Artificial Intelligence: Assignment Report

Nishant Gupta (120101045)

# 15-Puzzle With A\* and IDA\* and Manhattan Distance Heuristic

## Problem Statement

We need to solve [15-puzzle](https://en.wikipedia.org/wiki/15_puzzle) problem using A\* and IDA\* search algorithms, employing [Manhattan distance](https://heuristicswiki.wikispaces.com/Manhattan+Distance) as the guiding heuristic. The corresponding goal state is as follows:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 1 | 2 | 3 |
| 4 | 5 | 6 | 7 |
| 8 | 9 | 10 | 11 |
| 12 | 13 | 14 | 15 |

We are required to generate and solve 20 random instances of the problem, and compare the optimal solution and number of nodes generated. ‘0’ in the following table denotes the blank tile.

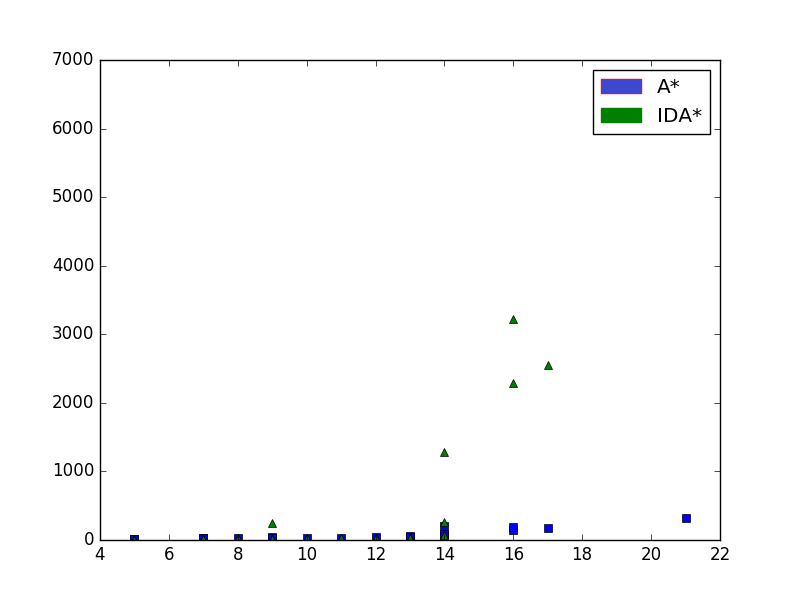
## Algorithms Used

I’ve used standard A\* and IDA\* algorithm, as described in our textbook.

## Results

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No.** | **Initial States** | **A\*** | | **IDA\*** | |
| **opt** | **#nodes** | **opt** | **#nodes** |
| 1 | [[4, 1, 2, 3], [0, 5, 6, 10], [8, 9, 11, 7], [12, 13, 14, 15]] | 9 | 50 | 9 | 250 |
| 2 | [[4, 1, 2, 3], [9, 13, 6, 7], [5, 10, 14, 0], [8, 12, 15, 11]] | 13 | 44 | 13 | 26 |
| 3 | [[1, 5, 2, 3], [0, 6, 10, 7], [4, 9, 14, 11], [8, 12, 13, 15]] | 9 | 29 | 9 | 17 |
| 4 | [[4, 9, 6, 1], [5, 2, 0, 11], [8, 13, 7, 3], [12, 14, 10, 15]] | 21 | 325 | 21 | 6747 |
| 5 | [[1, 5, 0, 7], [4, 6, 3, 2], [8, 9, 10, 11], [12, 13, 14, 15]] | 8 | 35 | 8 | 24 |
| 6 | [[1, 5, 2, 3], [0, 9, 6, 7], [4, 8, 10, 11], [12, 13, 14, 15]] | 5 | 17 | 5 | 9 |
| 7 | [[4, 1, 7, 2], [5, 6, 3, 11], [8, 9, 0, 10], [12, 13, 14, 15]] | 10 | 33 | 10 | 23 |
| 8 | [[4, 1, 2, 3], [8, 5, 7, 15], [12, 10, 6, 0], [13, 9, 11, 14]] | 13 | 58 | 13 | 25 |
| 9 | [[1, 5, 2, 3], [4, 9, 6, 11], [12, 8, 0, 10], [13, 14, 7, 15]] | 14 | 68 | 14 | 265 |
| 10 | [[1, 2, 6, 3], [4, 5, 10, 7], [8, 9, 14, 11], [12, 13, 0, 15]] | 5 | 17 | 5 | 7 |
| 11 | [[1, 2, 3, 10], [4, 5, 7, 0], [8, 9, 6, 15], [12, 13, 11, 14]] | 16 | 148 | 16 | 3219 |
| 12 | [[1, 5, 2, 3], [4, 6, 10, 7], [8, 11, 15, 14], [12, 0, 9, 13]] | 14 | 85 | 14 | 72 |
| 13 | [[4, 1, 3, 7], [9, 6, 2, 11], [5, 8, 10, 15], [12, 13, 14, 0]] | 12 | 40 | 12 | 24 |
| 14 | [[4, 2, 3, 0], [8, 1, 6, 7], [9, 5, 10, 11], [12, 13, 14, 15]] | 7 | 22 | 7 | 12 |
| 15 | [[1, 6, 3, 7], [8, 2, 5, 11], [9, 4, 10, 0], [12, 13, 14, 15]] | 13 | 44 | 13 | 25 |
| 16 | [[1, 5, 6, 3], [8, 0, 4, 7], [9, 2, 10, 11], [12, 13, 14, 15]] | 14 | 199 | 14 | 1282 |
| 17 | [[4, 1, 3, 0], [8, 5, 2, 6], [9, 10, 14, 7], [12, 13, 15, 11]] | 11 | 34 | 11 | 22 |
| 18 | [[0, 1, 2, 3], [4, 5, 6, 14], [8, 9, 7, 10], [12, 13, 15, 11]] | 16 | 190 | 16 | 2292 |
| 19 | [[4, 1, 6, 3], [0, 8, 5, 7], [9, 2, 10, 15], [12, 13, 11, 14]] | 17 | 172 | 17 | 2559 |
| 20 | [[1, 6, 5, 3], [4, 2, 0, 7], [8, 9, 10, 11], [12, 13, 14, 15]] | 7 | 26 | 7 | 13 |

## Plots



## Comments and Observations

A few noticeable points are:

* As we can see, as the number of moves required to solve the puzzle increases, the number of nodes generated by IDA\* rapidly increases in contrast with A\* algorithm.
* However, the memory consumption of A\* algorithm increases rapidly, therefore running out of memory for instances which require large number of moves (comes out to be >= 60 moves in my implementation).
* Although IDA\* is asymptotically comparable to A\*, it takes considerably more time, in practice, for random boards.
* Since Manhattan Distance is an admissible heuristic, both A\* and IDA\* find the optimal result, therefore the solution found by both are equal.

# 15-Puzzle with IDA\* and Recursive Best First Search

## Problem Statement

Next, we need to solve [15-puzzle](https://en.wikipedia.org/wiki/15_puzzle) problem using IDA\* and RBFS search algorithms, employing [Manhattan distance](https://heuristicswiki.wikispaces.com/Manhattan+Distance) as the guiding heuristic. The corresponding goal state is as follows:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 1 | 2 | 3 |
| 4 | 5 | 6 | 7 |
| 8 | 9 | 10 | 11 |
| 12 | 13 | 14 | 15 |

We are required to generate and solve 20 random instances of the problem, and compare the optimal solution and number of nodes generated. ‘0’ in the following table denotes the blank tile.

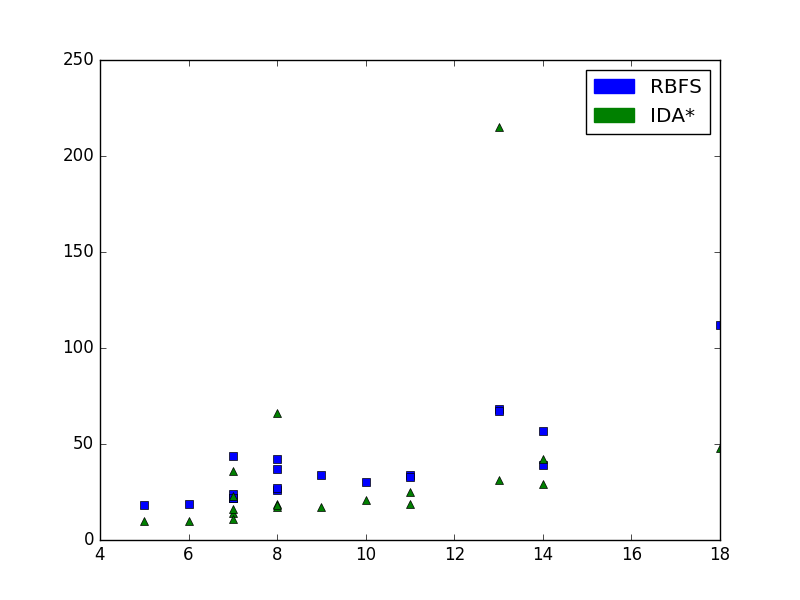
## Algorithms Used

I’ve used standard IDA\* and RBFS algorithms, as described in our textbook.

## Results

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No.** | **Initial States** | **RBFS** | | **IDA\*** | |
| **opt** | **#nodes** | **opt** | **#nodes** |
| 1 | [[1, 0, 2, 3], [8, 5, 6, 7], [9, 4, 10, 11], [12, 13, 14, 15]] | 7 | 23 | 7 | 44 |
| 2 | [[1, 2, 6, 3], [4, 5, 10, 7], [8, 9, 11, 15], [12, 13, 14, 0]] | 6 | 10 | 6 | 19 |
| 3 | [[1, 2, 3, 7], [4, 9, 5, 6], [0, 8, 10, 11], [12, 13, 14, 15]] | 8 | 19 | 8 | 26 |
| 4 | [[1, 6, 5, 3], [4, 2, 10, 7], [8, 9, 11, 0], [12, 13, 14, 15]] | 9 | 17 | 9 | 34 |
| 5 | [[4, 2, 7, 6], [5, 0, 3, 11], [8, 1, 9, 10], [12, 13, 14, 15]] | 14 | 29 | 14 | 57 |
| 6 | [[1, 2, 6, 3], [4, 7, 13, 11], [12, 5, 0, 10], [14, 8, 9, 15]] | 18 | 48 | 18 | 112 |
| 7 | [[1, 5, 2, 3], [4, 6, 10, 7], [8, 9, 11, 0], [12, 13, 14, 15]] | 5 | 10 | 5 | 18 |
| 8 | [[4, 2, 3, 7], [5, 1, 0, 6], [8, 9, 10, 11], [12, 13, 14, 15]] | 7 | 14 | 7 | 22 |
| 9 | [[1, 0, 3, 7], [4, 2, 5, 6], [8, 9, 10, 11], [12, 13, 14, 15]] | 7 | 16 | 7 | 22 |
| 10 | [[1, 5, 2, 3], [4, 9, 11, 6], [8, 10, 0, 7], [12, 13, 14, 15]] | 8 | 17 | 8 | 42 |
| 11 | [[4, 1, 2, 3], [5, 0, 6, 7], [8, 13, 10, 11], [9, 12, 14, 15]] | 8 | 66 | 8 | 37 |
| 12 | [[1, 0, 2, 3], [4, 8, 6, 7], [9, 5, 10, 11], [12, 13, 14, 15]] | 7 | 36 | 7 | 24 |
| 13 | [[1, 2, 3, 7], [4, 5, 6, 11], [0, 9, 15, 14], [8, 12, 13, 10]] | 14 | 42 | 14 | 39 |
| 14 | [[1, 9, 5, 3], [4, 2, 0, 6], [8, 10, 14, 7], [12, 13, 15, 11]] | 13 | 31 | 13 | 68 |
| 15 | [[4, 1, 2, 3], [8, 0, 5, 6], [12, 9, 10, 7], [13, 14, 15, 11]] | 10 | 21 | 10 | 30 |
| 16 | [[4, 1, 2, 3], [5, 12, 6, 7], [8, 10, 14, 0], [13, 9, 15, 11]] | 13 | 215 | 13 | 67 |
| 17 | [[1, 5, 2, 3], [4, 6, 7, 11], [8, 9, 14, 10], [12, 0, 13, 15]] | 8 | 18 | 8 | 27 |
| 18 | [[4, 1, 6, 2], [5, 9, 0, 3], [8, 13, 10, 7], [12, 14, 15, 11]] | 11 | 19 | 11 | 34 |
| 19 | [[4, 1, 2, 3], [8, 5, 6, 7], [13, 12, 10, 0], [9, 14, 15, 11]] | 11 | 25 | 11 | 33 |
| 20 | [[1, 5, 2, 3], [0, 9, 6, 7], [4, 13, 10, 11], [8, 12, 14, 15]] | 7 | 11 | 7 | 22 |

## Plot (Optimal Solution vs Number of Nodes Generated)



## Comments and Observations

A few noticeable points are:

* We cannot observe a major difference in number of nodes generated by IDA\* and RBFS, however, number of nodes generated by RBFS is higher.
* Since, our heuristic is admissible, both of the algorithms find the optimal solution.
* Since both the algorithms expand similar number of nodes, they take similar amount of time.

## Bonus Problem (Weighted Heuristic)

I employed as a heuristic for both the problems, where heuristic value of the corresponding board. I ran both the algorithms using the heuristic, and while both the algorithm found the optimal result, the number of nodes expanded by IDA\* is substantially higher. For example, on the board

[[1, 2, 3, 7], [4, 5, 6, 11], [9, 10, 15, 14], [8, 12, 13, 0]]

**RBFS expands 42 nodes** to find the solution, while **IDA\* expands 27809 nodes** to find the solution. The weighted heuristic is labelled as the function *bad\_h* in my code.

# 8-Queens With Random Restart Hill Climbing and Simulated Annealing

## Problem Statement

We need to solve [8-Queens](https://en.wikipedia.org/wiki/Eight_queens_puzzle) problem using Random restart hill climbing and simulated annealing algorithms, employing number of attacking queens as the guiding heuristic.



Figure 1: 8 Queens. Image Courtesy: Wikipedia

We are required to generate and solve 20 random instances of the problem, and compare the results obtained by both the algorithms.

## Algorithms Used

I’ve implemented the standard random restart hill climbing. The default number of iterations for my implementation is 1000.

For, simulated annealing, I run 5000 iterations with my cooling schedule as where k = current iteration number. To smoothen the schedule, after 4000th iteration, my cooling schedule always returns 0 as the temperature.

## Results

I’ve represented initial state as a list of integers, which denote the row number of the queen in the corresponding column. For example, the first entry denotes the row number of the queen in the first column. The table denotes number of restarts taken by Random Restart Hill Climbing and heuristic value of the final state returned by simulated annealing algorithm. The accuracy for Random Restart Hill Climbing came out to be 100%, while for Simulated Annealing came out to be 95%

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | Initial State | Random Restart Hill Climbing | | Simulated Annealing | |
| Final State | Number of Restarts | Final State | Final number of attacking pairs |
| 1 | [7, 6, 6, 1, 6, 4, 6, 2] | [3, 0, 4, 7, 5, 2, 6, 1] | 7 | [2, 7, 3, 6, 0, 5, 1, 4] | 0 |
| 2 | [5, 6, 2, 7, 4, 4, 0, 7] | [3, 6, 2, 7, 1, 4, 0, 5] | 0 | [3, 6, 2, 7, 1, 4, 0, 5] | 0 |
| 3 | [3, 0, 7, 2, 2, 3, 4, 2] | [5, 0, 4, 1, 7, 2, 6, 3] | 29 | [5, 2, 4, 7, 0, 3, 1, 6] | 0 |
| 4 | [1, 2, 3, 5, 5, 3, 3, 4] | [6, 4, 2, 0, 5, 7, 1, 3] | 14 | [4, 0, 7, 5, 2, 6, 1, 3] | 0 |
| 5 | [2, 4, 5, 2, 4, 7, 0, 1] | [4, 2, 0, 6, 1, 7, 5, 3] | 2 | [4, 0, 7, 3, 1, 6, 2, 5] | 0 |
| 6 | [1, 7, 5, 6, 3, 5, 5, 1] | [1, 5, 0, 6, 3, 7, 2, 4] | 7 | [6, 2, 7, 1, 4, 0, 5, 3] | 0 |
| 7 | [3, 3, 1, 7, 5, 1, 3, 4] | [2, 5, 7, 0, 3, 6, 4, 1] | 7 | [5, 7, 1, 3, 0, 6, 4, 2] | 0 |
| 8 | [3, 3, 7, 7, 6, 5, 1, 6] | [3, 1, 7, 5, 0, 2, 4, 6] | 5 | [1, 4, 6, 3, 0, 7, 5, 2] | 0 |
| 9 | [5, 1, 7, 5, 7, 2, 5, 3] | [1, 3, 5, 7, 2, 0, 6, 4] | 21 | [3, 1, 7, 5, 0, 2, 4, 6] | 0 |
| 10 | [6, 1, 0, 7, 5, 0, 7, 6] | [3, 1, 4, 7, 5, 0, 2, 6] | 0 | [5, 1, 6, 0, 3, 7, 4, 2] | 0 |
| 11 | [7, 7, 2, 0, 2, 1, 0, 6] | [3, 7, 0, 2, 5, 1, 6, 4] | 9 | [3, 1, 6, 2, 5, 7, 0, 4] | 0 |
| 12 | [0, 5, 6, 0, 2, 5, 5, 7] | [2, 5, 1, 6, 0, 3, 7, 4] | 4 | [5, 1, 6, 0, 2, 4, 7, 3] | 0 |
| 13 | [6, 2, 5, 2, 1, 4, 3, 0] | [4, 1, 5, 0, 6, 3, 7, 2] | 4 | [3, 7, 4, 2, 0, 6, 1, 5] | 0 |
| 14 | [7, 6, 7, 2, 2, 0, 7, 6] | [4, 1, 5, 0, 6, 3, 7, 2] | 13 | [3, 6, 4, 2, 0, 5, 7, 1] | 0 |
| 15 | [0, 5, 7, 1, 0, 0, 3, 4] | [3, 5, 7, 1, 6, 0, 2, 4] | 0 | [2, 4, 1, 7, 0, 6, 3, 5] | 0 |
| 16 | [0, 0, 1, 3, 3, 5, 7, 5] | [1, 5, 0, 6, 3, 7, 2, 4] | 10 | [4, 7, 3, 0, 6, 1, 5, 2] | 0 |
| 17 | [7, 2, 7, 7, 3, 6, 0, 5] | [4, 1, 3, 6, 2, 7, 5, 0] | 3 | [4, 2, 7, 5, 1, 5, 0, 6] | 1 |
| 18 | [6, 0, 4, 6, 1, 6, 7, 0] | [2, 5, 1, 6, 0, 3, 7, 4] | 1 | [4, 0, 3, 5, 7, 1, 6, 2] | 0 |
| 19 | [2, 7, 2, 2, 6, 6, 5, 2] | [3, 7, 0, 4, 6, 1, 5, 2] | 0 | [1, 6, 4, 7, 0, 3, 5, 2] | 0 |
| 20 | [6, 1, 6, 3, 1, 1, 5, 5] | [2, 5, 7, 0, 3, 6, 4, 1] | 12 | [4, 1, 7, 0, 3, 6, 2, 5] | 0 |

## Plots

# 8-Queens With Genetic Algorithm

## Problem Statement

We have to solve 8-Queens problem using genetic algorithm with single point cross over, roulette wheel selection and very small mutation probability.

## Algorithms Used

Genetic Algorithm with single point cross over, roulette wheel selection and very small mutation probability, as described in our textbook. Refer to genetic\_algorithm function in the file a4\_8queens.py

## Comments and Observations

* As the initial population increases, the algorithm performs better, i.e. it reaches to better heuristic values (low number of attacking queens) after a fixed number of generations (500 by default in my implementation).
* However, since our crossover and selection strategy involve lot of randomness, hence it takes a lot of generations to find the optimal solution.
* Mutation helps us overcome many local maxima. However, increasing the mutation probability to a high value also cripples the better parts of the individuals, therefore performing poorly.
* The advantage comes from the ability of crossover to combine large blocks of letters that have evolved independently to perform useful functions, thus raising the level of granularity at which the search operates.