

The Quantum 8-Puzzle: A Variation of the Classic 8-Tile Puzzle

Ian Adams
Graduate Student
Department of Computer Science
University of California, Santa Cruz
iadams@soe.ucsc.edu

February 17, 2009

1 Abstract

The quantum 8-puzzle, proposed by Dr. Robert Levinson, is a variation on the classic 8-tile puzzle where instead of a tile or space occupying one of the 9 available positions they can be distributed among the various positions. This ability to have the tile and space divided up and spread about the various positions causes a veritable explosion in the possible states. In the normal 8 tile puzzle at most there are 4 possible successor states in any given board state. In the Quantum version of the puzzle, depending on how many pieces one allows the space and tile to be divided into there is the possibility of thousands of successor states for any given board state.

The focus of my research was exploring heuristic search strategies as well as techniques for solving the puzzle in a mechanical fashion, e.g. move all space to one position irrespective of board state. It was found that heuristic searches on their own are unable to solve the puzzle, but can significantly improve the board state. Mechanical techniques can usually solve the puzzle albeit with an excess of moves but when combined with heuristics search after a partial solve can often, but not always, finish the puzzle, and these solved states used significantly fewer moves than to reach than those that did not implement a heuristic search.

2 Introduction

Why is this worthy of study you might ask? The fact that it has never been done before to my knowledge isn't necessarily important in its own right. However, its similarity to the 8 puzzle, but with a new rule set has the potential

to open up a new subfield within tile puzzle research that has the potential for real world applications, as its state space, even at the 8 tile level, is much larger than that of the standard 8 tile puzzle.

Before delving into the results of the research, we need to impart some basic vocabulary about the quantum puzzle, as well as properties, and rules for our specific quantum puzzle. For those already familiar tile puzzles the vocabulary and rules may be somewhat redundant as it is essentially the same as that used when discussing normal tile puzzles.

2.1 Vocabulary

Here we describe the terms we use in the rest of the paper to describe the puzzle.

- **Position**-This corresponds to a given spot on the board, for example, the upper left corner is a position, this is distinct from tiles. Tiles are contained within positions.
- **Tile(s)**-Corresponds to portions of a given specific tile, usually referred to as a tile number within some position.
- **Board**-Used to refer to the puzzle as a whole and its current state and distribution of tiles.
- **Space**-Considered to be a special tile, that can be swapped with any other tile in an adjacent position.
- **Entropy**-In addition to being a heuristic value (see later in the paper for its use) it is also a useful metric for measuring how close the board is to be solved. A higher entropy value corresponds to larger portions of the tiles being outside of their home positions.
- **Home**-A tile or portion of a tile is said to be in its home position if that is the position it needs to be in for in the final "Solved" puzzle state.

2.2 General Rules

Here are the general "Rules" for solving the puzzle.

- A tile in a given position may only be swapped with a space from an adjacent position.
- In any given position, the total of all the tile portions and space must add up to 1.0.
- The total portions of a tile found on the board must add up to 1.0, or again 1000 for the purposes of this research.
- The board is said to be in a solved state if all of every tile is within its home position.

T0	0334	T1	0000	T2	0111	T0	0000	T1	0445	T2	0000	T0	0111	T1	0000	T2	0334
T3	0001	T4	0110	T5	0111	T3	0110	T4	0001	T5	0111	T3	0111	T4	0111	T5	0000
T6	0111	T7	0111	T8	0111	T6	0111	T7	0111	T8	0111	T6	0111	T7	0111	T8	0111
T0	0000	T1	0111	T2	0111	T0	0111	T1	0000	T2	0111	T0	0111	T1	0111	T2	0000
T3	0445	T4	0000	T5	0111	T3	0000	T4	0556	T5	0000	T3	0111	T4	0000	T5	0445
T6	0000	T7	0111	T8	0111	T6	0111	T7	0000	T8	0111	T6	0111	T7	0111	T8	0000
T0	0111	T1	0111	T2	0111	T0	0111	T1	0111	T2	0111	T0	0111	T1	0111	T2	0111
T3	0000	T4	0111	T5	0111	T3	0111	T4	0000	T5	0111	T3	0111	T4	0111	T5	0000
T6	0334	T7	0000	T8	0111	T6	0000	T7	0445	T8	0000	T6	0111	T7	0000	T8	0334

Figure 1: An Example board state, each large position of the 3x3 grid represents a *position* and each of the 3x3 tables of numbers represents how much of each tile is in that given position .

- A swap of x-amount of space for x-amount of tile, is said to be a move of vale x.

2.2.1 Rules Specific to This Research Project

- The tiles and space are restricted to 3 significant digits of division. In essence meaning that each tile and the space can be divided into 1000 pieces. From now on I will be referring to moves in integer values up to a thousand for single moves.

2.3 Properties of the Quantum 8-Puzzle

- Any portion of the space, given enough moves, may be gathered in ANY position.
- Given 2 tiles, if there is at least 1 space between them one can execute a "swap chain" and switch all tile portions between the 2 positions. See figure 2 for an example.
- Given the above 2 properties it has been shown (albeit in a somewhat hand-wavy fashion) that it is possible to solve ALL possible states of the quantum puzzle. This is due swapping with adjacent positions repeatedly one can move any portion of any tile to any position on the board.

3 Methods and Approach

Here we describe our approach to solving the puzzles.

<u>Pos 1.</u>	<u>Pos 2.</u>
A-0	A-999
S-0	S-1
B-1000	B-0

A-0	A-999
S-1	S-0
B-999	B-1

A-1	A-998
S-0	S-1
B-999	B-1

Figure 2: A and B represent arbitrary proportions of one or more tiles in . S represents the Space. Notice how the space returns to its original position, and we have switched one piece of A for one of B. This can be repeated until all of A and B have switched positions

3.1 Board Generation

The following describe the various generated boards that the A* solver ran on.

- **Random**-A randomly generated that scattered the space and tiles across the board based on pseudo-random numbers from a random number generator.
- **Symmetric**-A board where the tiles and space are evenly distributed across the board. Each position contains 111 of each tile and the space, with the extra 1 tile for each tile going into its home position.
- **Normal**-A board where all of a tile occupies some position. These boards look exactly like the standard 8 tile puzzle initially.

3.2 A* Heuristic Search to Improve the Board State.

Since it does not appear that the Quantum 8 Puzzle has ever been done before (at least not to the author or the proposer of the puzzle's knowledge) , there was little in the way of previous work to leverage beyond heuristics already proposed for the standard 8 tile puzzle. That said however, previous heuristics and small variations upon them did improve the state of the board (recalling the we are using the value of entropy as explained earlier as our metric for the goodness of the board state), and in the case of Manhattan distance significantly so.

The A* search ran until it was determined that no more good moves could be made. This was determined via a flag that was tripped if any of the following 3 conditions were met: 1) A move is made and the following move immediately undoes it, 2) 2 moves from the same position with the exact same tile and space

proportions are done in row, or 3) 25 consecutive moves where only 1 tile piece is being moved . Admittedly it is possible that the search might have found good moves again after the last condition caused it to terminate, but it is my belief that this is highly unlikely as I observed infinite loops in every case I saw.

One question that is likely to be posed is: *Why not back track when no good moves are found?*. It is for 2 reasons that we did not implement any backtracking. The first reason being that backtracking would result in exponential explosion due to the huge number of possible moves that can be made on any give board state. The second is that it is of interest to see what heuristics provide the greatest improvement in board state on the first try so to speak.

3.2.1 Key for Figures

- **Start Hval**-Starting Heuristic Value, this is the value the heuristic provides on the board before any moves are made.
- **End Hval**-Ending Heuristic Value, this is the value the heuristic provides on the state of the board after it stops making good moves.
- **Entr Imp**-Entropy Improvement, how much of an improvement in entropy the A* search provided.
- **Moves**-How many moves were made in the course of the search. Recall that one piece of the tile swapped for one piece of the space counts as ONE move.

Note that most results have an accompanying bar chart that may be found at the end of the paper for an easier visual comparison.

3.2.2 Start State Data

This data provides the average entropy and Manhattan distances (see the immediately following subsection for explanation of Manhattan distance) of the various board generations used in the research. It is helpful to the reader examine figure 3 results to get a better idea of the improvement of the board state, as well what the average starting state might look like in addition to the possible extremes.

3.2.3 Manhattan Distance

This heuristic is much the same as it was for the 8 tile puzzle, it is a measure of how many moves each tile has to make in order to be in its home position assuming nothing is in the way. See figure 4.

Random Generation		Symmetric Generation		"Normal"	
Entropy		Entropy		Entropy	
Average	7115.98	7104		7320	
High	7735			8000	
Low	6317			5000	
Manhattan		Manhattan		Manhattan	
Average	14717.75	14652		15330	
High	17668			22000	
Low	12186			7000	

Figure 3: Generated Board Statistics

Board Type	Start Hval	End Hval	Entr Imp	Moves
Sym	14652	3220	4995	11434
Rand	15230	1205	6273	14027
Norm	15800	11698	1200	4103

Figure 4: Manhattan Distance Heuristic Search Results

3.2.4 Distance to Row

This heuristic is just a variation on the Manhattan distance but instead of calculating to distance to reach the home position, it just calculates how far the tiles must go to reach their correct row. See figure 5.

3.2.5 Distance to Column

This heuristic is also a variation on the Manhattan distance, exactly the same as the distance to row heuristic, but this time it is based on the distance to the column. See figure 6.

Board Type	Start Hval	End Hval	Entr Imp	Moves
Sym	7326	2886	1000	4441
Rand	7449	2224	1340	5770
Norm	6550	5392	402	1111

Figure 5: Distance to Row Heuristic Search Results

Board Type	Start Hval	End Hval	Entr Imp	Moves
Sym	7326	2886	666	3022
Rand	7150	2398	132	5230
Norm	7600	6786	202	815

Figure 6: Distance to Row Heuristic Search Results

Board Type	Start Hval	End Hval	Entr Imp	Moves
Sym	5340	6131	245	691
Rand	5440	6892	740	2051
Norm	5400	5702	2	305

Figure 7: Quadrant Heuristic Search Results

3.2.6 Quadrants

This heuristic takes a measure of how much of the tile is within the correct quadrant of the board. Note that since this heuristic treats higher values as better, it tries to increase its value. This is why the ending Hvals are higher than the starting ones. See figure 7.

3.2.7 Entropy

A measurement of how much of the tiles are out of position. Not particularly useful as a heuristic by itself. See figure 8.

3.3 Conclusions on Single Heuristic A* search

Far and above the most useful heuristic of the ones that were tried was the venerable Manhattan distance heuristic. It provided by far the greatest board improvement, and made the most moves, and though not quantified here, some of the best moves before the search quit.

The distance to row and distance to column heuristics also proved to decent

Board Type	Start Hval	End Hval	Entr Imp	Moves
Sym	7104	4884	2220	2221
Rand	7028	5485	1542	1548
Norm	7000	6398	601	603

Figure 8: Entropy Heuristic Search Results

job in improving the state of the board as far as entropy was concerned. Quadrants and entropy by themselves were not all that useful as a means of quickly getting to a close to solved state.

The logical thing to do in the future would be to start combining and alternating heuristics within a search to see if one can further improve upon the Manhattan distance.

Unsurprisingly the normal board states were impacted the least by A* search, this is due to the non-backtracking nature of my implementation as often the heuristics provide no obvious good move with the states of those boards due to the space and tiles being highly concentrated into one position initially.

4 Position Solver: A method for mechanically solving puzzles

With the objective of the puzzle being to gather all pieces of the tile in their home positions, it is intuitive to solve a single position at a time. The method used was relatively simple in concept, though hardly perfect.

The idea is to iteratively shuffle each tile towards its respective home position. Each time you move want to move some portion of a tile towards its position, you will need to ensure that there is enough space in the position you want to move to, if there isn't, then you must also shuffle space towards that position until there is enough to complete the proposed move. See figure 10 for pseudo-code.

This process, when done repeatedly, gathers all the tile up into its home position over the course of multiple moves. The problem comes from when there is not enough space in a position the tile needs to move in. Its easy enough to simply grab some random space and move it to the needed position, but this can be highly damaging to the board, and even mess up the proposed move that the space is meant for as the space might be moved through the source position where the needed tile is, or it may damage an already completed position. These problems can be abated through the use of tile and position locks, though these come with their own set of problems.

Tile and positions locks work by locking some tile or position and saying that any given move may not use the locked tile or position. These locks come with their own sets of problems though as the solver can now get stuck. In examining the board in figure 9 (remember T4 is the space) we see just such a problem where the solver has gotten stuck. T2 is gathered all in one position (center row right column), and is a single position away from being solved. However, because T0 and T1 are solved there home positions are locked. Additionally

T0 1000 T1 0000 T2 0000	T0 0000 T1 1000 T2 0000	T0 0000 T1 0000 T2 0000
T3 0000 T4 0000 T5 0000	T3 0000 T4 0000 T5 0000	T3 0417 T4 0000 T5 0333
T6 0000 T7 0000 T8 0000	T6 0000 T7 0000 T8 0000	T6 0105 T7 0000 T8 0145
T0 0000 T1 0000 T2 0000	T0 0000 T1 0000 T2 0000	T0 0000 T1 0000 T2 1000
T3 0000 T4 0000 T5 0010	T3 0000 T4 1000 T5 0000	T3 0000 T4 0000 T5 0000
T6 0123 T7 0592 T8 0275	T6 0000 T7 0000 T8 0000	T6 0000 T7 0000 T8 0000
T0 0000 T1 0000 T2 0000	T0 0000 T1 0000 T2 0000	T0 0000 T1 0000 T2 0000
T3 0275 T4 0000 T5 0119	T3 0103 T4 0000 T5 0280	T3 0205 T4 0000 T5 0258
T6 0246 T7 0137 T8 0223	T6 0169 T7 0213 T8 0235	T6 0357 T7 0058 T8 0122

Figure 9: A stuck board state

T2 is tile locked since any gathering of the space should avoid disturbing its position. These locks combine into a deadlock, as the space cannot move up or right to get closer to where it needs to be for T2 to reach its home position.

4.1 Dealing with Stuck Board States

There are 2 methods that I use of alleviating these stuck conditions, though they are crude, they work. One is to simply make 30 random moves using any unlocked positions and try again. The other is to solve the corners first, as this helps prevent most of the states where a stuck position could occur from happening. Neither of these methods is perfect, but they reduce rate of the solver getting stuck by a significant portion, where on random board states it only gets stuck permanently stuck about 1 percent of the time.

It must be noted however that this individual tile solving method by itself, again, has problems much like the heuristic search with getting stuck very quickly on normal boards. The method of alleviating this particular problem was to equally distribute the space initially before attempting to solve the board.

4.2 Position Solver by Itself

This section briefly covers results found by running the position solver with no heuristic help.

4.2.1 Corners First

This particular method ran the position solver on the corners of the puzzle first in clockwise order starting in the upper left. Afterwards it solved the remaining positions also in a top down, left to right order. See figure 11

```

Position Solver(tile to solve)
  lock(tile to solve)
  while(tile is unsolved)
    for each position
      if current position is locked, break
      propose tile move towards home position
      if move is into a locked position, break.
      if insufficient space available for move, need Space(proposed move,amount)
      do move
    if all of tile is in the home position, tile is unsolved=false
  lock(tile home position)

Need Space(where needed,amount needed)
  while(not enough space where needed)
    for each position
      if current position is locked, break
      propose move towards where needed position, not using locked tiles
      do move
    if space in where needed>amount needed ,not enough space where needed=false

```

Figure 10: Pseudocode for the position solver

Random	Symetric	Normal
#of Moves	#of moves	#of moves
28150	27282	*

Figure 11: Corners First Position Solver Results.

4.2.2 Distribute Space Initializer then Tile by Tile

As you can clearly see, the average number of moves made with the space distribution and the random move method for getting unstuck causes a very high number of moves to be made to solve the puzzle, particularly with the more difficult start states that stem from normal puzzle distribution. See figure 12

4.3 Combining Methods-Heuristics with Position Solver

A surprisingly effective method I found was solving the corners first, and then letting Manhattan distance run on its own, and interestingly enough, with the

moves Made	65913
------------	-------

Figure 12: Here are the results on a normal board start state when distributing the space evenly before a tile by tile solver.

Hval Start	14020	Approx 55% solved
Hval end	2810	
Ent Improv	5259	
Hval improv	11209	
moves	57438	

Figure 13: Distribute Space, Corner Solve, then Manhattan Distance on normal start states.

Hval Start	14660	Approx 60% solved
Hval End	2187	
Ent Improv	5834	
Moves	24720	
hval improv	12473	

Figure 14: Distribute Space, Corner Solve, then Manhattan Distance on normal start states.

corners solved, more than half the time on both the randomly generated and normal puzzle start states it was able to run to completion with a solved puzzle state.

4.3.1 Distribute Space Initializer, Corner Solver, Manhattan Distance on Normal State States

This methodology distributes the space evenly throughout the board initially, then solves the corners and ends by running a manhattan distance heuristic search. Though not guaranteed to solve, we see a great improvement in entropy, and it generally significantly fewer moves to achieve this state. 13

4.3.2 Corner Solve, then Manhattan Distance on Random Start States

Here we run the corner solver, followed by the manhattan distance on the random start states. Though not guaranteed to solve the puzzle, again we see great improvement in the board state, and with FAR fewer moves than brute force solving, though still more than letting a Manhattan distance search finish. 14

5 Conclusions

There weren't too many surprising results from this initial work, so I wont waste time repeating the above results verbatim, but there were a few that were

rather interesting. In particular the find that it seems to be advantageous to solve corners first, and how effective Manhattan distance is in improving board state on randomly generated boards. The significance of these results stems from the fact that it is almost directly opposite of what is done on a normal 8 tile puzzle as there the corners are much more in flux being moved until the very last few states before being solved. The results from this research also are significant in that they could possibly open new fields of research (assuming we have not missed this line of puzzles under a different name) that could be useful both as toy problems and with real world applications. As for the puzzle as a whole and my research experience, Ive found this initial work to be gratifying and quite educational as it illustrated a lot of the trial and error that goes one when examining a completely new problem with little to work on. It also gave me a great appreciation of the need for better planning earlier on as I wasted a significant amount of time just trying to come up with a good board representation.

5.1 Comments

As a comment about the nature in general of the Quantum 8 Puzzle, I feel it shows great promise as a field to be expanded upon. Its larger state space makes it much less toy like than the standard 8 puzzle, and its rules allow for a great many variations. Obviously you could expanded the size to 4x4 and up, but also for non-square boards to be investigated. One could also change the puzzle up in many ways: by making certain positions impassable from the start, setting good enough solved conditions based on entropy or other heuristics, making certain positions impassable from the start, or even having competitive solving where an agent is attempting to mess with the state as you are solving it, there are a huge variety of possible variations that this rule set allows.

5.2 Future Work

I am curious to see if it is possible to convert the quantum tile puzzle to other known computer science problems, as it bears at least a superficial resemblance to some logistics problems (how to move material around with limited transportation) and network flows, moving material around with limited bandwidth, in the puzzle case, the space could be considered the bandwidth. For future work, given more time, there is MUCH that could be done. The results here were from a rather scattered approach, and with these initial conclusions and information I could make a much more focused effort. There are many other possible heuristics to be tried, in addition to combinations of the ones above. There are also many areas within the position solver that could be improved using heuristics themselves, such as when gathering the space, move tiles towards there home positions in addition to moving the space to where it is needed.

A * Heuristic Search Results

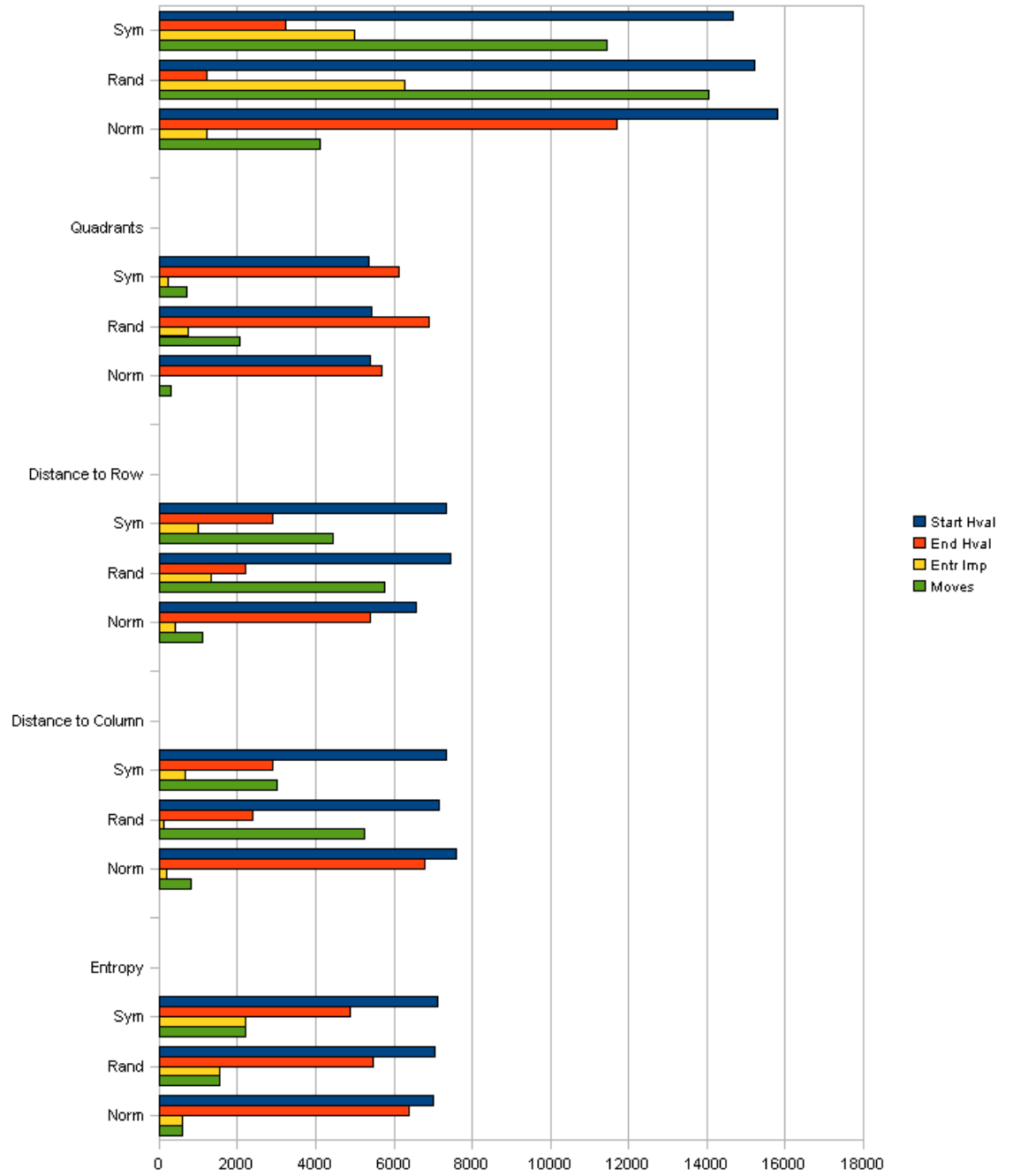


Figure 15:

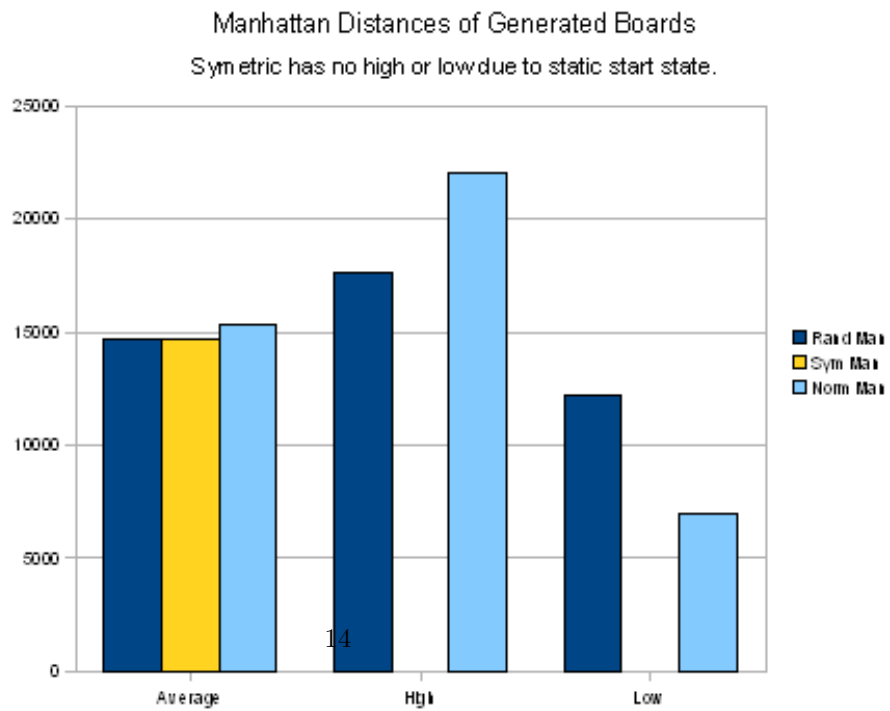
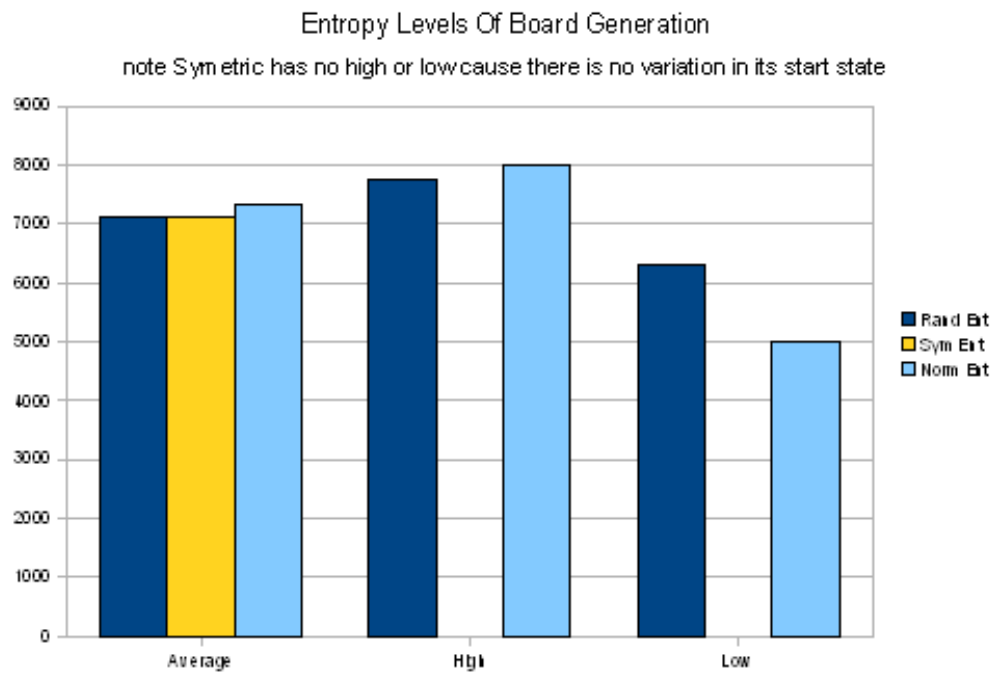


Figure 16:

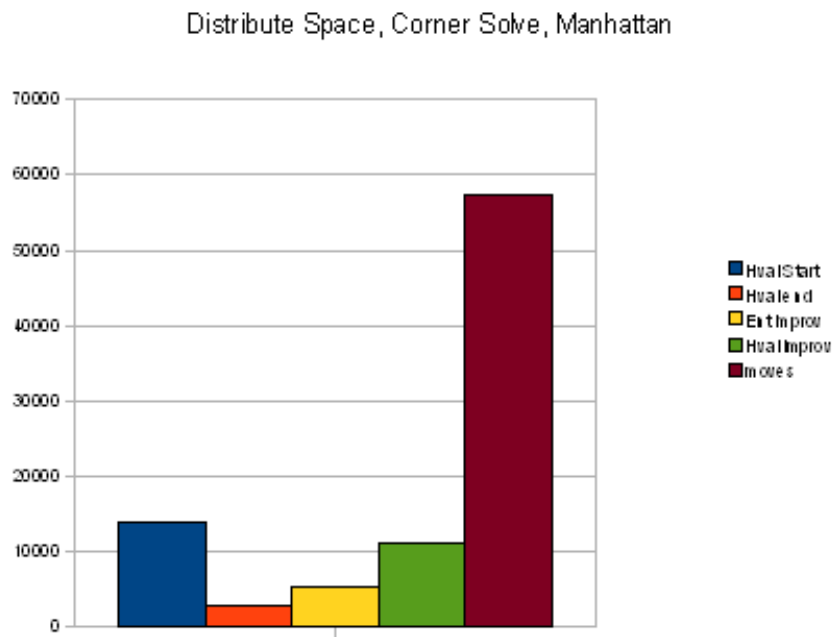
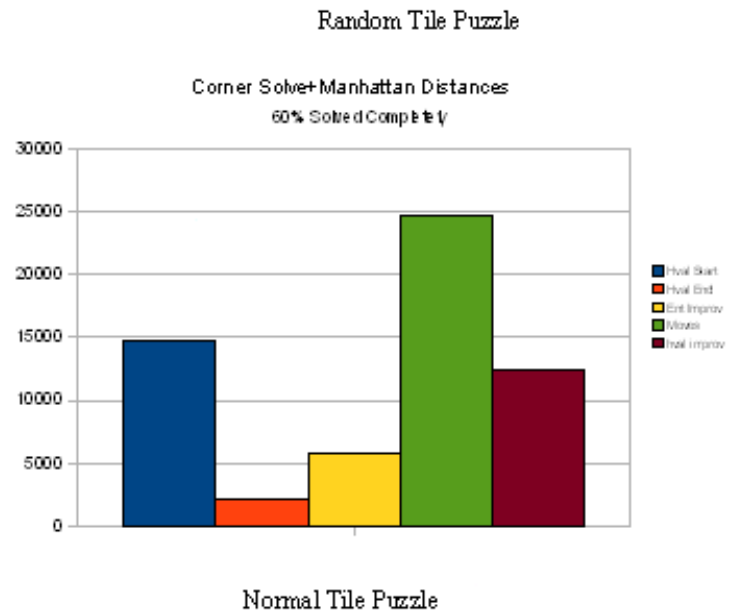


Figure 17: