**ANALYSIS**

1. What is the mean effective branching factor for each of the 6 heuristics?

The mean effective branching factor is the following for each of the 6 heuristics:

1: 1.027

2: 1.026

3: 1.026

4: 1.026

5: 1.026

6: 1.015

We realize that this is very small for a branching factor (especially for the 0 heuristic) but were unable to diagnose the problem.

1. How do the 5 heuristics vary in effectiveness?

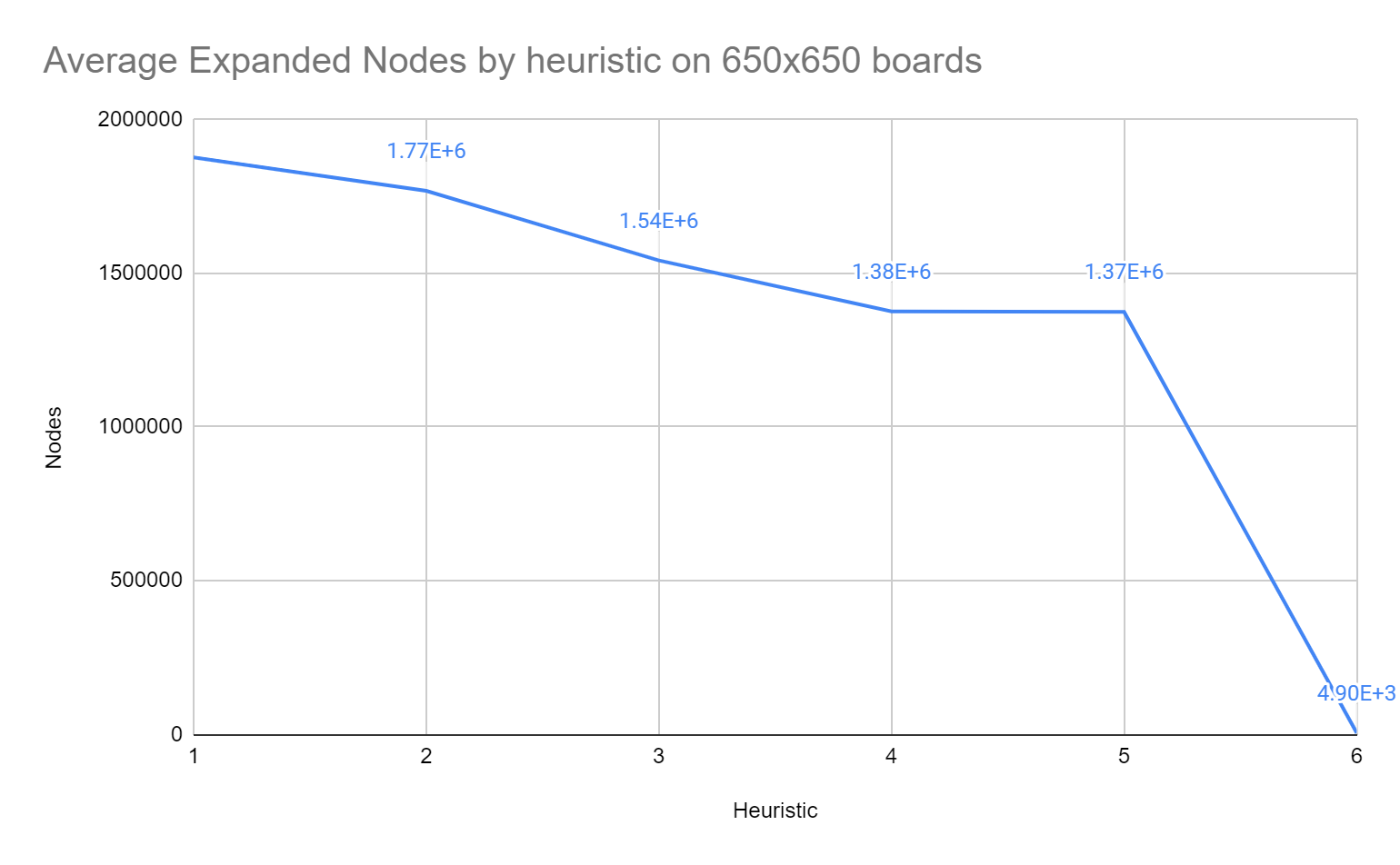
Heuristic One is usually the slowest as it always leads to the algorithm processing the most nodes. However, sometimes it is faster than Heuristics Two through Five because it requires almost no calculations.

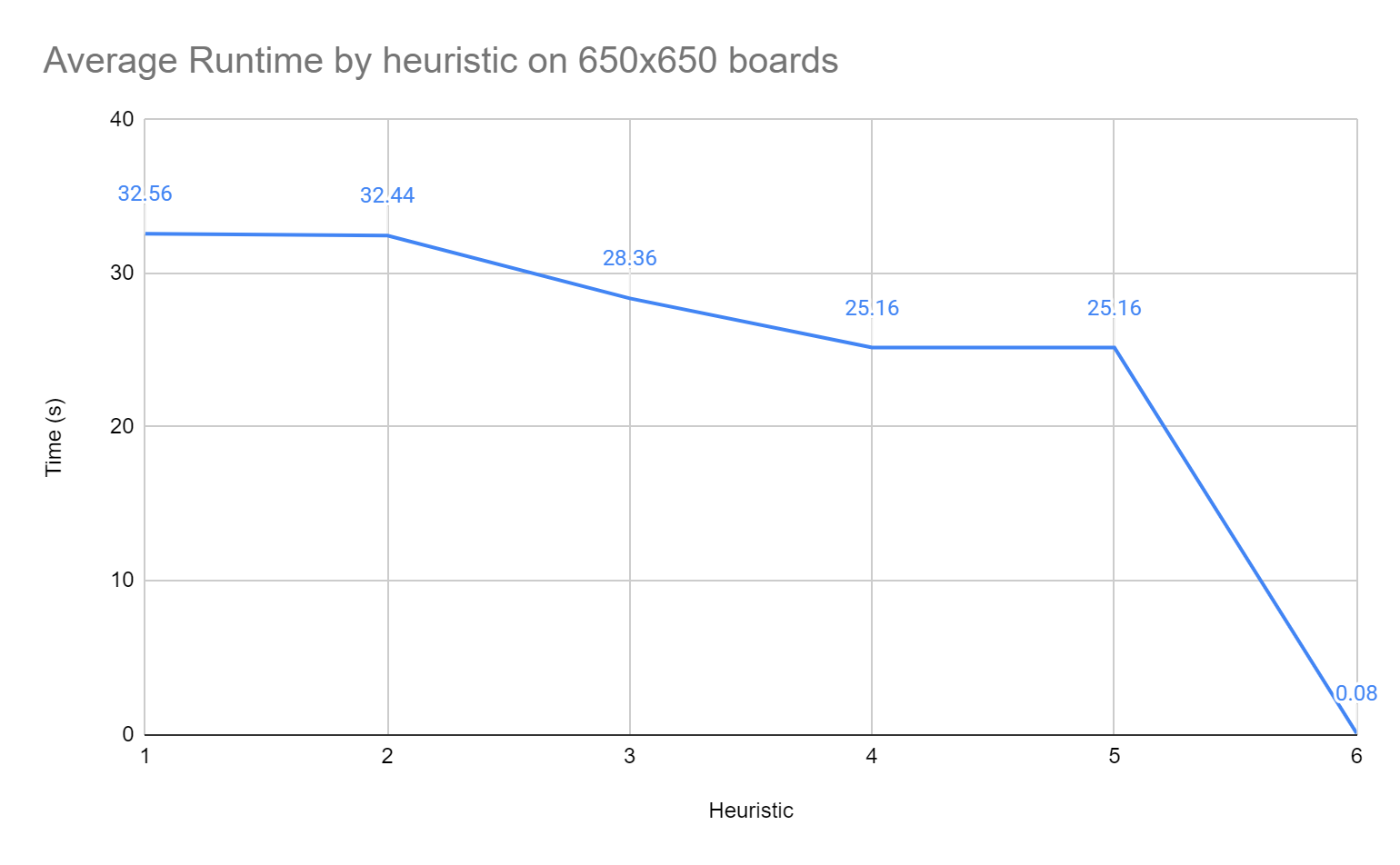
Heuristic Two through Four are respectively the second through fourth slowest and each lowers the number of nodes expanded by the algorithm by a noticeable amount (Between 2 and upwards of 30 percent)

Heuristic Five is marginally better than Heuristic Four in most cases and causes the algorithm to visit fewer nodes, but it does sometimes increase overall runtime because it involves more calculations than Heuristic Four.

Each of the first five heuristics find the cheapest path.

Heuristic Six runs much, much faster than any of the other algorithms and expands hundreds of times less nodes. However, it does not find the cheapest path. It runs very quickly, even for the largest board sizes.





1. How much gain is there to using any heuristic (#1 vs. #2)?

If the shortest path is desired, Heuristics Four or Five should be used. If node expansion is prioritized, Heuristic Five should be used to minimize the expanded nodes. Otherwise, the time cost of Heuristic Four can sometimes be less than that of Five. If a working path is desired in a short time, Heuristic Six can be used to deliver a working (but not best) result very quickly.

1. Is #5 noticeably more effective than the other heuristics?

Heuristic Five is marginally more effective than Four at limiting expanded nodes, but because of the higher heuristic calculation, its cost can sometimes take longer than Heuristic Four.

1. For heuristic #6: how does its solution quality compare with #5? Is it performing noticeably worse? How much more efficient is it?

Heuristic Six is no longer admissible so it will not necessarily find the most optimal answer. We see from the data that the cost of the path found by heuristic 6 is a bit larger than the rest (about 10%) but can find that path in much less time. Heuristic 6 needed a few thousand expanded nodes to find the solution as opposed to the millions required for the rest, increasing efficiency over 500x.

1. How large of a map can you solve in 30 seconds using Heuristic 5?

Using heuristic 5, the algorithm can solve a map that is, on average, 550 by 550.

(32.65 s + 17.33 s + 29.03 s + 27.71s + 49.24 s + 19.86 s = 175.92 s , 175.92/6 = 29.32 seconds)

1. How much memory is needed? Perform a “back of the envelope” calculation for how large a board you could solve with 16 GB of memory using each of the 6 heuristics (you can assume the board is square for this analysis).

Based on the queue size, we can assume the approximate amount of memory used (we will use about 50 bytes per node in the queue). These were for boards of size 650x650.

For heuristics 1-5:

We have max queue lengths of about 10000, the program would use about 0.5 MB.

To find how this scales, we can find the proportionality constant between 0.5MB and the space complexity of the algorithm O(b^d). This comes out to around 6e-10 to find that the max depth we achieve with 16GB memory to be just shy of 1200.

For heuristic 6: We have max queue length of 5333, which would use about 0.25MB.

Following the same process, we find the max depth to be about 1400

1. How long would it take to solve that board?

To solve the board above, with a path length of about 1200 for heuristics 1-5 would take about 300 hours while the last board would take about 1.5 hours.

1. How much memory would you need for a problem that requires 24 hours to solve with Heuristic 5? With heuristic 6?

With heuristic 5, given that there are 86400 seconds in 24 hours, and around 440 MB (peak) are spent on a search that takes 118 seconds, 24 hours (86400 seconds) would take around 23,170 MB which would be 23GB.

With heuristic 6, given that there are 86400 seconds in 24 hours, and around 4MB (peak) are spent on a search that takes 0.3 seconds, it would take around 6,480 MB which would be around 6.5 GB.

For heuristic 5:

44 sec -> 187 MB

47 sec -> 208 MB

93 sec -> 357MB

118 sec -> 437.MB

For heuristic 6:

0.3 seconds -> 4MB

Results for 10 test boards can be found in the accompanying spreadsheet.