**ABHINAV SHARMA – 4 CREDITS**

**SHYAM RAJENDRAN - 4 CREDITS**

**SAIKAT ROYCHOWDHURY - 4 CREDITS**

# Summary

We have implemented the following enhancements in code for faster solution convergence.

1. Since a given board instance heuristic is constant, we pre compute the heuristic value for each instance of the board during its initialization (constructor) using a map and bring the score lookup time to O(1).
2. We have used a one dimensional representation of 8 puzzle. This helps us avoid any costly one-one matching with a goal board. (board\_array[i] = i

is the invariant for our goal board). Also such a layout allows an easy offset based access of the board tiles. This also means space efficiency, since we do not explicitly store any goal board for each board instance.

## Gashnig’s Heuristic

We compute Gashnig’s heuristic score as follows

1. CASE I - Blank cell is at correct position.

We maintain a list of misplaced tiles in a list. The list is sorted by tile number i.e a misplaced tile with lower id is placed before a higher misplaced tile. We peek at the head of the sorted list this gives the tile and its current position in the 8 puzzle. We swap the tile with the blank space and update the new position of the tile.

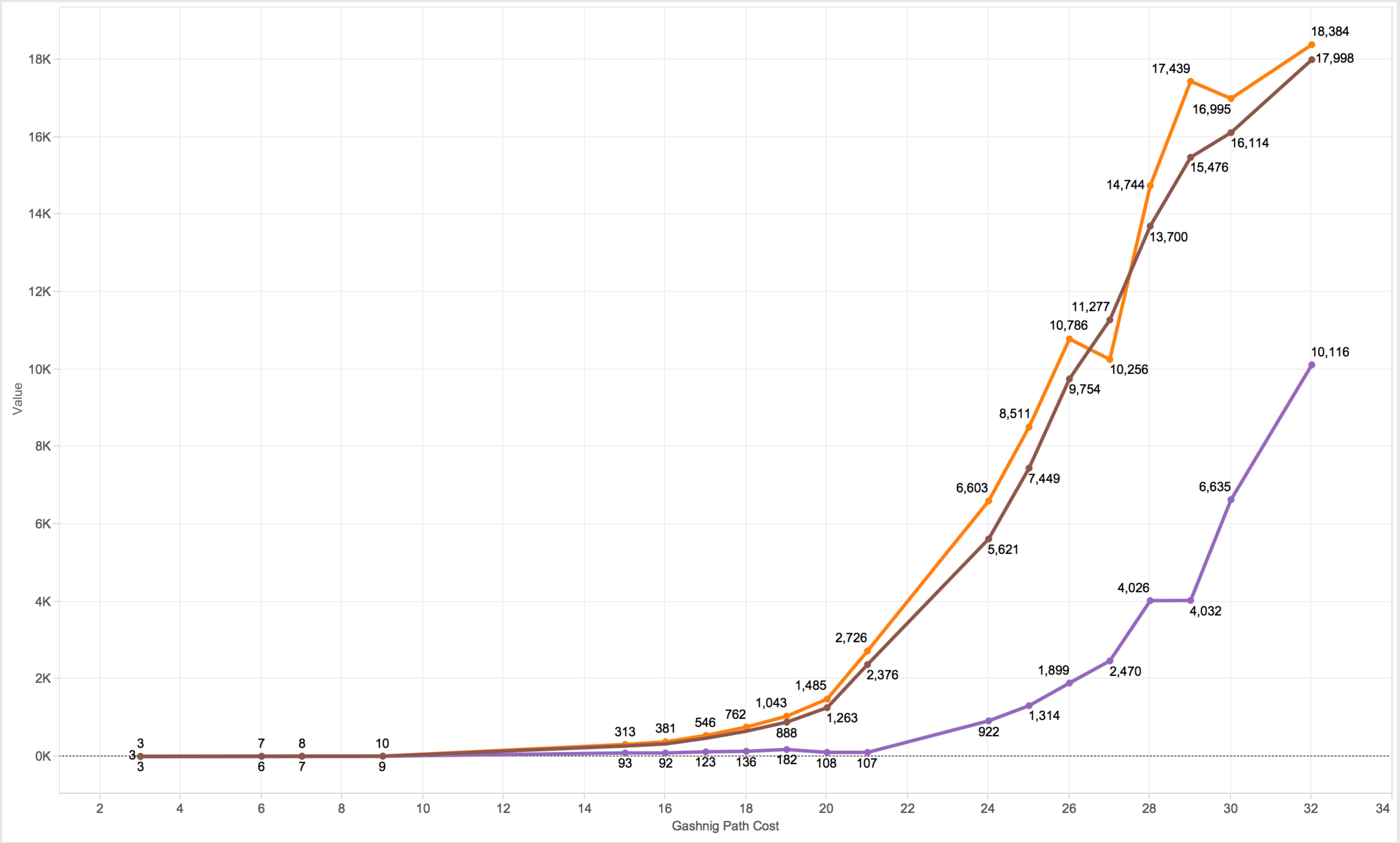
1. CASE II - Blank cell is not at correct position

For this scenario, we determine which tile is the correct owner of the blank cell. Then that tile is located and replaced with this blank cell. Thus the swapped tile has now reached its correct position in the 8 puzzle. So we remove it off the list of misplaced tiles.

Every shuffle is counted as one step. The aggregate number of steps is the heuristic score.

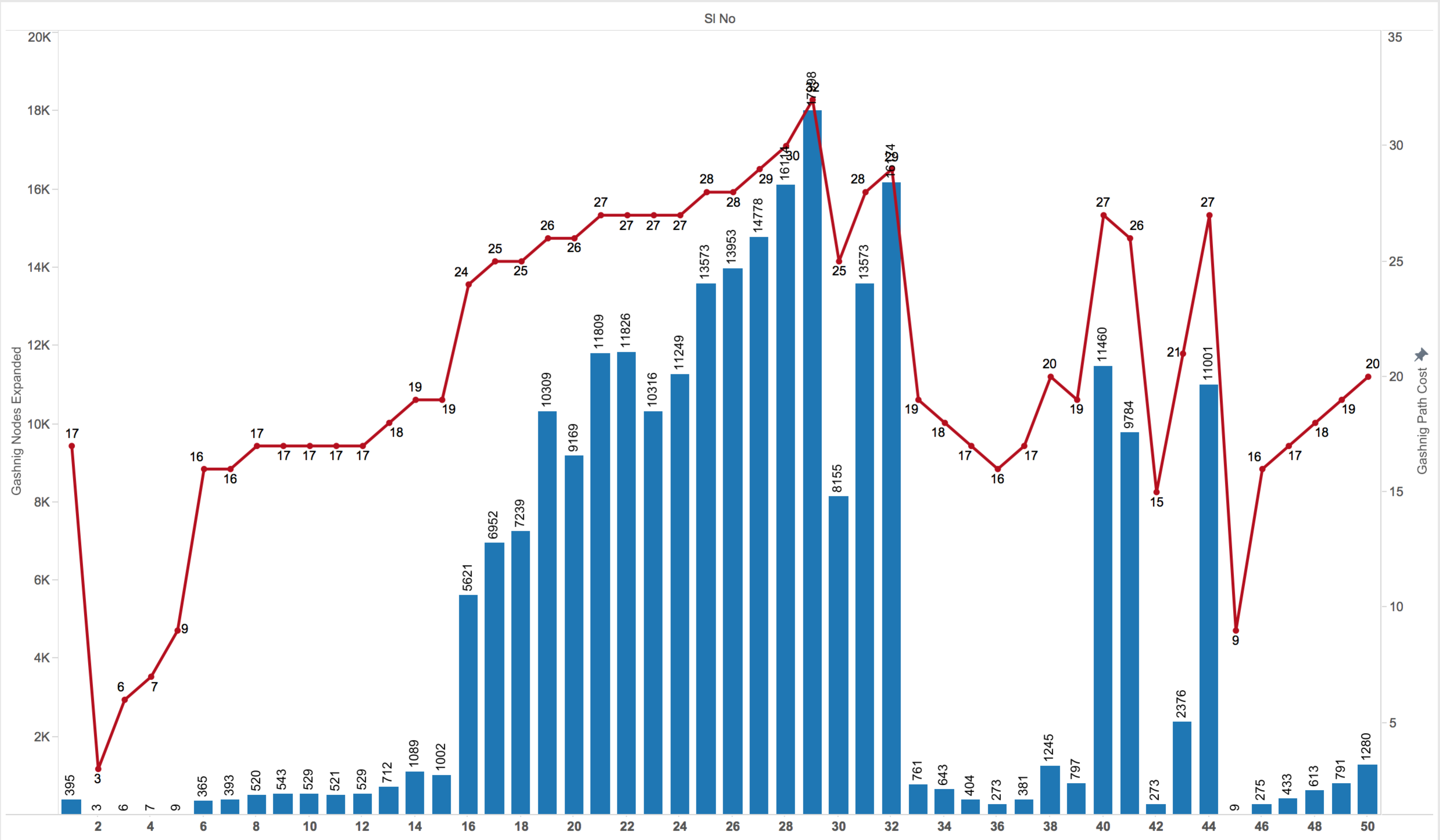
## Inference and observation from the different heuristics applied in solving 8 puzzle game.

The plot of nodes expanded against the path cost proves our understanding that relaxed heuristics such as Gaschnig's actually expand far more nodes to converge to solution than heuristic based on Manhattan distance.



***Graph 1:***

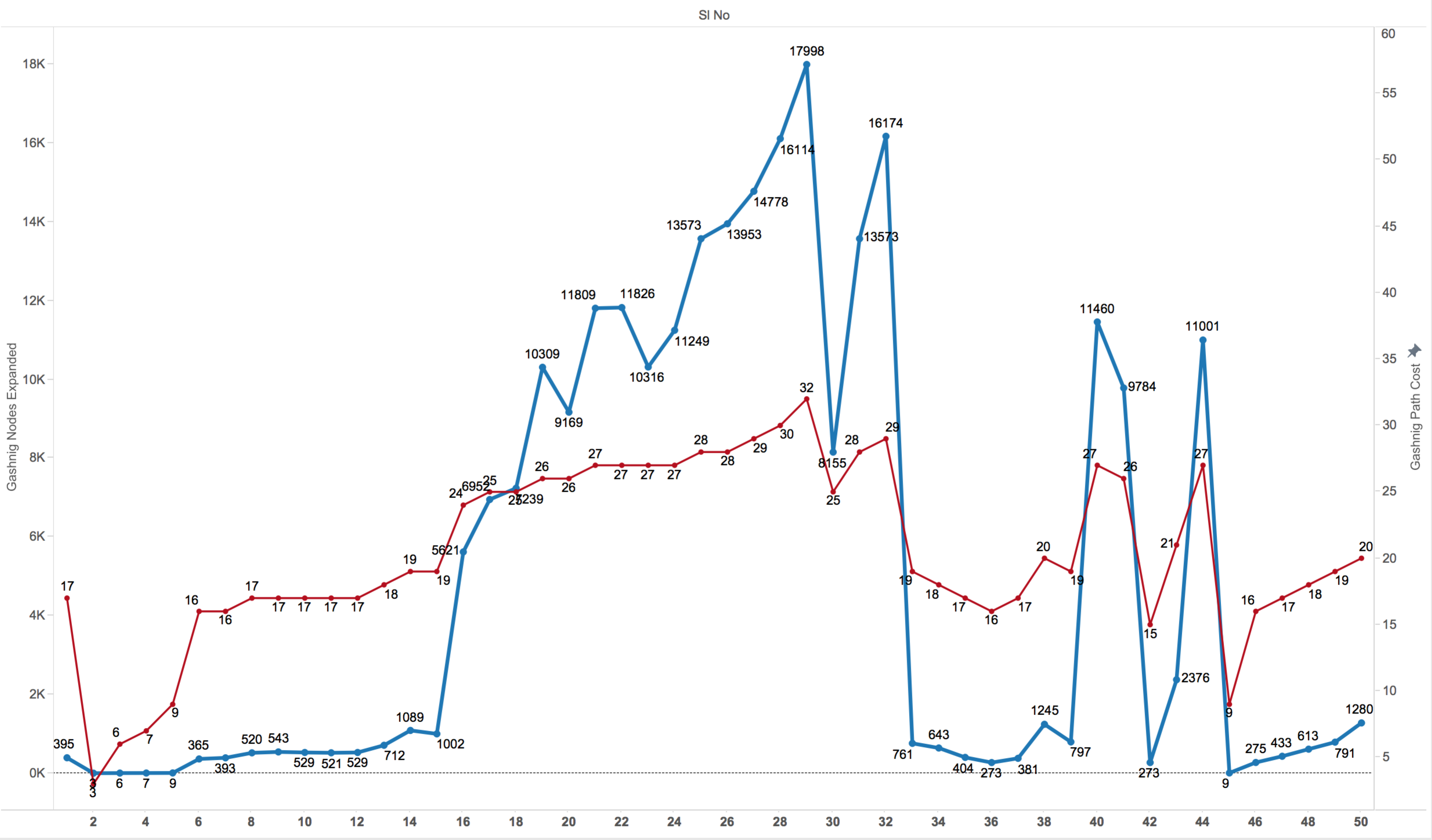
*Nodes Expanded vs Actual Path cost for Manhattan Vs Misplaced Tiles Vs Gaschnig's*



***Graph 2***

*Path cost vs Nodes expanded for 50 puzzles solved using* ***Manhattan heuristic***

Another interesting observation is when we plotted path cost vs nodes expanded across the 50 puzzle experiment for each heuristic. [ Graph 2 ] . We can observe the relation of how complexity of the puzzle pulls the nodes expanded cost more. But its also relatively constant for easier puzzles ( puzzles solved with path cost less than 10 ) where the nodes expanded is at an average around 50-100.



***Graph 3***

*Path cost vs Nodes expanded for 50 puzzles solved using* ***Gaschnig's***

***heuristic***

For the same 50 puzzles, the average nodes expanded when solved using Gaschnig stands at around 200-300 for [ **Significant increase** of expanded nodes from Manhattan heuristic solution].

We can observe the trend for Misplaced Tiles heuristics as well from the Graph 4.

The expanded nodes are slightly higher compared to Gashnigs heuristic solution.

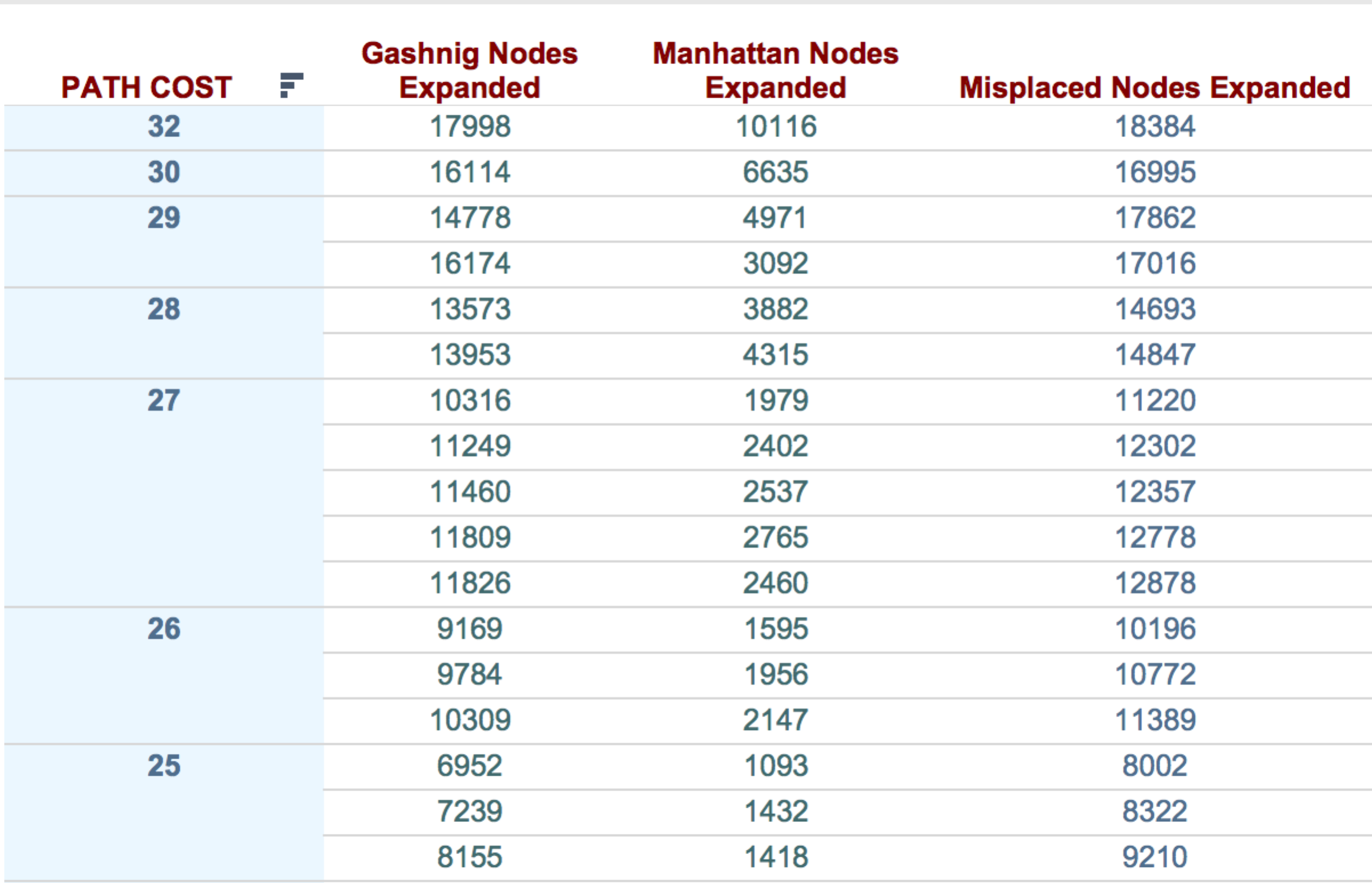


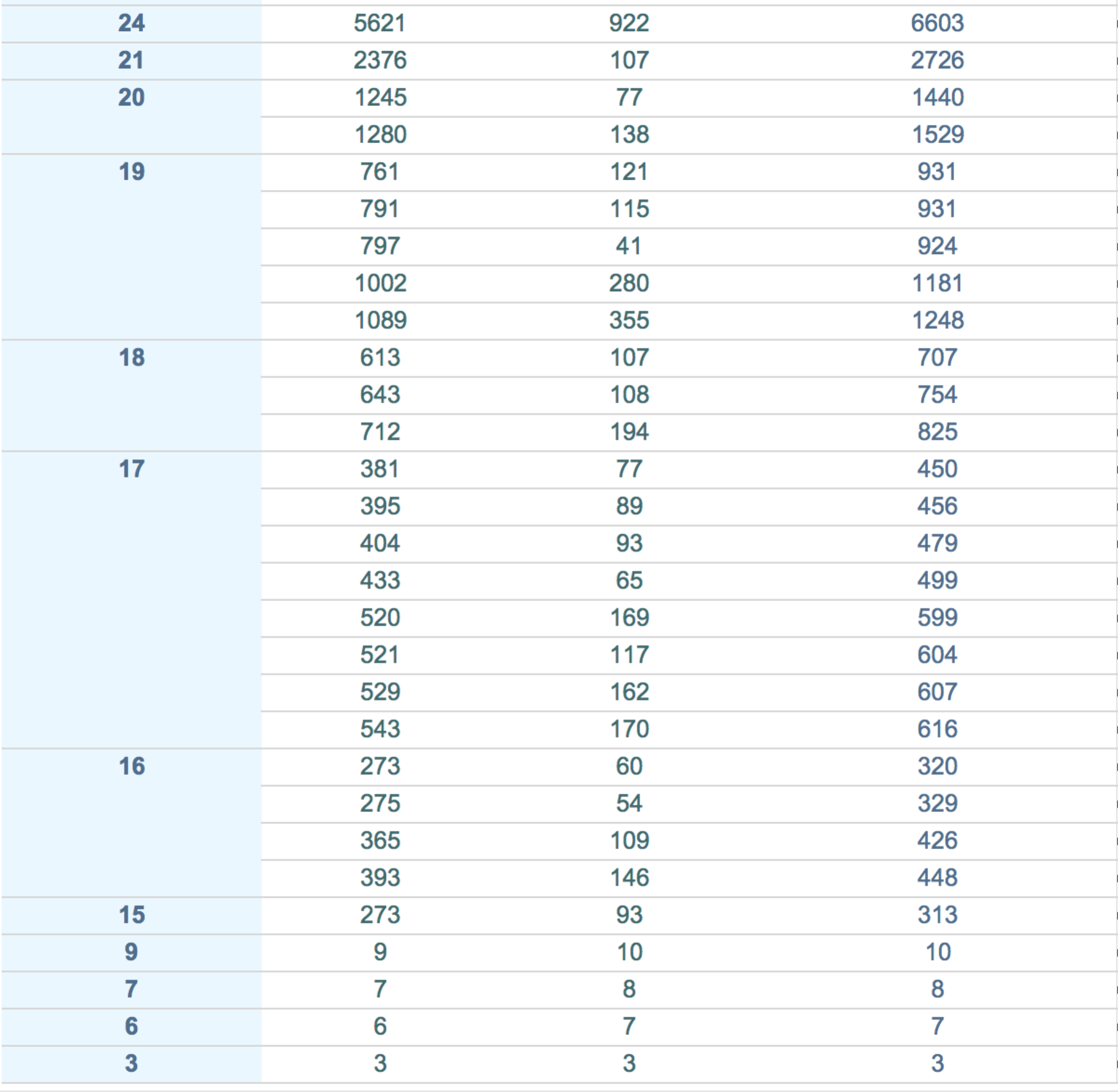
**Graph 4**

*Path cost vs Nodes expanded for 50 puzzles solved using* ***Misplaced Tile***

***Heuristic***

## Tabulated Summary of different Heuristic’s Performance across 50 Puzzles





The above table shows the performance of each heuristic in terms of nodes expanded across the 50 8 puzzles games solved.