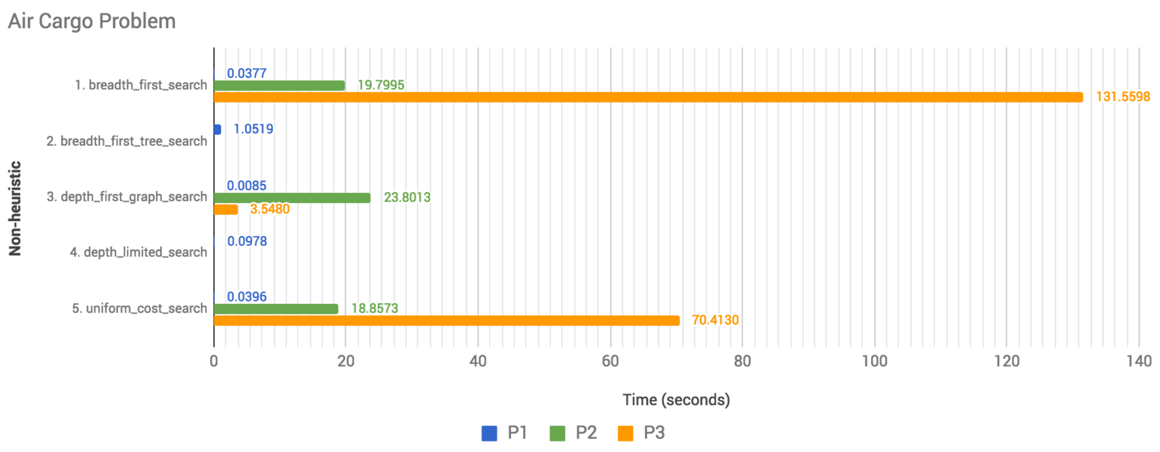
```
Solving Air Cargo Problem 3 using depth_first_graph_search...

Expansions   Goal Tests   New Nodes
   592          593          4927

Plan length: 571  Time elapsed in seconds: 4.535606079996796
Fly(P1, SFO, ORD)
Fly(P2, JFK, ORD)
Fly(P1, ORD, ATL)
Fly(P2, ORD, ATL)
Fly(P1, ATL, JFK)
Fly(P2, ATL, SFO)
Load(C1, P2, SFO)
Fly(P2, SFO, ORD)
…
Fly(P1, ORD, ATL)
Fly(P2, ATL, JFK)
Fly(P1, ATL, JFK)
Unload(C1, P1, JFK)
```

In terms of finding the **optimal plan** length in less time, the best is **A* search with "ignore preconditions"**.

Optimal plan:

```
Load(C2, P2, JFK)
Fly(P2, JFK, ORD)
Load(C4, P2, ORD)
Fly(P2, ORD, SFO)
Unload(C4, 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)
Unload(C2, P2, SFO)
```

2 - Non-heuristic metrics:

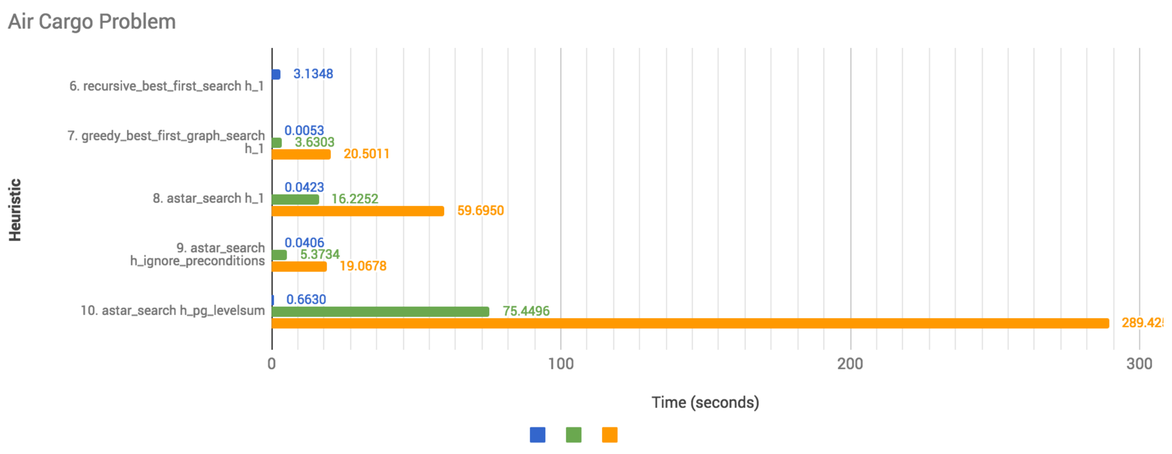


For the problems 1 and 2, **Breadth first search (BFS) is an acceptable choice because is complete, but not the best neither the optimal.**
Its downside is memory usage, if the problem's branching factor is high. As we can see in problem 3, using BFS is the worst choice. This algorithm must search following a strategy of expanding nodes of the same level first. Using **Depth First Graph search (DFS)** for the three problems in average is the best choice regarding time execution (in case you can't apply different algorithms for different problem definitions) but it is not optimal (for problem 3 it is a 571 path length! :S). Particularly, is the best choice for problem 1 and 3, and although it is the slower choice for problem 2, it will find the goal in less than 24 seconds. **Uniform cost search (UCS), complete and optimal,** is the best option for problem 2, and a good one for problem 1, very close to the results of BFS. In problem 3, it took 70 seconds to reach the goal, but it is still better than using BFS for the problem with branching factor.

We cannot compare Breadth First and Depth Limited in the three problems because were skipped for Problem 2 and 3.

To sum up, taking into account the 3 problems, time taken to reach the goal state by DFS in comparison to BFS and UCS is the lowest: **DFS < UCS < BFS**. Regarding reaching an optimal plan in less time, the best is **UCS**.

3 - Heuristic metrics:



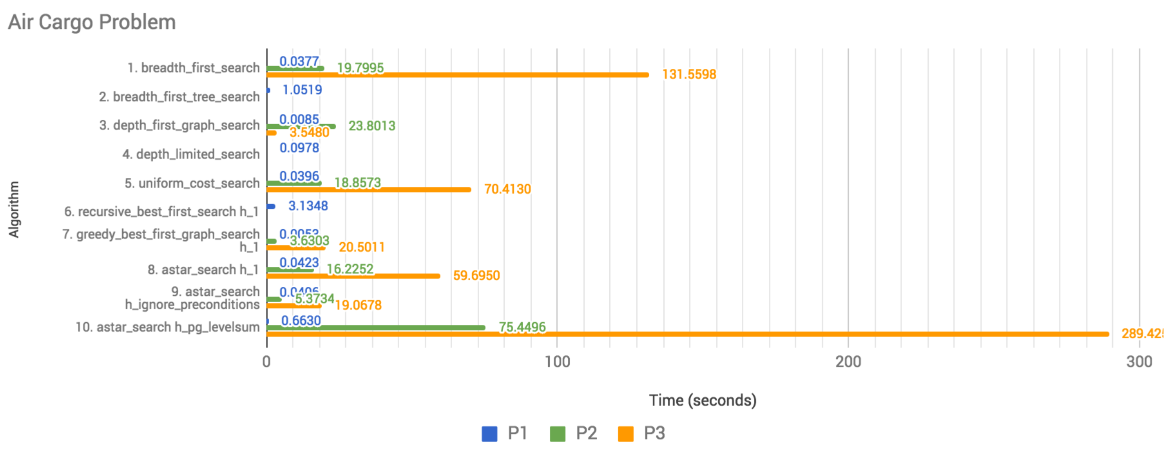
For the problems 1 and 2, **Greedy best first search is the best and optimal choice in terms of time execution**. In problem 3 is very close to the optimal solution given by A* search with "ignore preconditions", but for the three problems in average (in case you can't apply different algorithms for different problem definitions). Very close in terms of time execution, using **A* search with "ignore preconditions**". This solution finds an optimal plan, so for the three problems in average is the best choice in we want to find the optimal path in less time. Particularly, is the best choice for problem 3. The heuristic **A* with "levelsum"** is a good option for problem 1, but in problem 2 and 3 time execution gets worst. Maybe for a non-optimal implementation of the "levels" heuristic. Note that if we look at expansion and new nodes, this algorithm expands less and creates fewer nodes than the rest of the algorithms.

Our goal with both heuristics is to define a relaxed problem that is easier to solve and gives an admissible heuristic [AIMA 3rd edition, 10.2.3]. The heuristic "ignore preconditions" drops all preconditions from actions, so the number of steps to solve the relaxed problem is the number of unsatisfied goals. This is an instance of the set-covered problem that is NP-hard. Fortunately, the Greedy algorithm is guaranteed to return a set covering, but loses the guarantee of admissibility. In "levelsum", we use sub-goal independence, computing the cost of solving each subgoal in isolation, and combining the costs. It is s a reasonable estimate but not admissible, but works well in practice for problems that are largely decomposable. Only is admissible if the goals are independent, so the sum of the costs is admissible.

We cannot compare Recursive Best First Search in the three problems because were skipped for Problem 2 and 3.

To sum up, taking into account the 3 problems, time taken to reach the goal state by Greedy in comparison to A* "ignore preconditions" and A* "levelsum" is the lowest and very close to A* "ignore preconditions": **Greedy <= A* "ignore preconditions" < A* "levelsum".** Regarding reaching an optimal plan in less time, the best is **A* "ignore preconditions"** because Greedy takes more steps to reach the goal (for problem 3 22 steps instead of 12) and A* "levelsum" that expands less nodes and creates few new nodes but takes so long to reach the goal (for problem 3, 290 seconds instead of the 20 seconds if you use "ignore preconditions").

4 - Best heuristics (overall)



Time execution:
In terms of time execution, for problem 1 and 2, the best option is heuristic (Greedy), better than the best non-heuristic option (BFS or UCS). In case problem 3 the best option is non-heuristic (DFS) although its plan length is 571 nodes, but takes less time to execute than the best heuristic option (A* or Greedy). Why? Problem 3 has a large definition than Problem 1 and 2, so it takes longer to create the graph planning and the best option is a strategy that goes depth into the graph although it is not the best finding the optimal plan.

Optimal plan:
In terms of finding an optimal plan in less time, for all problems beat heuristic searches: **A* "ignore preconditions"** the best for problem 2 and 3 and in problem 1 is a good choice too. Taking into account number of expansions and new nodes, the one that expands less and creates fewer nodes is **A* with "levelsum" heuristic**, although this option is the one that takes more time to reach the goal, because the implementation for this heuristic may be not optimal.