Heuristic analysis

By Søren Pedersen, December 10 2017

Abstract

This document will investigate how the different search algorithms perform on problem 1, 2 and 3. First section will describe the performance of non-heuristic searches and the second will describe A* with heuristics.
The results will be analysed and concluded on at the end of the document. It is a rather long document, but most of it is diagrams.

Overall performance for the algorithms is described in the tables below.

```
P1 = Problem 1, P2 = Problem 2, P3 = Problem 3

B1 = Breadth first search
B2 = Breadth first tree search
D1 = Depth first graph search
D2 = Depth limited search
U   = Uniform cost search
R   = Recursive best first search
G   = Greedy best first graph search
A1 = A star search h1
A2 = A star search h ignore preconditions
A3 = A star search h pg level sum
```

Node expansions

	B1	B2	D1	D2	U	R	G	A1	A2	A3
P1	43	1458	21	101	55	4229	7	55	41	11
P2	3343	INF	624	INF	4852	INF	990	4852	1450	86
P3	14663	INF	408	INF	18235	INF	5614	18235	5040	325

Goal tests

	B1	B2	D1	D2	U	R	G	A1	A2	A3
P1	56	1459	22	271	57	4230	9	57	43	13
P2	4609	INF	625	INF	4854	INF	992	4854	1452	88
P3	18098	INF	409	INF	18237	INF	5616	18237	5042	327

Time

	B1	B2	D1	D2	U	R	G	A1	A2	A3
P1	0.03	0.82	0.01	0.07	0.03	2.41	0.009	0.04	0.04	0.8
P2	11.6	INF	2.92	INF	9.88	INF	2.05	9.92	3.68	129
P3	83.9	INF	1.6	INF	43.6	INF	12,89	43.7	14.07	680

Plan length

	B1	B2	D1	D2	U	R	G	A1	A2	A3
P1	6	6	20	50	6	6	6	6	6	6
P2	9	INF	619	INF	9	INF	21	9	9	9
P3	12	INF	392	INF	12	INF	22	12	12	12

Non-heuristic search analysis

The search algorithms that I have chosen to illustrate in this section is breadth first (B1), depth first (D1) and greedy best first (G). For each problem there is a deeper analysis of how they behave on these problems.

Problem 1 search analysis

Problem 1 is defined as

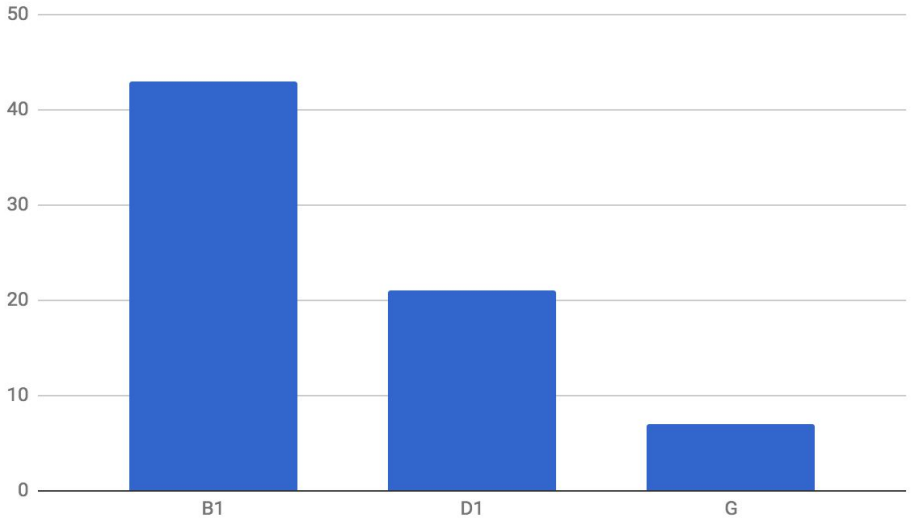
```
Init(At(C1, SFO) ∧ At(C2, JFK)
     ∧ At(P1, SFO) ∧ At(P2, JFK)
     ∧ Cargo(C1) ∧ Cargo(C2)
     ∧ Plane(P1) ∧ Plane(P2)
     ∧ Airport(JFK) ∧ Airport(SFO))
Goal(At(C1, JFK) ∧ At(C2, SFO))
```

The Optimal plan is:

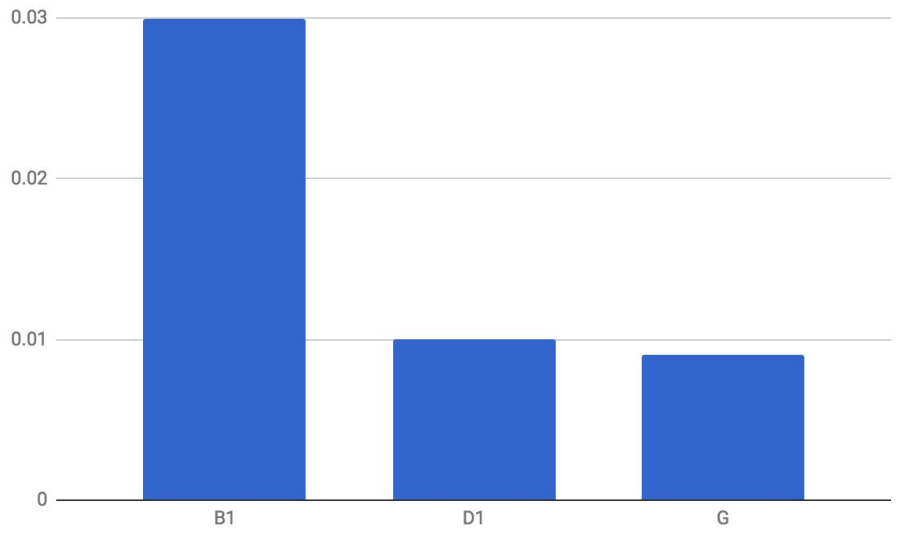
```
Load(C1, P1, SFO)
Load(C2, P2, JFK)
Fly(P2, JFK, SFO)
Unload(C2, P2, SFO)
Fly(P1, SFO, JFK)
Unload(C1, P1, JFK)
```

The diagrams below shows that G (Greedy best first graph search) is the most effective for this problem. It expands fewest nodes, spends less time and returns the optimal plan. D1 (depth first) returns a plan that is over 3 times larger than the optimal plan and is not the right algorithm in this case.

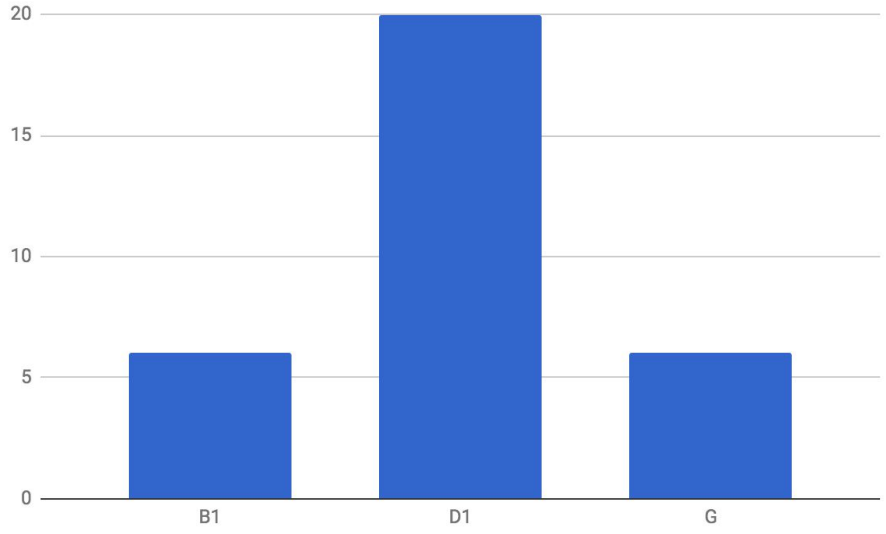
Nodes expanded:



Time elapsed:



Plan length:



Problem 2 search analysis

Problem 2 is defined as:

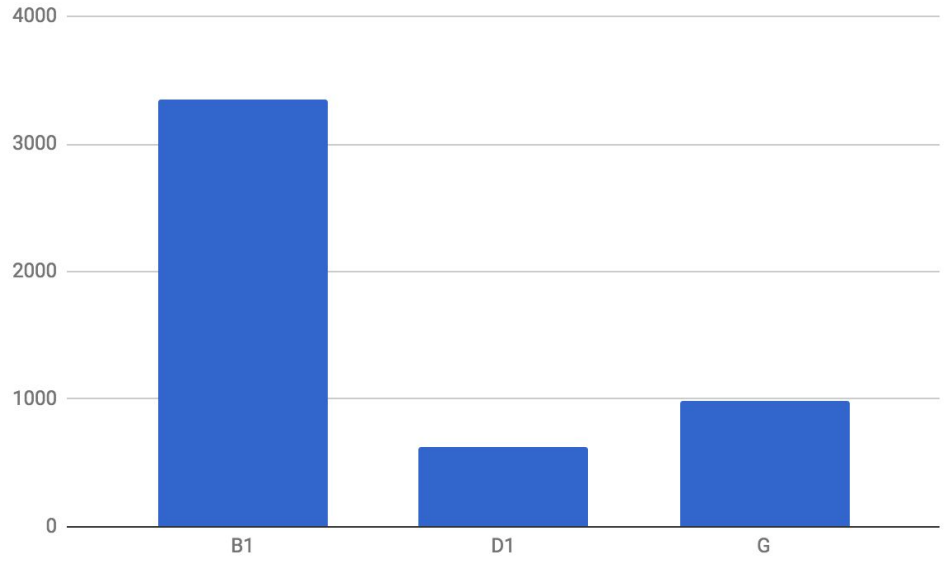
```
Init(At(C1, SFO) ∧ At(C2, JFK) ∧ At(C3, ATL)
     ∧ At(P1, SFO) ∧ At(P2, JFK) ∧ At(P3, ATL)
     ∧ Cargo(C1) ∧ Cargo(C2) ∧ Cargo(C3)
     ∧ Plane(P1) ∧ Plane(P2) ∧ Plane(P3)
     ∧ Airport(JFK) ∧ Airport(SFO) ∧ Airport(ATL))
Goal(At(C1, JFK) ∧ At(C2, SFO) ∧ At(C3, SFO))
```

The optimal plan is:

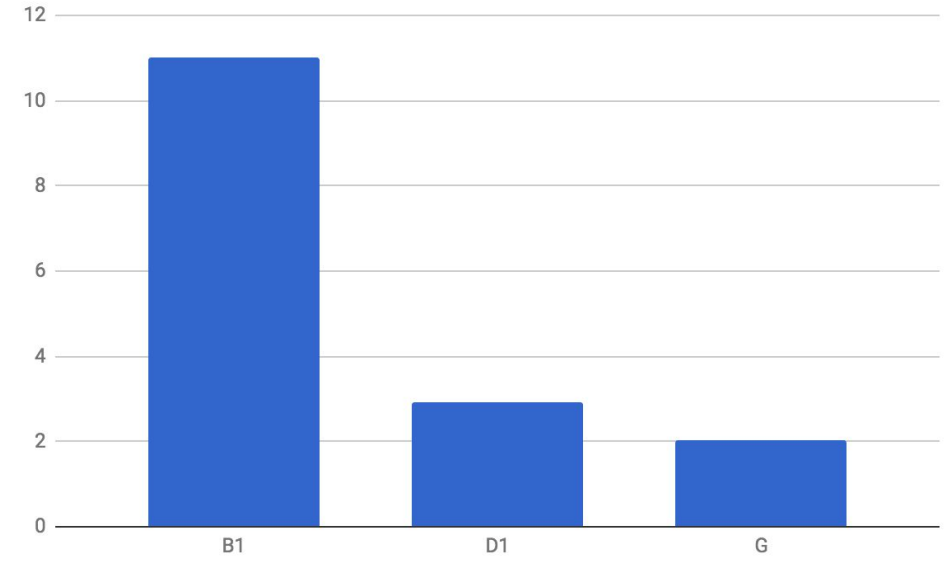
```
Load(C1, P1, SFO)
Load(C2, P2, JFK)
Load(C3, P3, ATL)
Fly(P2, JFK, SFO)
Unload(C2, P2, SFO)
Fly(P1, SFO, JFK)
Unload(C1, P1, JFK)
Fly(P3, ATL, SFO)
Unload(C3, P3, SFO)
```

The breadth first spends over 11 seconds on finding the optimal path. It also expands most nodes. Depth first expands fewest nodes but returns a plan with length of 621 which is way too much. Greed first expands 990 nodes which is less than breadth first. It is the fastest search but it does not return the optimal plan length. It returns 21 and the optimal plan is 9. This leaves us with a situation where we need to chose between correctness and speed. If we want the plan to be most correct using non-heuristic-search, we should select breadth first.

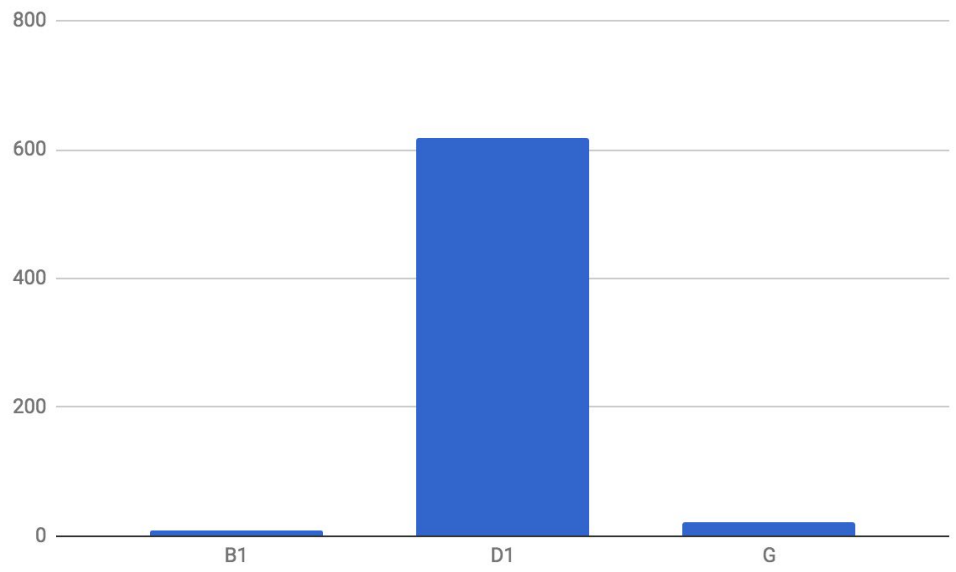
Nodes expanded:



Time elapsed:



Plan length:



Problem 3 search analysis

Problem 3 is defined as:

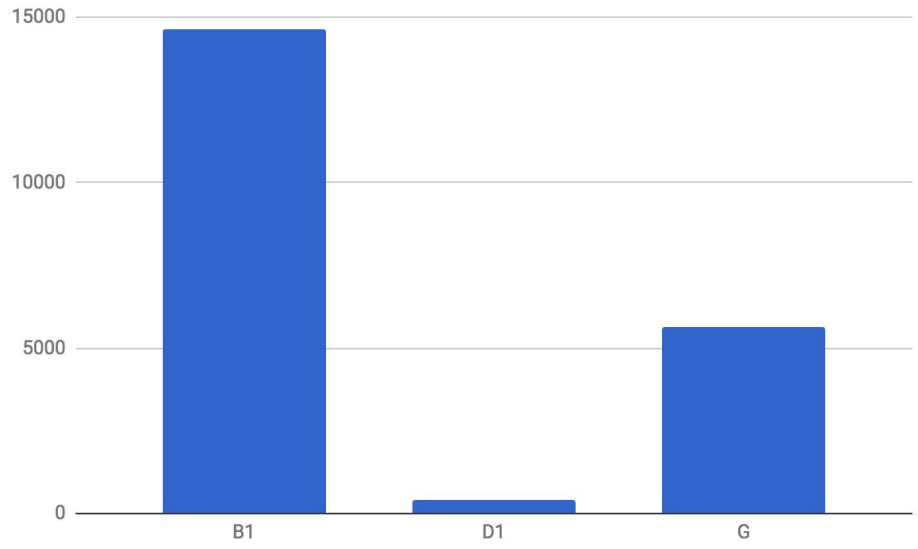
```
Init(At(C1, SFO) ∧ At(C2, JFK) ∧ At(C3, ATL) ∧ At(C4, ORD)
      ∧ At(P1, SFO) ∧ At(P2, JFK)
      ∧ Cargo(C1) ∧ Cargo(C2) ∧ Cargo(C3) ∧ Cargo(C4)
      ∧ Plane(P1) ∧ Plane(P2)
      ∧ Airport(JFK) ∧ Airport(SFO) ∧ Airport(ATL) ∧ Airport(ORD))
Goal(At(C1, JFK) ∧ At(C3, JFK) ∧ At(C2, SFO) ∧ At(C4, SFO))
```

The optimal plan is:

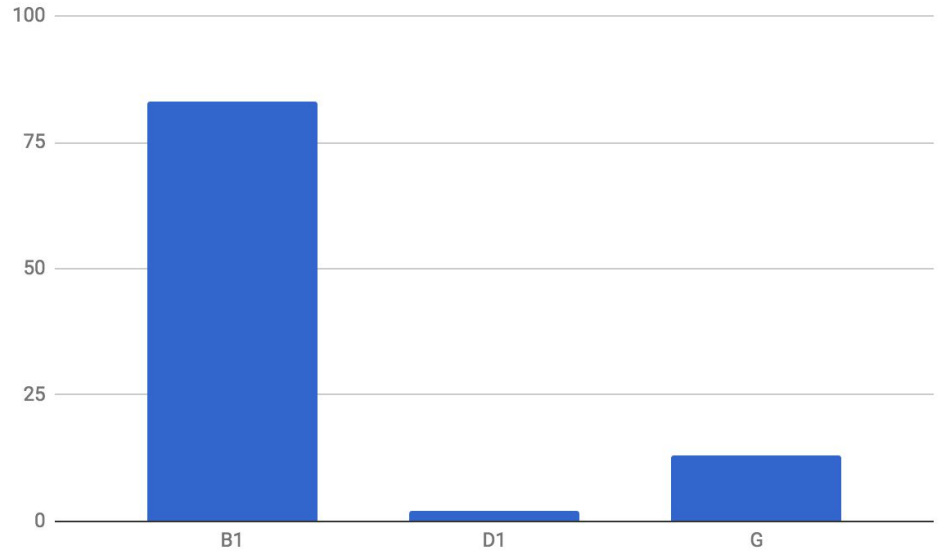
```
Load(C1, P1, SFO)
Load(C2, P2, JFK)
Fly(P2, JFK, ORD)
Load(C4, P2, ORD)
Fly(P1, SFO, ATL)
Load(C3, P1, ATL)
Fly(P1, ATL, JFK)
Unload(C1, P1, JFK)
Unload(C3, P1, JFK)
Fly(P2, ORD, SFO)
Unload(C2, P2, SFO)
Unload(C4, P2, SFO)
```

As with problem 2, this problem also gets the optimal plan length from breadth first. However, this is also the most time consuming search since it takes 83.94 seconds. Depth first is the fastest but it returns a plan length of 392. Greedy first returns a plan which is 22 elements long and it takes 12.89 seconds to execute.
If we only have these three algorithms to chose between, we would have to chose between speed and accuracy.

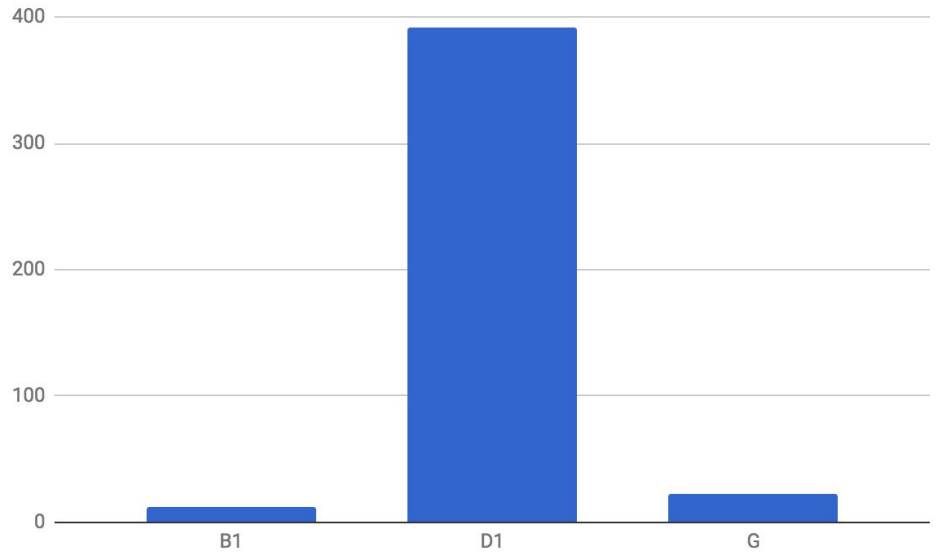
Nodes expanded:



Time elapsed:



Plan length:



Conclusion on non-heuristic searches

The selection of algorithms for these searches seems far from optimal. None of them seems very attractive and if we only had breadth first, depth first and greedy first search to chose between it would either be a very slow, but precise solution, or a fast, but not very reliable, solution. We have not discussed the uniform cost search algorithm. According to the results for each problem it returns the optimal plan in all cases. It is way slower than depth first but on problem 3 it is almost twice as fast. So if we had to stick with non-heuristic searches, the uniform cost search seems to have a fair balance between speed and accuracy.

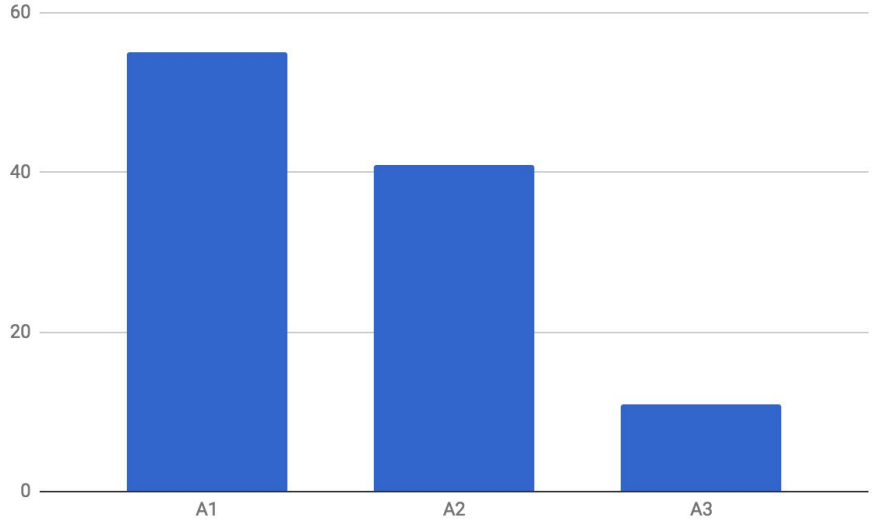
A * searches analysis

In this section we are going to discuss the A * search with different kinds of heuristics. The heuristics will be ignore preconditions and level sum. All A* searches returns the optimal path, so I will not put that in a histogram, only state the optimal length for each problem.

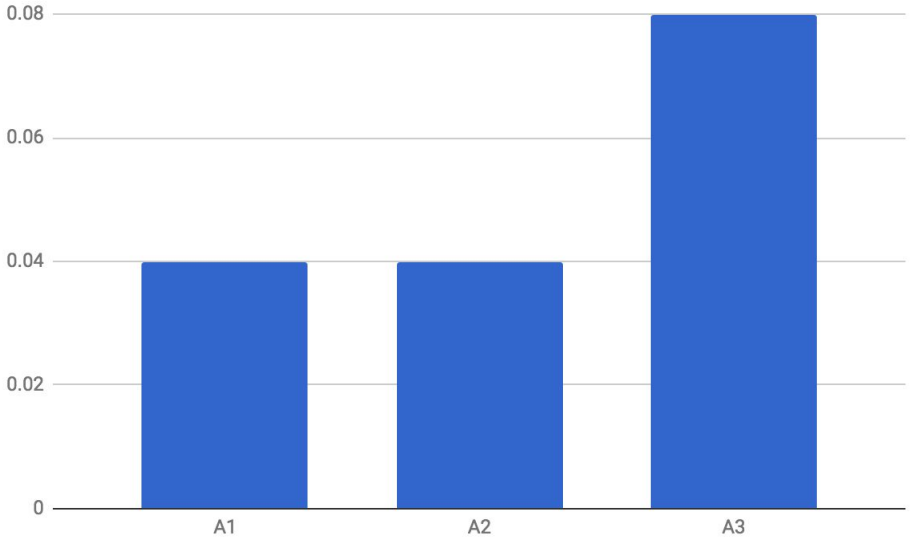
Problem 1

The optimal length of this problem is 6. All searches returns that. Since this is a real small problem, the only thing significant is the number of nodes expanded. The level sum (A3) expands only 11 where the basic A* expands 55. It takes 0.4 seconds longer for the level sum to return and this could be a problem in a system that requires fast planning.

Nodes expanded:



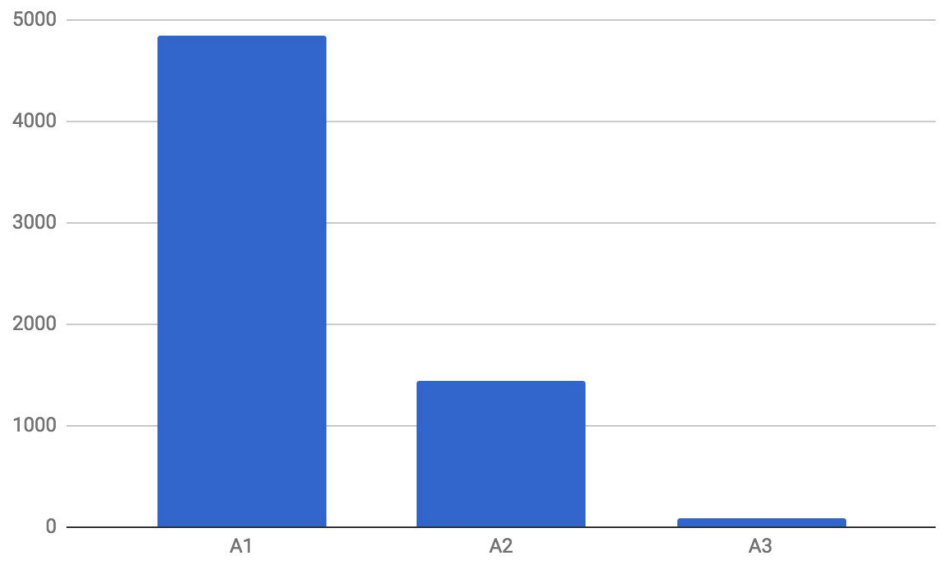
Time elapsed:



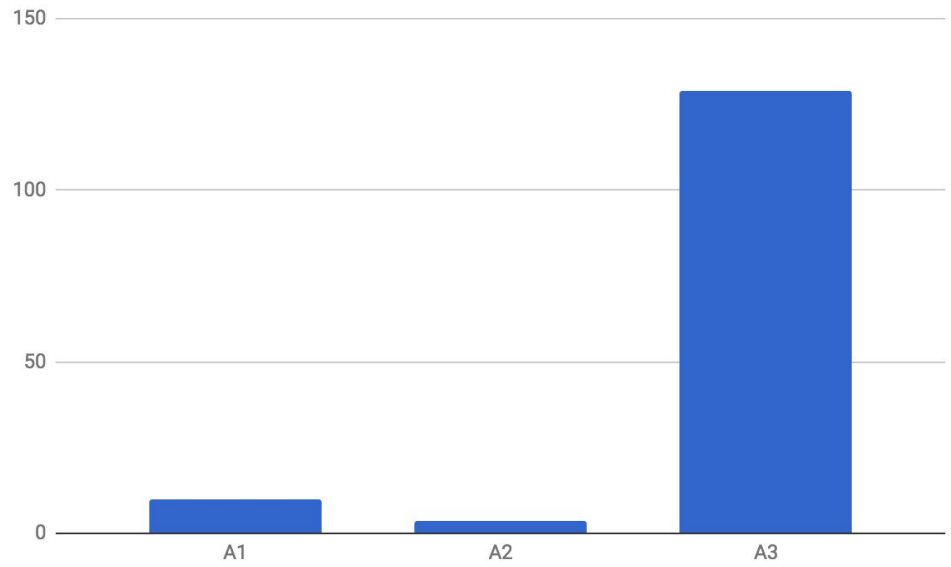
Problem 2

The optimal length for this problem is 9. The significance in these results is that the basic A* (A1) expands almost 5,000 nodes and the level sum (A3) expands 86 nodes. This comes with a cost, however, so that level sum spends 129 seconds. The fastest and still correct search is the ignore preconditions (A2) heuristic and this seems optimal on this problem.

Nodes expanded:



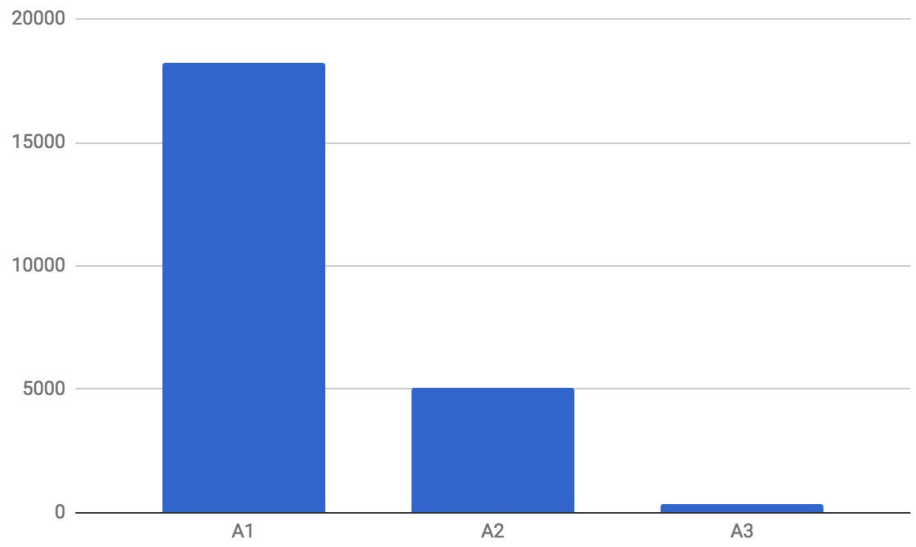
Time elapsed:



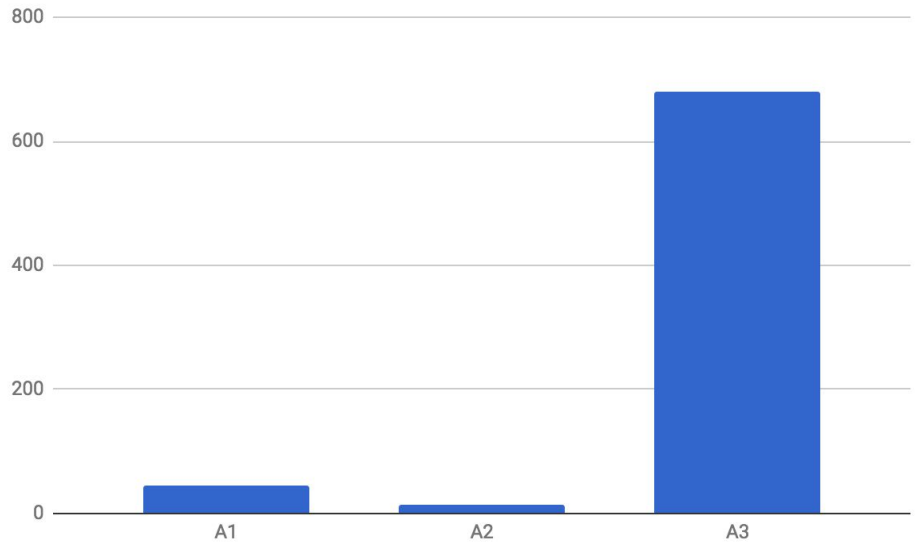
Problem 3

The optimal plan length is 12 for this problem. We are facing the same situation as with problem 2. Level sum expands fewest nodes but spends much longer time than the two other heuristics. Balancing the forces, the best selection is again the ignore preconditions heuristic (A2).

Nodes expanded:



Time elapsed:



Conclusion

The overall consideration is to get an algorithm that is both fast and precise. Looking only at speed, the depth first graph search (D1) performs best. That comes with a price though, and it returns plans that are far from optimal. Looking at precision we have more to chose from. Breadth first and A* searches seems to return the optimal plan for all problems. For breadth first search (B1) this is very time consuming and it spends 83 seconds on problem 3, which is way longer than A* search with ignore preconditions heuristic (A2).

If it was important to expand fewest nodes then A* search with level sum heuristic is the right choice. This, however, is very time consuming due to the level sum calculation that searches through all levels for each goal.

Considering all this it seems that the best solution is A* with ignore preconditions since it is both fast and precise.