**Assignment 2**

Uriel Zaed 319435608

Ran Weissman 312236466

**Implementation Design:**

The 8-tile puzzle consists of a 3x3 grid with 8 numbered tiles and an empty space. The goal is to rearrange the tiles from a given initial state to a goal state using valid moves (up, down, left, right) by sliding the tiles into the empty space.

Each state of the puzzle can be represented as a 3x3 grid, where each cell contains a number from 0 to 8 (0 represents the empty space).

A node in the tree search contains:

* State
* Node parent
* Moves from initial state - g(n)
* Estimate moves to goal state - h(n)

The main program initializes the initial state, goal state, and calls the A\* or Weighted A\* algorithms with a chosen heuristic.

**Heuristics:**

We implemented 2 admissible heuristics as follow:

1. Manhattan distance

Measures the distance between a tile's current position and its goal position, disregarding any obstacles in the way. To calculate the Manhattan Distance heuristic in the 8-tile puzzle, we sum up the Manhattan distances (absolute gap of row distance and column distance) for each tile from its current position to its goal position.

The Manhattan Distance is admissible because it always underestimates the actual number of moves required to reach the goal state from the current state.

1. Misplaced Row Column tiles

The Misplaced Tiles heuristic counts the number of tiles out of their respective rows and columns of the goal position.

The heuristic is an admissible heuristic as it calculates the number of tiles out of their respective rows and columns. Since each misplaced tile necessitates at least one move to reach its correct position, it provides a conservative estimate of the total moves required to solve the puzzle.

**Algorithms:**

We implemented 2 algorithms as follow:

1. A\*

The A\* algorithm is an extension of Dijkstra's algorithm, but with the addition of heuristic to improve its efficiency.

* Completeness:

A\* with an admissible heuristic will eventually find a path if exists, because in the worst case the algorithm goes over the hole nodes of the tree. So, if there is a solution A\* will find it.

* Optimality:

A\* is optimal with an admissible heuristic because the heuristic guides the search towards the goal while ensuring that the algorithm never overlooks the possibility of a better path. This combination of guidance and completeness ensures that A\* finds the shortest path in a graph efficiently and optimally.

* Soundness:

A\* will either find the optimal solution or correctly report that no solution exists because in the worst case the algorithm goes over the hole nodes of the tree.

* Termination:

The A\* algorithm has an explored set that contains the visited nodes. Therefore, the number of nodes that will expend is final, meaning the algorithm will terminate in a finite time.

1. Weighted A\*

Weighted A\* is a variant of the A\* search algorithm which multiplies the heuristic estimation with parameter w in the f(n) calculation.

Weighted A\* formula: f(n) = g(n) + w \* h(n)

The weighted A\* is like A\* over the completeness, soundness, and the termination for the same reasons. But is not guaranteed optimality.

* Optimality:

The Weighted A\* algorithm is not necessarily optimal due to the H(n) = w\*h(n) is not always admissible. It depends on weight (w) and the heuristic (h(n)).

If H(n) <= g(n) for all n, then the algorithms remain optimal.

In addition, the ratio between the solution of weighted A\* and the optimal solution is maximum w.

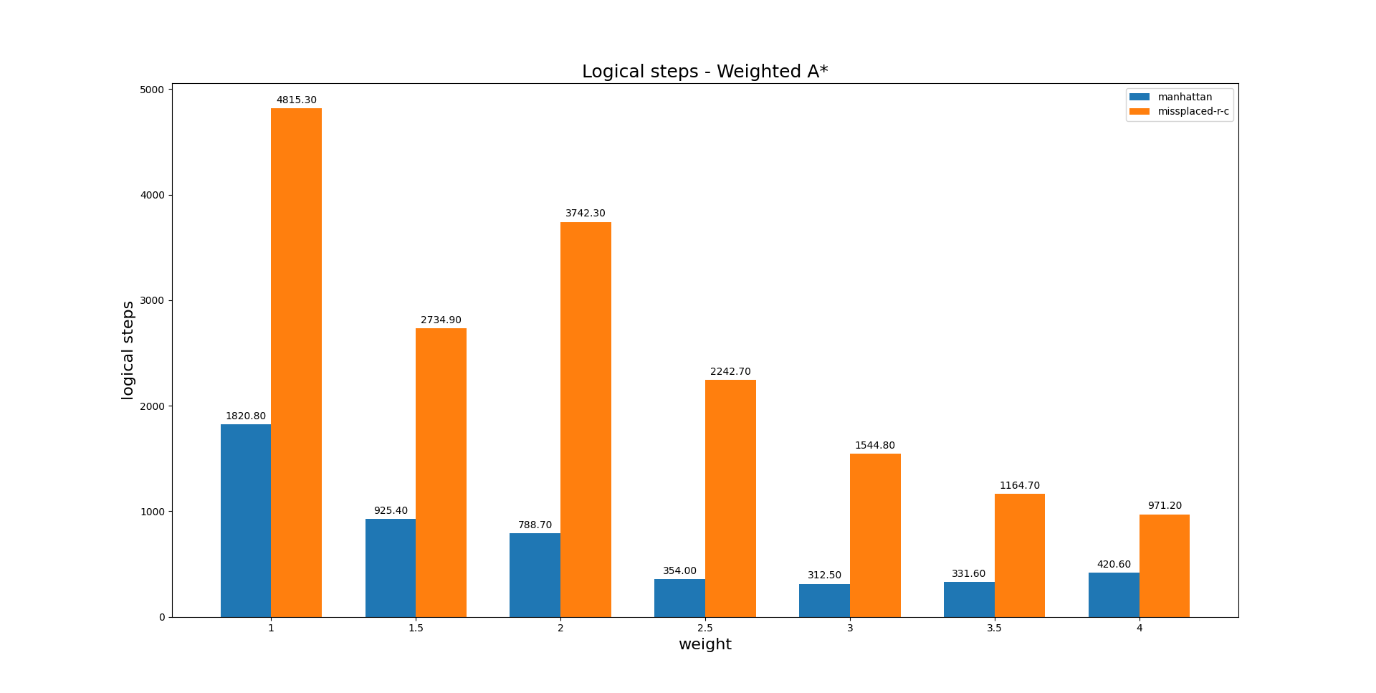
**Graphs:**

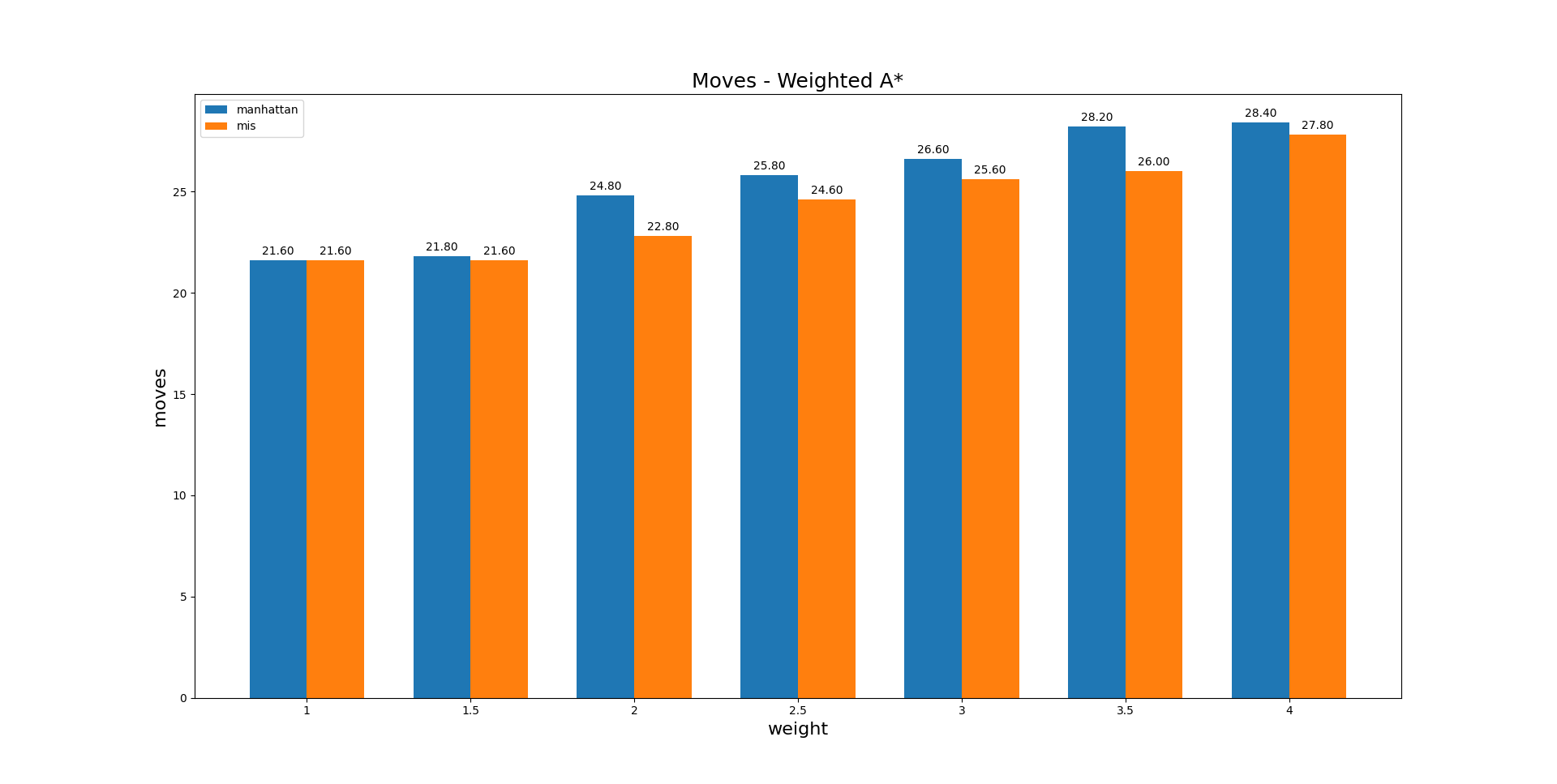
Properties of the runs:

N = 3

Number of problems = 10

w = [1,1.5,2,2.5,3,3.5,4] (w = 1 is regular A\*)



**Appendix:**