Heuristic analysis

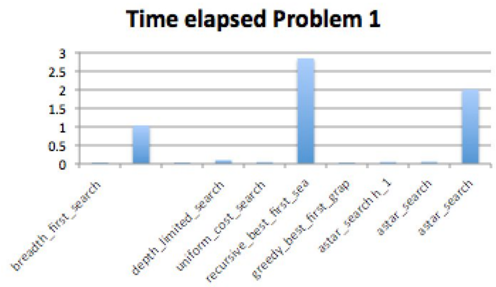
Tests results are presented below. Some rows are left incomplete as tests took too long to run.

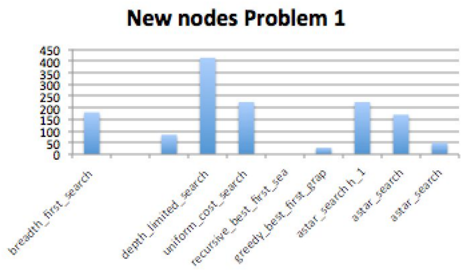
Problem 1

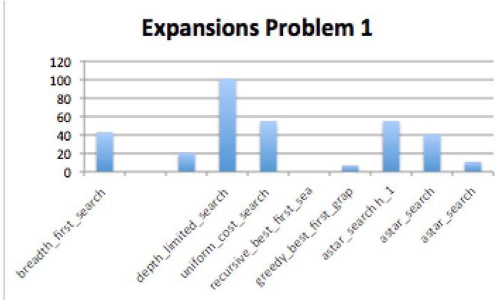
Results

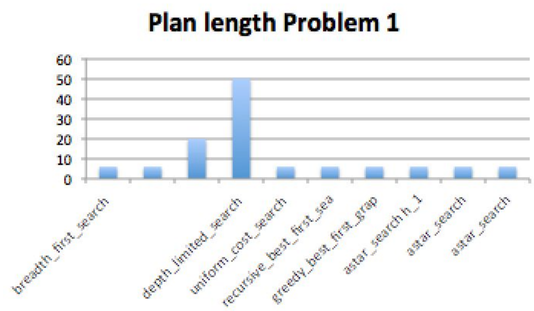
Algorithm	Expansions	Goal tests	New nodes	Plan length	Time elapsed
breadth_first_search	43	56	180	6	0.034227725
breadth_first_tree_search	1458	1459	5960	6	1.03013512
depth_first_graph_search	21	22	84	20	0.015699345
depth_limited_search	101	271	414	50	0.101213609
uniform_cost_search	55	57	224	6	0.050646812
recursive_best_first_search h_1	4229	4230	17023	6	2.834844509
greedy_best_first_graph_search h_1	7	9	28	6	0.007298584
astar_search h_1	55	57	224	6	0.05323923
astar_search h_ignore_preconditions	41	43	170	6	0.060153324
astar_search h_pg_levelsum	11	13	50	6	1.999543108

Graphs









Optimal sequence

Given:

```
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))
```

An optimal sequence is (length 6):

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

Analysis

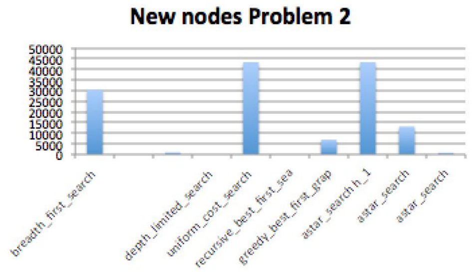
Problem 1 has a search space of 2^{12} states. The search space can be calculated as 2 to the power of the number of fluents which is given by: the amount of planes by the amount of airports plus the amount of cargos by the amount of planes plus the amount of cargos by the amount of airports.
It is possible to search through the entire search space in a small amount of time. Here greedy_best_first_graph_search h_1 is the optimal algorithm as it expands less nodes, perform less goal tests and takes less time to compute. It is not expected to behave as well as the search space increases in the following problems. In those cases, an heuristic is expected to outperform this algorithm. Most algorithms achieve the goal with an plan length of 6.

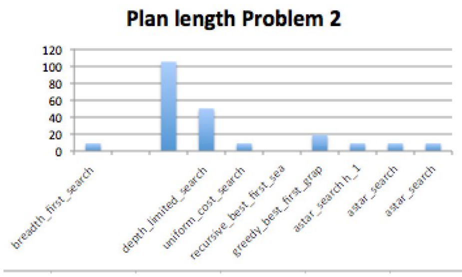
Problem 2

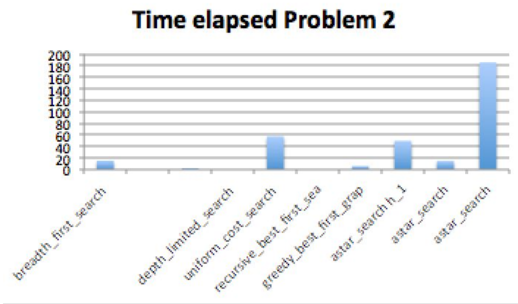
Results

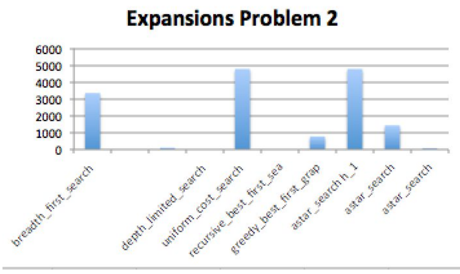
Algorithm	Expansions	Goal tests	New nodes	Plan length	Time elapsed
breadth_first_search	3346	4612	30534	9	15.2440457
breadth_first_tree_search					
depth_first_graph_search	107	108	959	105	0.338394214
depth_limited_search	213491	1967093	1967471	50	942.3392241
uniform_cost_search	4778	4780	43379	9	57.44520001
recursive_best_first_search h_1					
greedy_best_first_graph_search h_1	774	776	6938	19	6.021835573
astar_search h_1	4778	4780	43379	9	49.68677234
astar_search h_ignore_preconditions	1432	1434	13130	9	14.90964134
astar_search h_pg_levelsum	82	84	801	9	185.6548508

Graphs









Note: Results in red are not plotted as they are too big to use the same axis.

Optimal sequence

Given:

```
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))
```

An optimal sequence is (length 9):

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

```
Fly(P2, JFK, SFO)
Unload(C2, P2, SFO)
```

Analysis

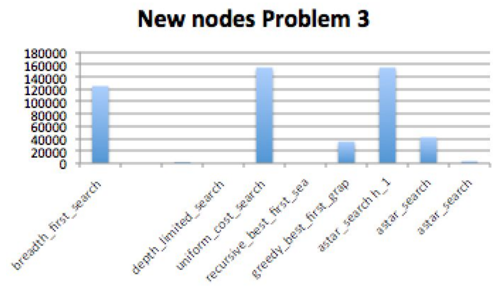
Now the space search has increased to 2^{27} states (3 airports, 3 planes and 3 cargos). Here depth_first_graph_search takes significantly less time to compute than all other algorithms but achieves the worst plan length.
Astar_search h_ignore_preconditions is the winner algorithm as it achieves an optimal plan length of 9 with minimum time. Astar_search h_pg_levelsum expands less nodes and performs less goal tests it takes over 12 times more time to compute. Breadth_first_search had similar results than in plan length and time but it expanded more nodes and perform more goal tests and therefore it is more expensive to compute.
It can be concluded that the search space is still small enough for performing a search of the entire space but the benefits of using heuristics can already be observed.

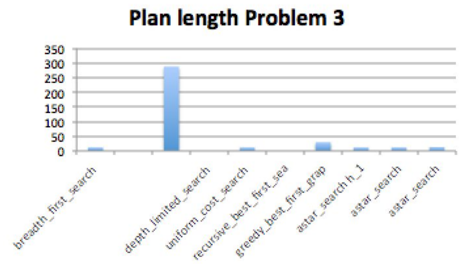
Problem 3

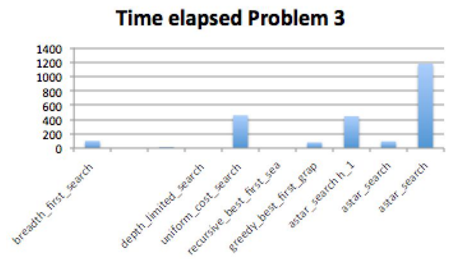
Results

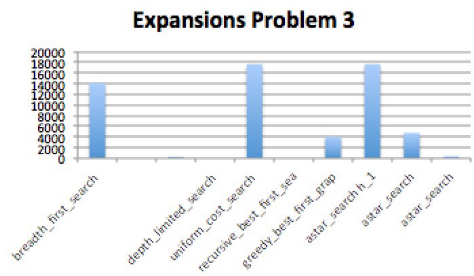
Algorithm	Expansions	Goal tests	New nodes	Plan length	Time elapsed
breadth_first_search	14120	17673	124926	12	101.6278573
breadth_first_tree_search					
depth_first_graph_search	292	293	2388	288	1.221368979
depth_limited_search					
uniform_cost_search	17653	17655	154877	12	458.7988739
recursive_best_first_search h_1					
greedy_best_first_graph_search h_1	4002	4004	35036	30	80.65345561
astar_search h_1	17653	17655	154877	12	446.7233794
astar_search h_ignore_preconditions	4757	4759	42172	12	94.73408909
astar_search h_pg_levelsum	378	380	3461	13	1179.361333

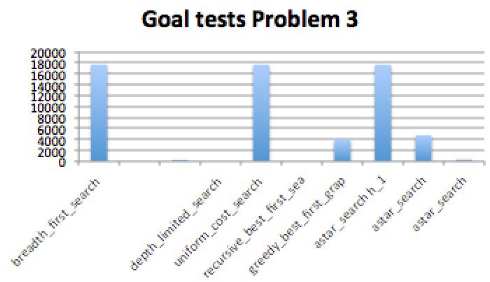
Graphs











Optimal sequence

Given:

```
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))
```

An optimal sequence is (length 12):

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

Analysis

Now the space search has increased to 2^{32} states (4 airports, 2 planes and 4 cargos). Again Astar_search h_ignore_preconditions is the winner as it achieves an optimal plan length of 12 in the minimum amount of time. As the search space increases, algorithms based on heuristic search tend to be better.

Evolution with problem complexity

Based on the following graphs, it can be concluded that A* algorithms, i.e. heuristic based algorithms, tend to perform better as the search space increases. As the space increase is exponentially related to the number of fluents, heuristics provide a powerful planning tool.

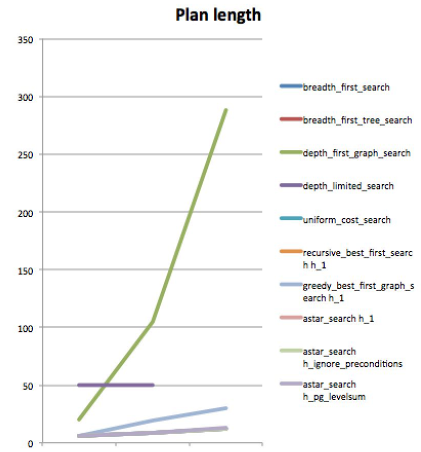
As breath first search expands all nodes at the frontier of search graph before going deeper, as the search space grows larger, it will take longer to find a path to the goal (always taking the shorter path available).

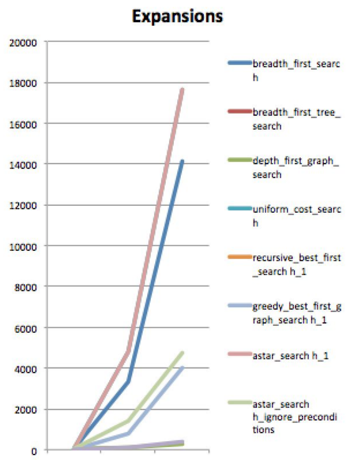
Depth first search graph will expand the node before exploring other nodes in the frontier. This will result in inefficiencies and too much time consumption as the search tree grows.

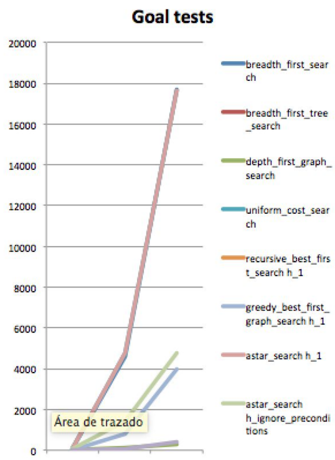
Uniform cost search is guaranteed to find the path with the cheapest total cost, but it will continue expanding nodes after reaching a goal so it will take longer than previous algorithms that stop searching when reaching a goal (and in the case of breath first search also exploring other nodes in the frontier).

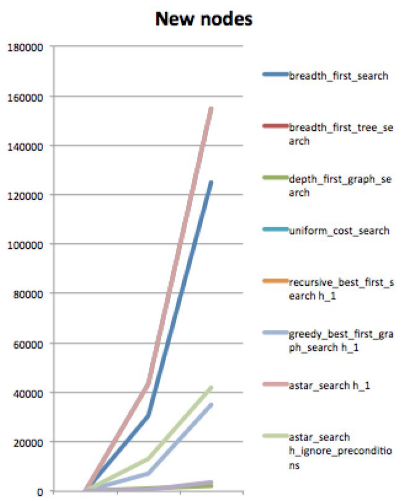
Greedy is a fast algorithm but does not guarantee to find the best solution. Its speed was good at problem 1 where it did find the best solution but not in problems 2 & 3 where the plan length was more than double of the optimal plan length.

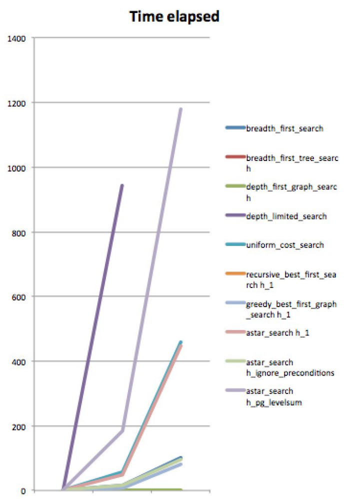
A star was the best algorithm in problems 2 & 3 as the heuristic was optimistic or admissible. Although h_ignore_preconditions expanded more nodes and made more goal tests, it always found the optimal plan and was faster than h_pg_levelsum that turned out to be too expensive to compute.











References

- Heuristics for state-space search, Chapter 11 - Planning of "Artificial Intelligence: A Modern Approach" of Peter Norvig, Stuart J. Russell
- Udacity's AIND course, Lesson 7 - Search.
- Udacity's AIND course, Lesson 11 - Planning.