MSc. Computer Science for Communication Networks

Advanced Software Engineering for Smart Devices

Performance Analysis 15-puzzle

Authors:
Oscar Guillen
Patricia Reinoso

Adviser:
Paul Gibson

February 9th, 2018

Contents

15-puzzle

0.1 Introduction

In the following report we have the performance analysis of the 15-puzzle solution with the use of different fitness functions used together with the algorithm A* to reach the solution of the puzzle. To develop the performance analysis, a script has been created to reproduce simulations of different puzzles and thus be able to calculate certain values to perform their respective analysis. In the script random movements are recreated on a solved board in order to create a new board and try to solve it with the algorithm created using each of the fitness functions.

Our solution is performed using the A* algorithm with a fitness function to calculate a cost of each puzzle and be able to find the most efficient way to achieve the solution. The algorithm is based on a tree that is generated step by step and that creates states from the possible steps that can be taken and decide which state is the most correct to take based on its cost.

In general, the analysis is performed to compare the performance of three fitness functions based on three different heuristics used in the A * algorithm:

- Linear Distance Heuristic
- Disorder Heuristic
- Manhattan Distance Heuristic

For each puzzle, the solution based on each of these heuristics is calculated. First, a comparative analysis is made based on the length of the solution obtained. Second, another analysis based on the maximum depth reached when generating states in the algorithm. Third, a comparison is made based on the analysis of the duration to obtain each solution and finally it is done based on the number of states created when looking for the solution.

Finally, we reach a conclusion in which we define what would be the most optimal algorithm to use.

0.2 Generation Script

In the script, 70 puzzles are randomly generated for each fitness function. The generation is made by the generation of white space movements in the solved puzzle. For the purposes of performance analysis, puzzles are created from a specific number of movements, that is, the index of the puzzle is the number of movements made on the puzzle solved to generate it.

Then, the puzzles have been generated in the following way:

- Generated Puzzle Nro. 1 with 1 move. One per fitness function.
- Generated Puzzle Nro. 2 with 2 moves. One per fitness function.
- Generated Puzzle Nro. 3 with 3 moves. One per fitness function.
- ...
- Generated Puzzle Nro. 70 with 70 moves. One per fitness function

For each generated puzzle solved with each fitness function, the values used for the analysis were saved or written in a file.

0.3 Solution Length Analysis

The fitness value represents the cost of the initial puzzle to be solved and the solution length indicates the number of movements to arrive at the solution of the board.

We can see in the figures for Manhattan distance and disorder distance that the dispersion goes according to the cost of each board. That is, the higher the cost, the board requires more moves to be resolved. However, it is clearer for the Manhattan distance, because for the disorder the dispersion is more expanded. On the other hand, for linear distance the points seem to have a similar behavior but they are very dispersed in comparison.

Despite the scattering of the points, we can note that for Manhattan distance it provides shorter solutions for problems. On the other hand, disorder and linear distance provide longer solutions to problems.

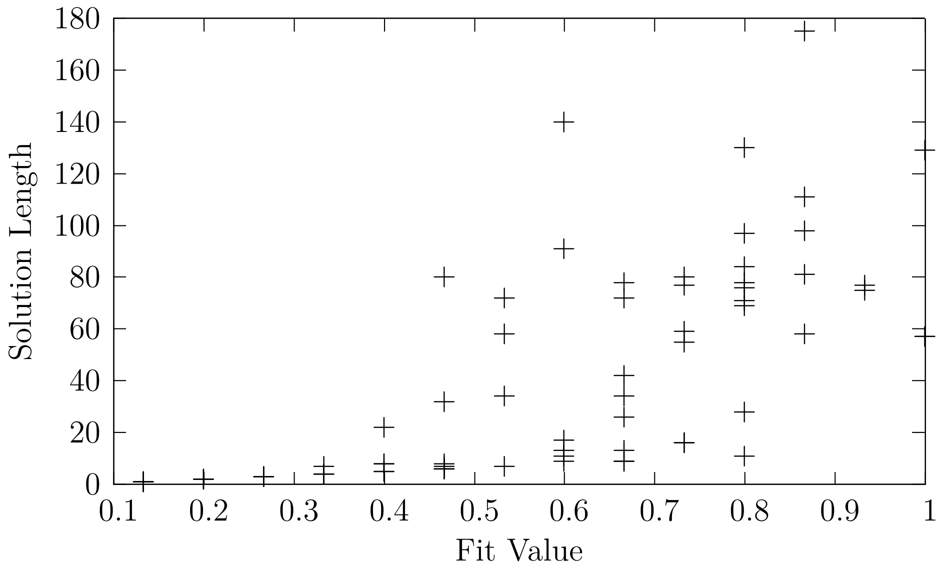

Figure 1: Solution length of disorder heuristic

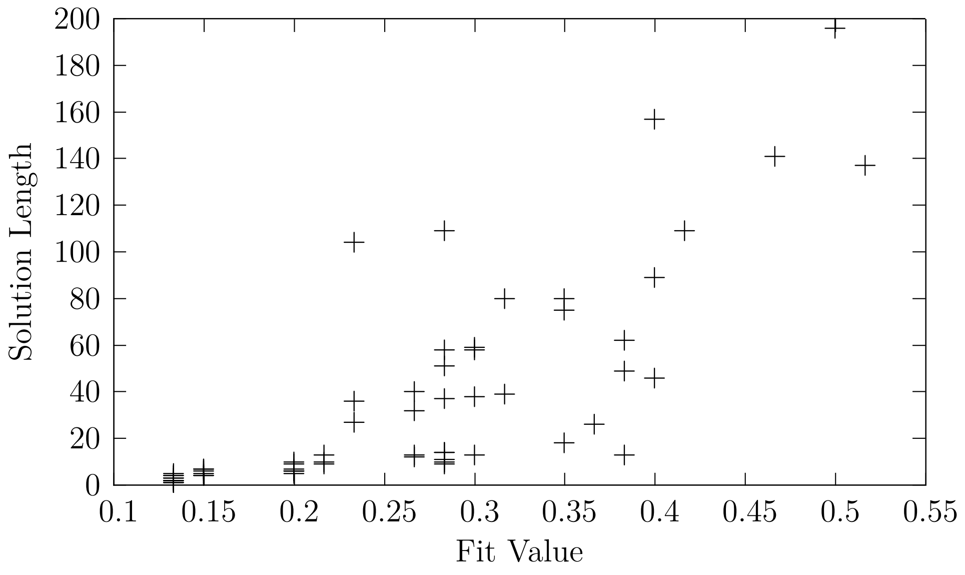

Figure 2: Solution length of linear distance heuristic

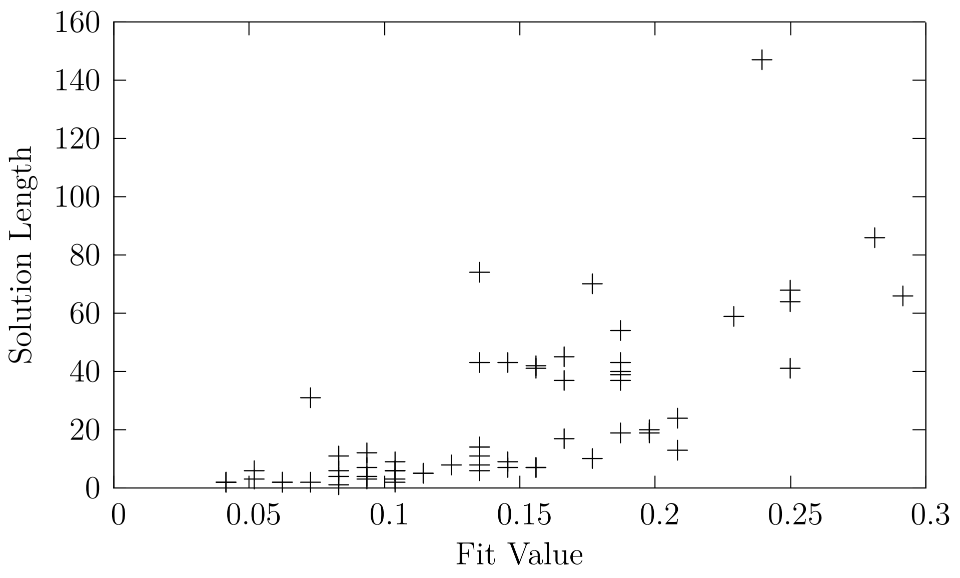

Figure 3: Solution length of Manhattan distance heuristic

0.4 Solution Depth Analysis

The fit value represents the cost of the puzzle generated to be solved and the max depth inside the tree represents the maximum depth reached when looking for solutions within the tree that is being generated.

With respect to the size of the solution, which can be translated as the complexity of a puzzle, we can see that it grows almost linearly. This means, the longer the solution, the more it entered the tree to look for solutions. Even almost the same for the 3 functions. However, it can be seen that Manhattan distance does not go so deeply with respect to other functions. In fact depth over 80 approximate can be a cap, beyond a couple of solutions that went beyond this limit. While for the others this limit is intermediate. Linear distance remains as the most depth reaches in more cases.

For figures 4,6 and 8 we can see that the points are very scattered. However, we can conclude with more clarity that Manhattan distance does not go so deep in the tree to reach its solution. On the other hand, the points are very dispersed for all but it can be observed that for all when the cost is very close to 0 then it does not go so deep in the tree.

The deeper it is generated in the tree, the more states are generated then the more memory is needed.

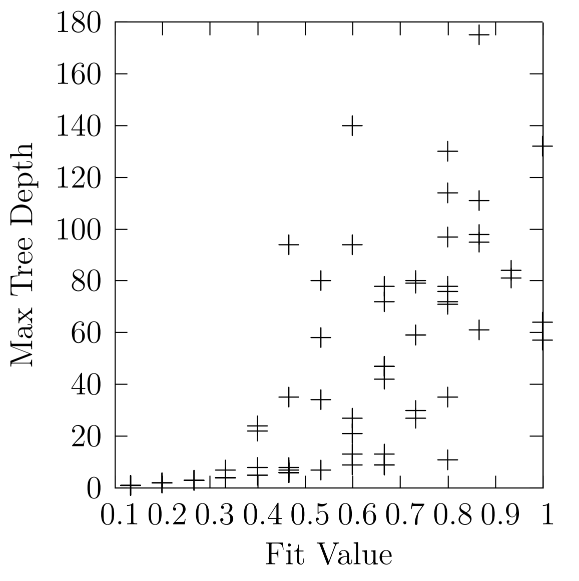

Figure 4: Max Depth for disorder

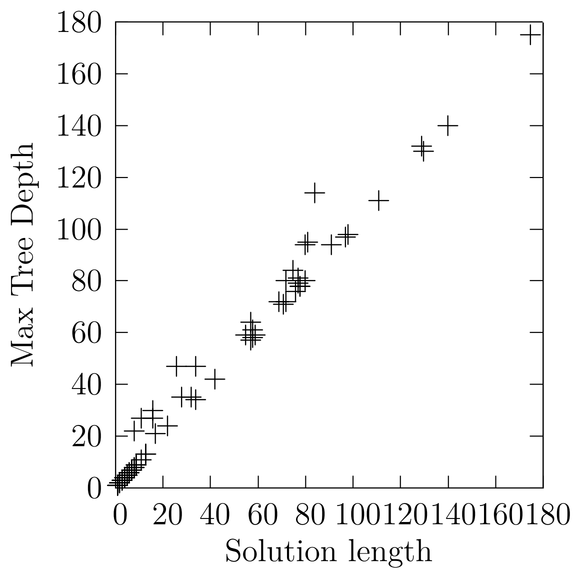

Figure 5: Max Depth for disorder

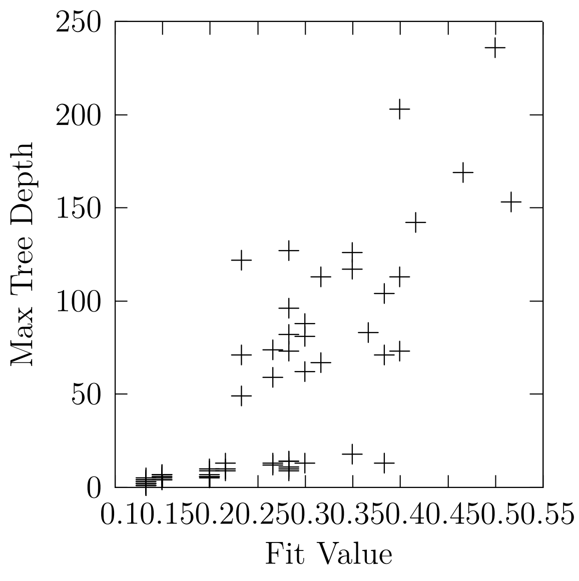

Figure 6: Max Depth for linear distance

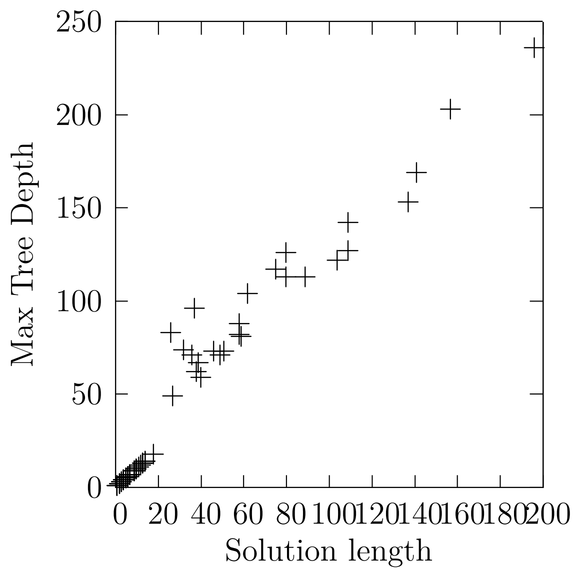

Figure 7: Max Depth for linear distance

0.5 Time Complexity Analysis

The time complexity means the time of duration in getting the solution of the puzzle.

It can be seen in figures 14 and 15 that the Manhattan distance is the function that takes less time to reach their solutions. In fact, the large distribution of points below 1x10e7 nanoseconds unlike the other two functions that have a lot of points above this

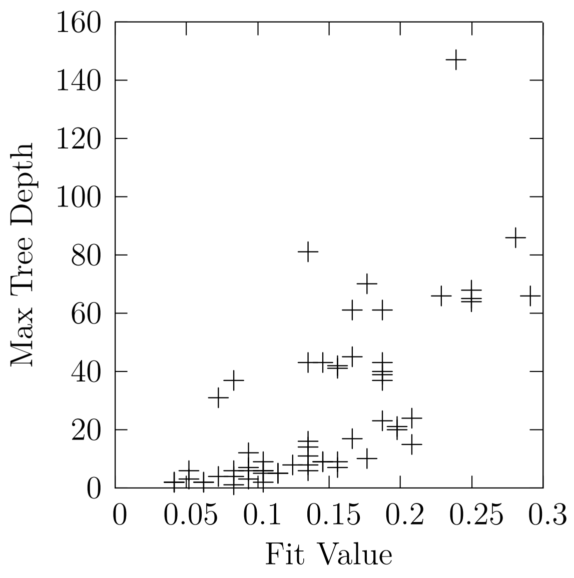

Figure 8: Max Depth for Manhattan

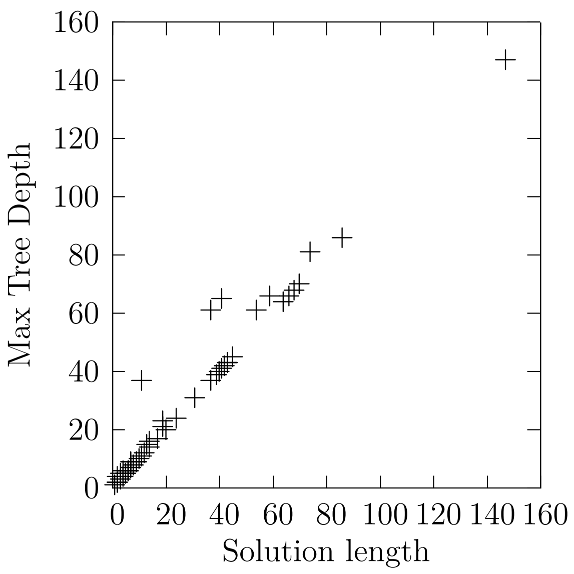

Figure 9: Max Depth for Manhattan

limit. The following is the heuristic disorder and finally the one that takes the longest time to achieve its solutions is the linear heuristic.

On the other hand, by figures 10, 12 and 14 using the length of the solutions we can infer that the time of use is increasing while the solution is even larger, that is, the longer the solution, the longer it will take to get it. However, for manhattan distance it grows slower than for the others. But not as uniform as for distance and linear heuristics. Well, about roughly the size 35 solution there is a sharp growth. But nothing serious in comparison to the times reached by the linear and the disorder.

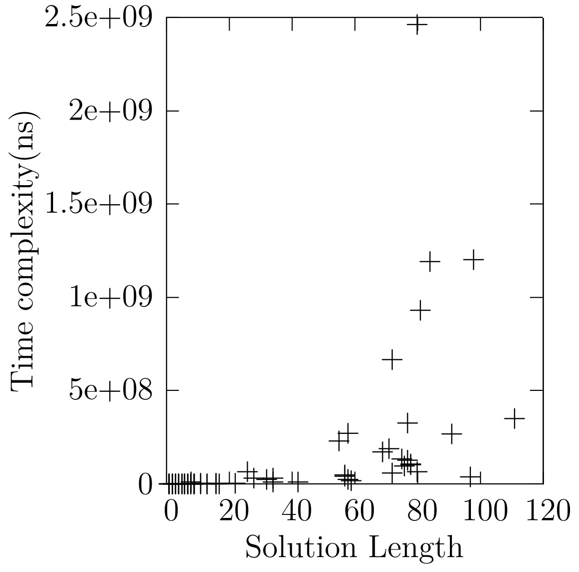

Figure 10: Duration for disorder

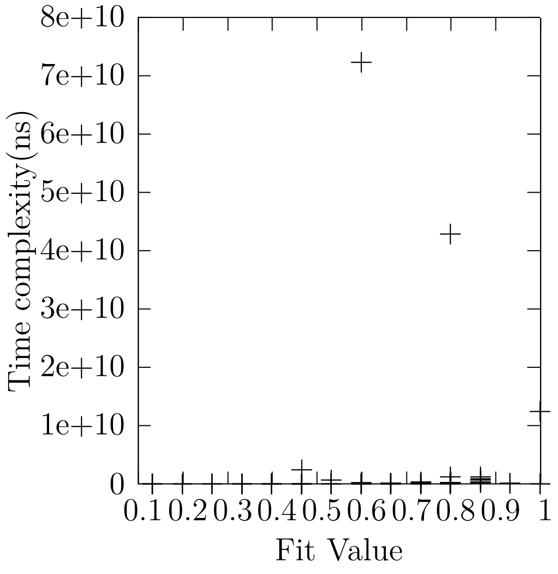

Figure 11: Duration for disorder

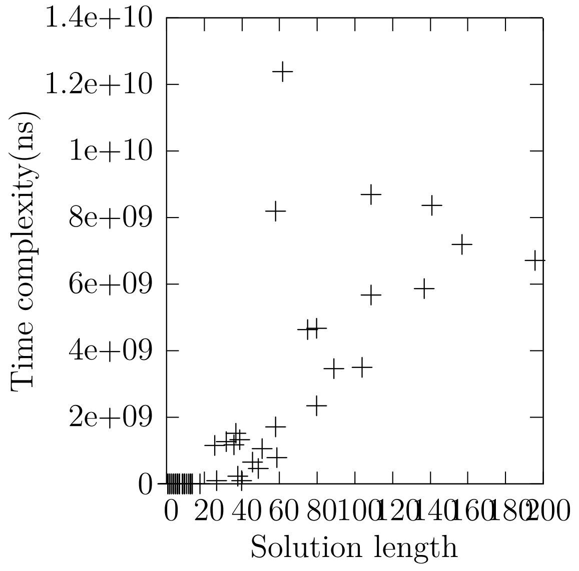

Figure 12: Duration for linear distance

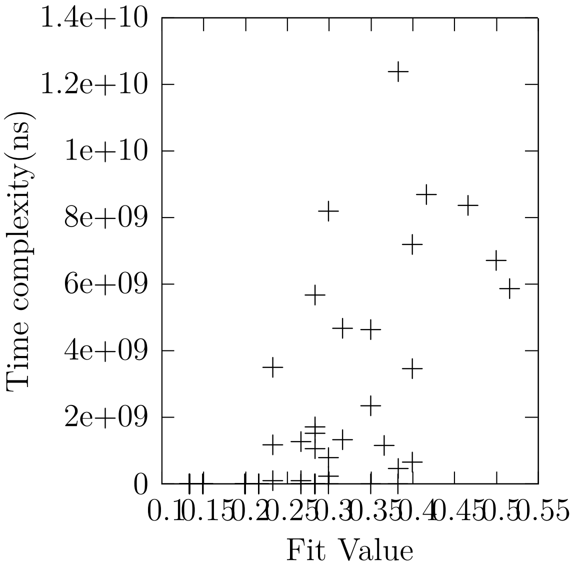

Figure 13: Duration for linear distance

0.6 States Opened Analysis

The number of open states refers to the number of states discovered at the time of generating the state tree in search of the solution. Thus, the number of states can be translated into memory usage by the program. A deeper analysis could have been done at the time of memory complexity used literally calculating in run time.

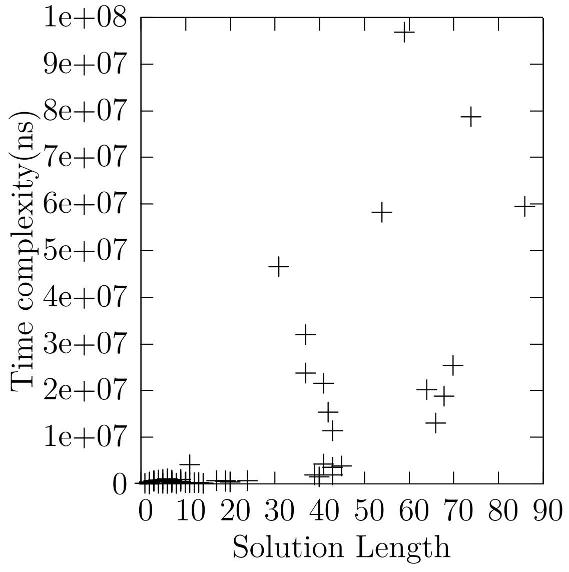

Figure 14: Duration for Manhattan

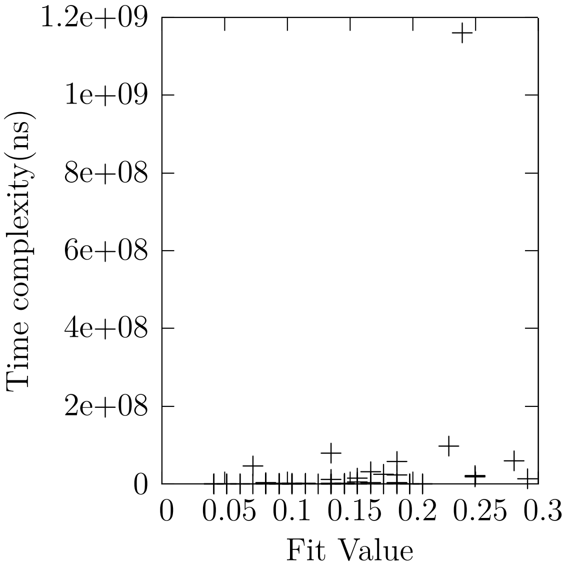

Figure 15: Duration for Manhattan

In figure 18 we can see that the Manhattan distance does not generate so many states to generate its solutions with respect to those present in figures 16 and 17. In fact, only very few points were calculated on the 100 generated states. Regarding the cost of the initial puzzle we can not conclude much, only that for values close to 0 we have very low generated statuses. But then the growth is very scattered.

In Figure 17 we can see that many states are generated with respect to the other functions. This translates to using a lot of memory in search of more roads. Therefore, not only a lot of memory is used but it will also take a long time to find the solution, a feature that was already considered in previous points. With regard to the cases generated with respect to the initial costs, nothing can be concluded because they are very scattered points.

In addition, in figure 16 we can see that it is not as efficient as the Manhattan distance but there is an improvement with respect to the linear distance. On the other hand, with respect to the costs of the initial puzzles we can see that there is a growth that is a bit more uniform with respect to the others. Thus, it is clear that the more expensive the initial puzzle, the more states will be generated in the resolution process.

Finally, using figures 19, 20 and 21, which show the number of open states and the maximum depth reached, despite the dispersion of the points, we can infer that, as we had thought, the number of generated states increases every time we go with more depth in the tree generated. The growth is slower for the heuristic disorder. While for the other two functions there is a point where it starts to grow with more speed.

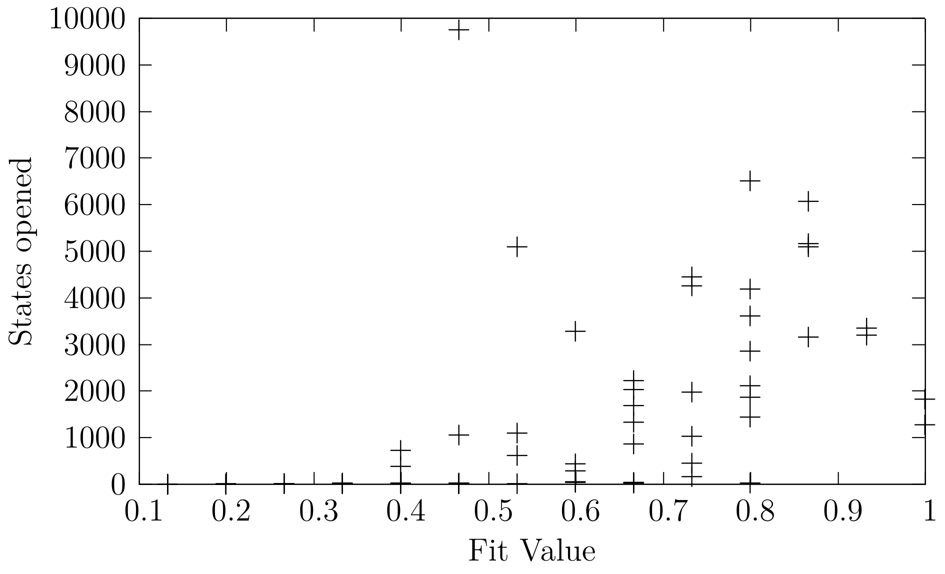

Figure 16: States opened of disorder heuristic

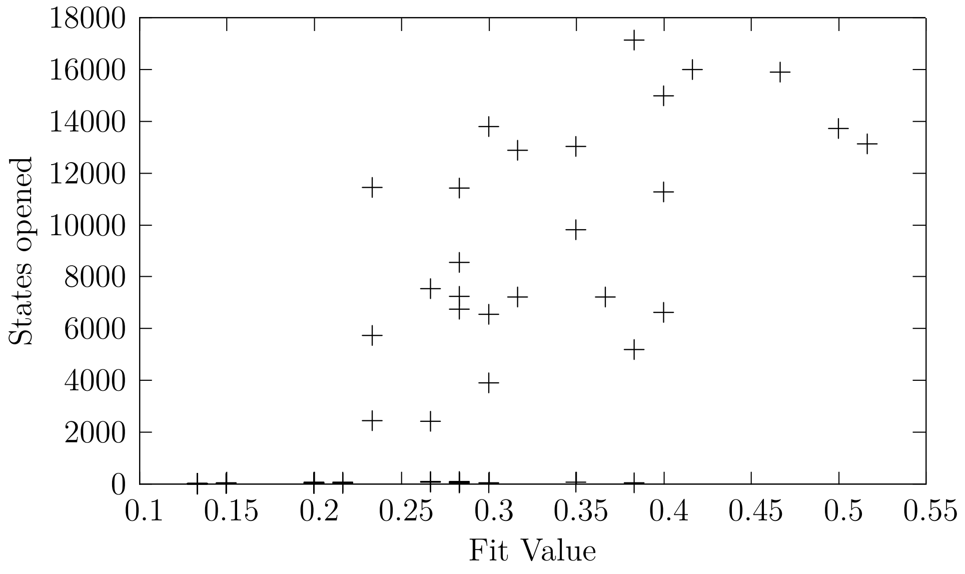

Figure 17: States opened of linear distance heuristic

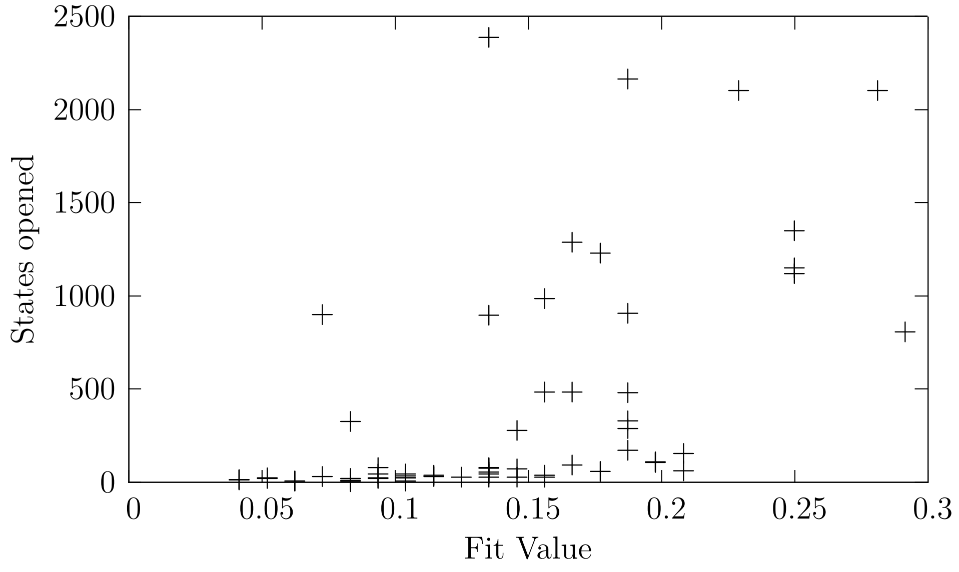

Figure 18: States opened of Manhattan distance heuristic

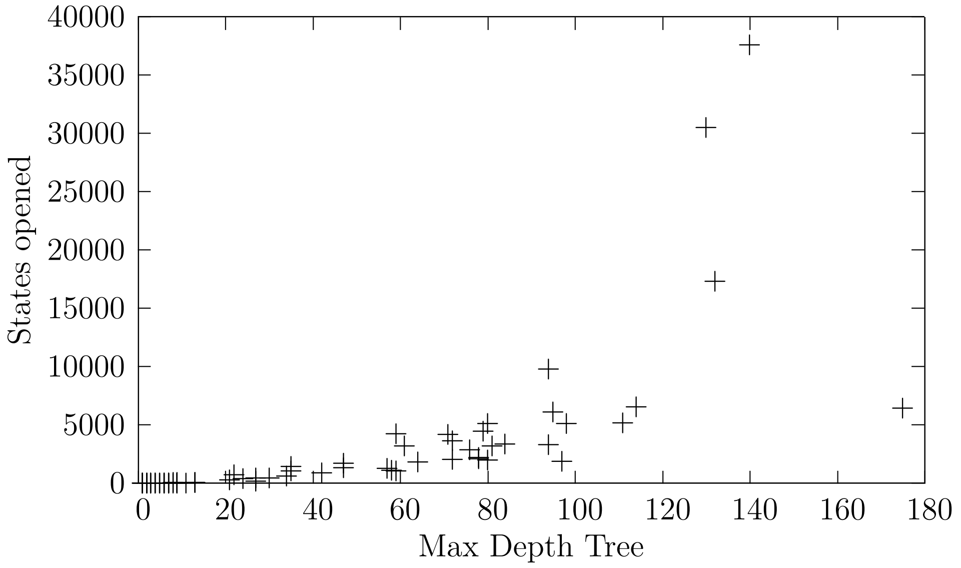

Figure 19: States opened of disorder heuristic

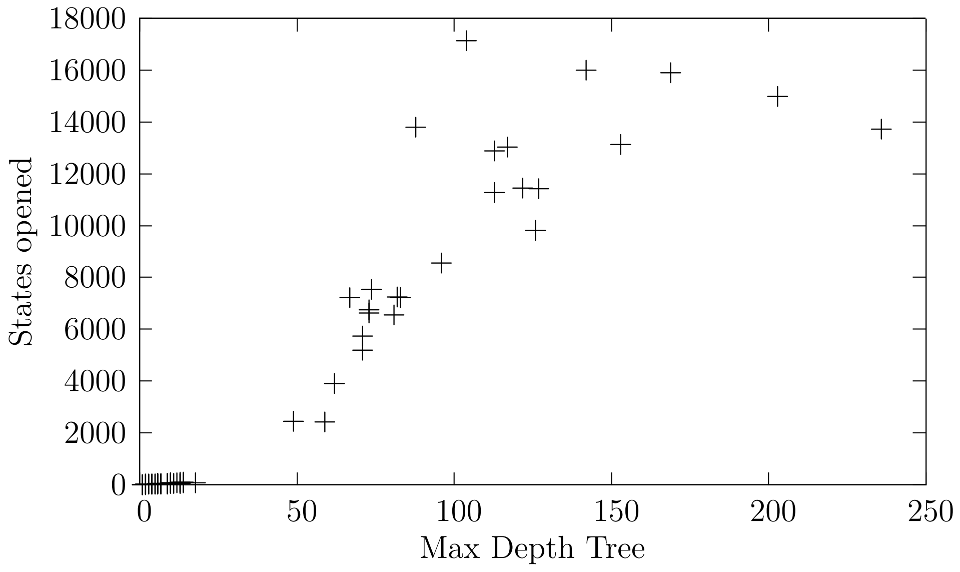

Figure 20: States opened of linear distance heuristic

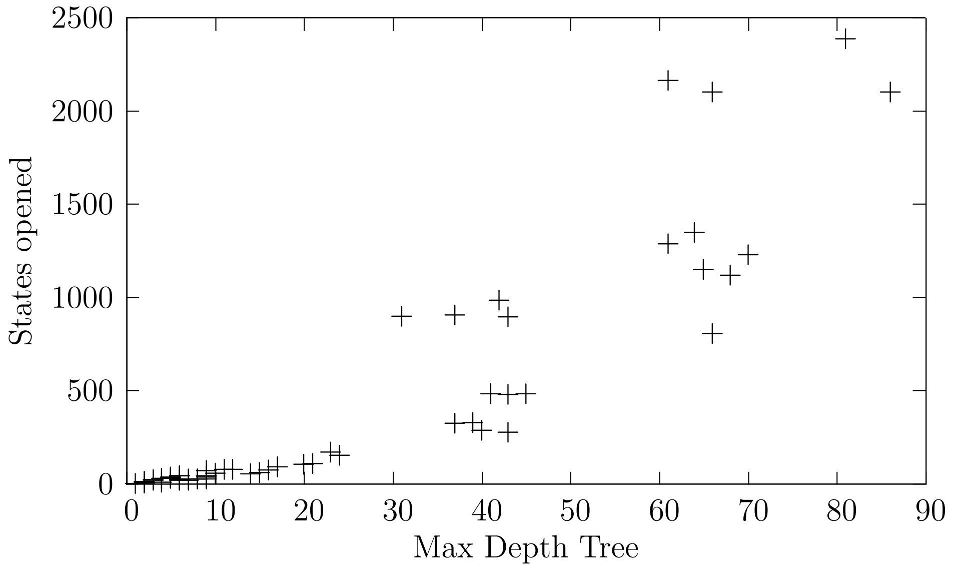

Figure 21: States opened of Manhattan distance heuristic

0.7 Bugs

Among the puzzles tested 13% was not resolved. This led us to look for a pattern in search of the bug that did not allow to find the solution. For this, in addition, a stop condition was added in the algorithm A * to be able to stop the search for solutions.

However, it was not possible to achieve a pattern because they were generated in both even and even movements, for any type of fitness function and for different numbers of iterations. Also, because the puzzles were generated randomly, it was not possible to visualize if it was due to an error in the generation of puzzles.

Thus, we can indicate that there is a problem with the resolution of certain problems or at best in the generation of random puzzles to work on performance analysis. For this reason, a maximum number of iterations has been left at the time of looking for the solution and it was added, in spite of already supposedly having been taken into account, one more condition to avoid duplicates along the way.

0.8 Conclusions

In general we can conclude that the best behavior is the Manhattan Distance heuristic. It would follow the disorder heuristic and finally the linear distance heuristic.

Firstly, we can observe that the fitness function for all the functions can show a bit the behavior of the size of the final solution. Despite the scatter of points and that all the functions behave very similar, we can see that Manhattan distance has more short solutions than the others.

Second, we can see that the maximum depth reached in the generated tree translates in general more states and so at the same time, take longer and use more memory. For this case, the Manhattan distance goes with less depth in comparison with the linear distance and the disorder that have a very similar behavior.

Third, making an analysis of the time used to find the solution, the Manhattan distance uses less time to reach its solutions, followed by the disorder function and finally the linear distance.

Finally, taking a look at the number of states generated, you can see that Manhattan distance generates fewer states, compared to the disorder that also uses less than the linear distance function. In addition, the growth of generated states goes hand in hand with the maximum depth reached.