**15-Puzzle:**

**Genetic && Wisdom of the Crowds**

Samuel Nwosu

CECS

Speed School of Engineering

University of Louisville, USA

[scnwos01@louisville.edu](mailto:scnwos01@louisville.edu)

For this project we the students got to choose an np- complete problem to solve using Genetic algorithms and wisdom of Crowds. For this project I choose to tackle the problem of trying to find a minimum solution to the 15-Puzzle. The 15-puzzle is a sliding game that consists of a frame of numbered square tiles in random order with one tile missing and the object of the game is to place the tiles in order using the empty space. Now finding a solution to a 15-puzzle is a problem that is of class P, but finding a minimal solution is NP-Hard( a fact I realized after writing most of my code). Essentially, I am trying to find a solution to the 15-puzzle using a few moves as possible, a task that turned out to be a lot harder than originally expected something I’ll go further into detail later in the report.

My project starts off by generating an already solved 15-puzzle and then randomly shuffling it with 1000 moves. Shuffling a solved 15-puzzle with valid moves insure we end up with a valid 15-puzzle that can be solved. I then grab the population size from the user and generate the initial population. Each member of the population contains a list of moves, a fitness value, and a 15-puzzle with the members moves applied. I then set up a while loop that repeats until the max length of moves is down to two. In this loop I try to find the best solution for each max size of moves starting with 80, this is done by applying a genetic algorithm and using WoC to generate with better fitness values. If I go for too long without seeing an improvement in fitness, I decrement the max length of moves and generate a new population and try to find a better fitness with fewer moves. This is done until the max length of moves is less than two and the final results are printed to the screen.

The reason I chose to start with 80 as the max length of moves is because in the paper “*The parallel search bench ZRAM and its applications”* it was shown that the lengths of optimal solutions ranged from 0 to 80. I chose to keep my lower bounds to 2 in order to keep my array logic sound.



The first genetic algorithm implemented my project was Single Crossover. For this algorithm I selected a random index within a given path and used it a swapping point. The elements from the first parent would fill the new path up until that point and then the elements of the second parent would fill the rest. The other implemented genetic algorithm was a double cross over, where to random indexes were selected as cross over point. Each method applied a mutation the children they produce, by swapping two random elements within the child move list.



Figure Single Crossover

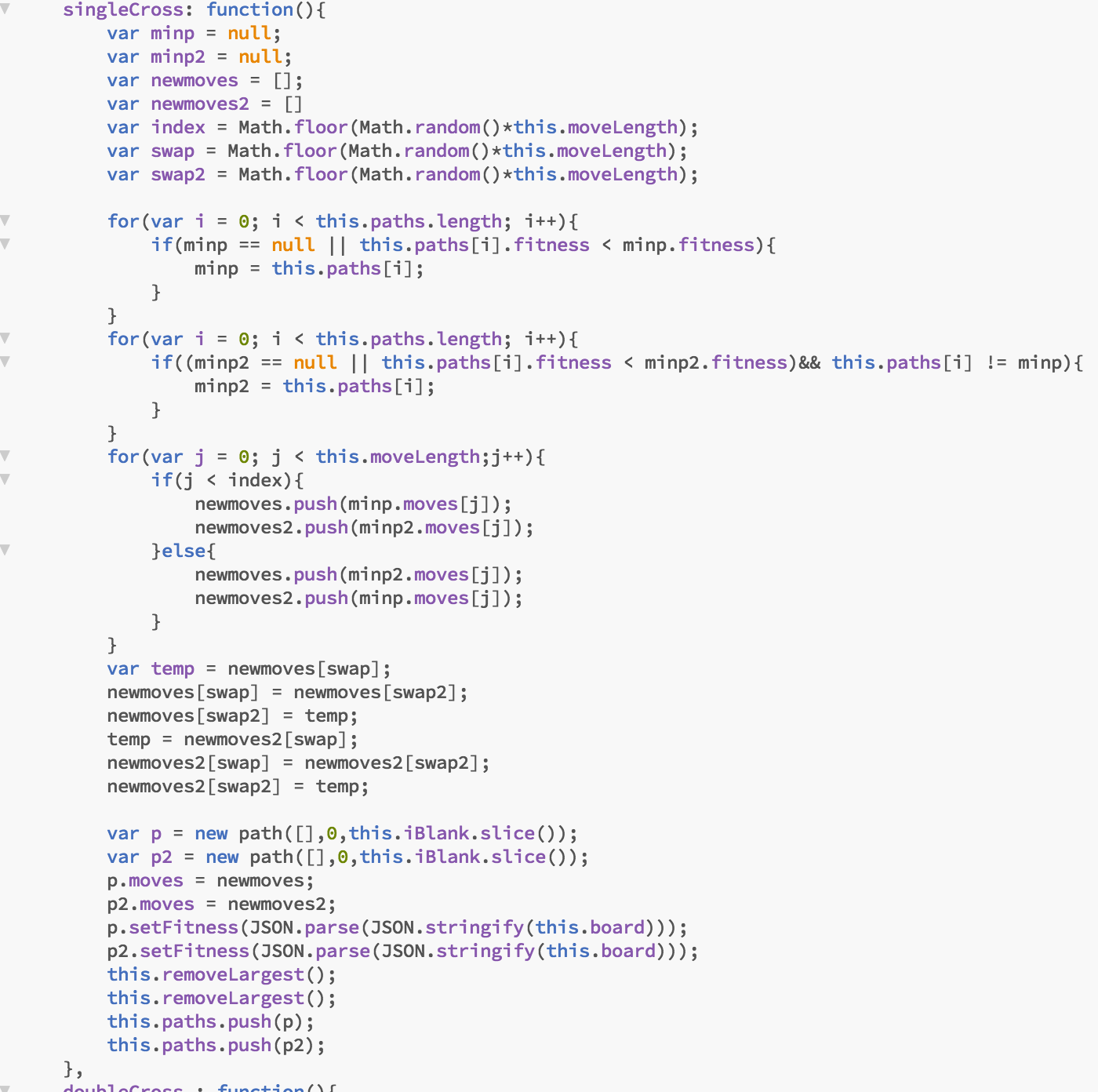


Figure Double cross over

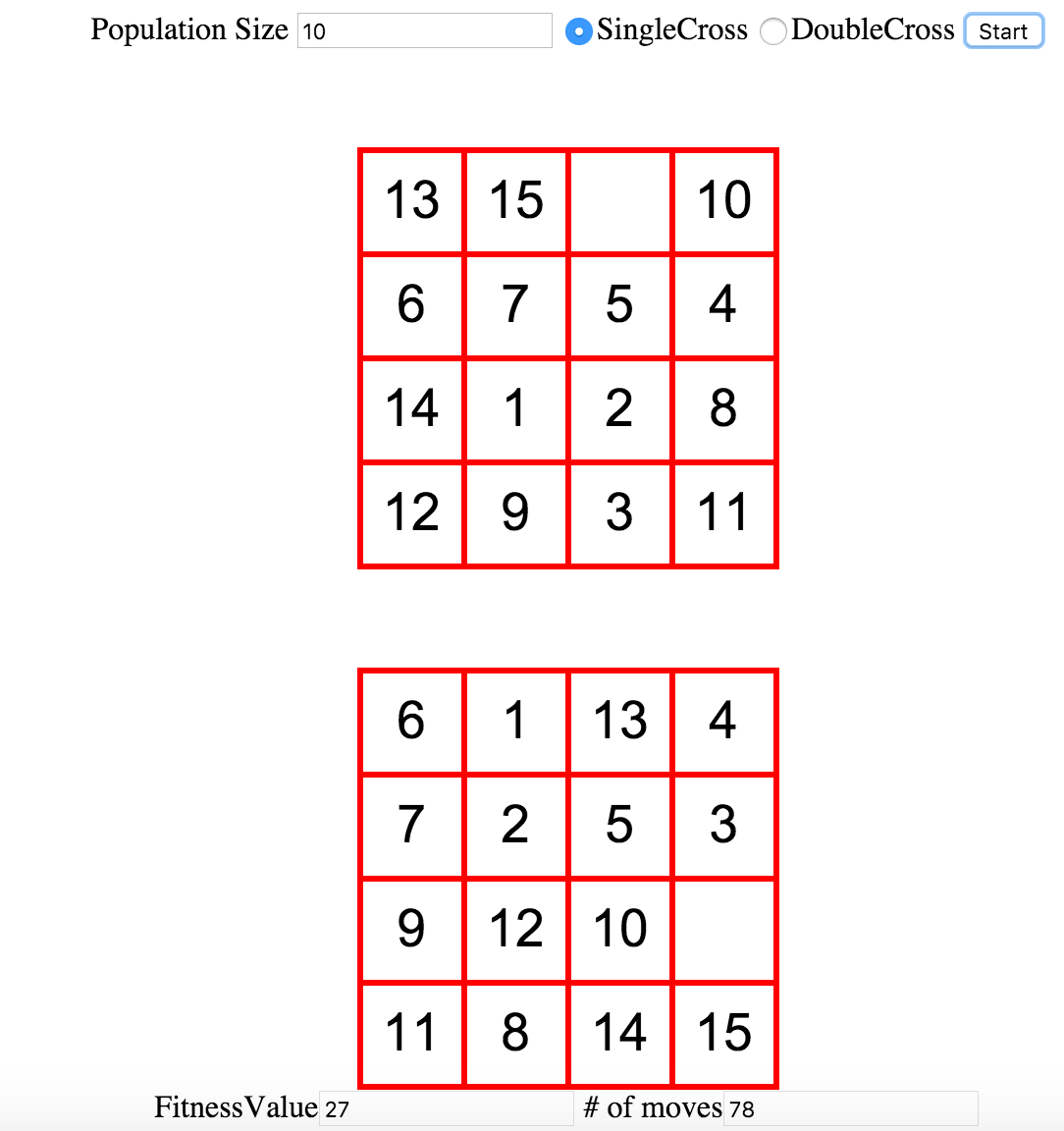
I had to do a bit of thinking to figure out a way to create a new child using wisdom of crowds. I ended up making it so that each move in the child move list was the most frequent move at that time. For example, if the most frequent first move was “Up”, the first move of the child would be up and so on. To do this, I create a list of the top ten members with the smallest fitness values and created a frequency mapping for each moves. Afterwards I crated the new child with the moves with the highest frequency.



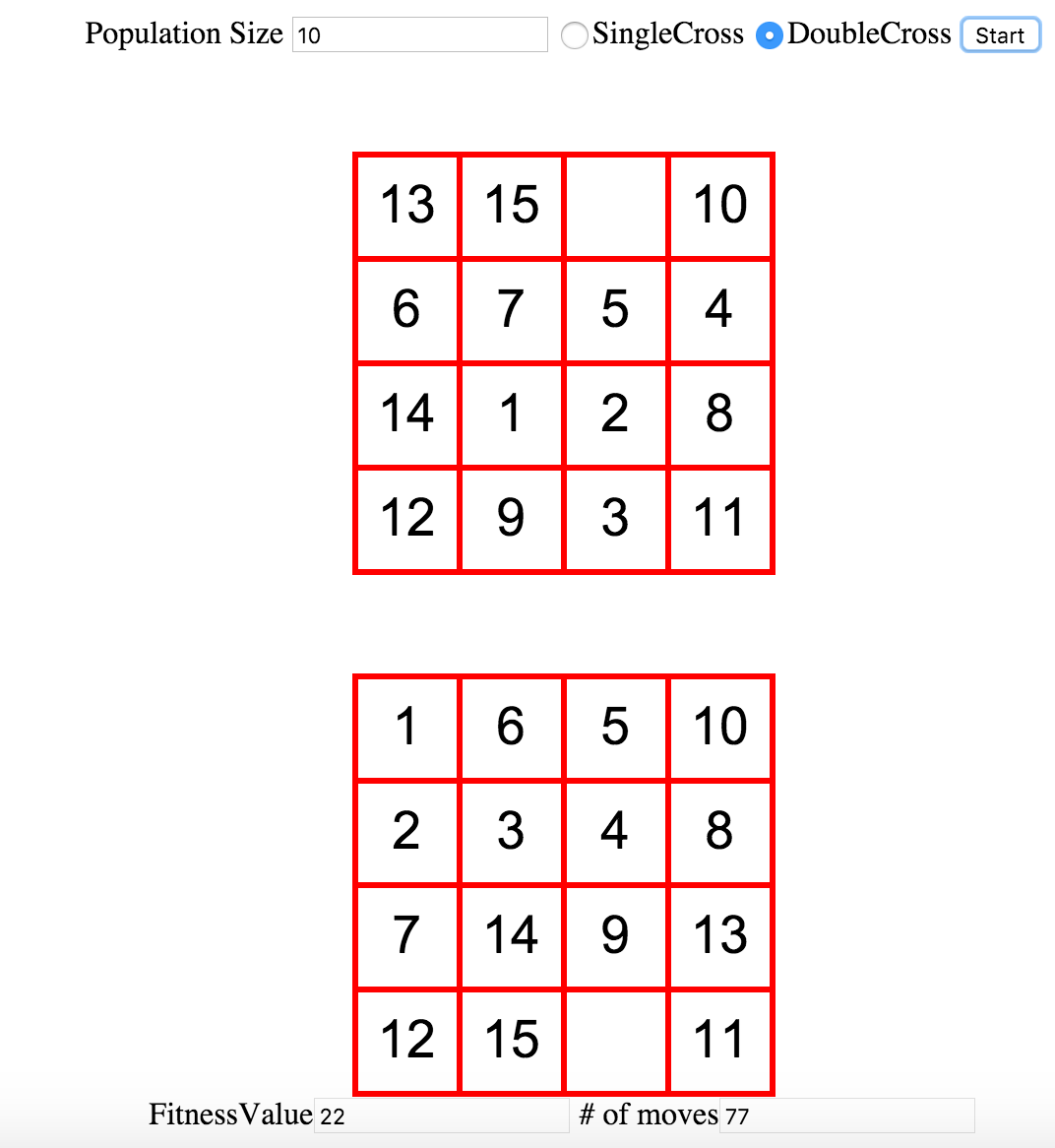
Results:

**Population = 10**

**SingleCross(WoC)**

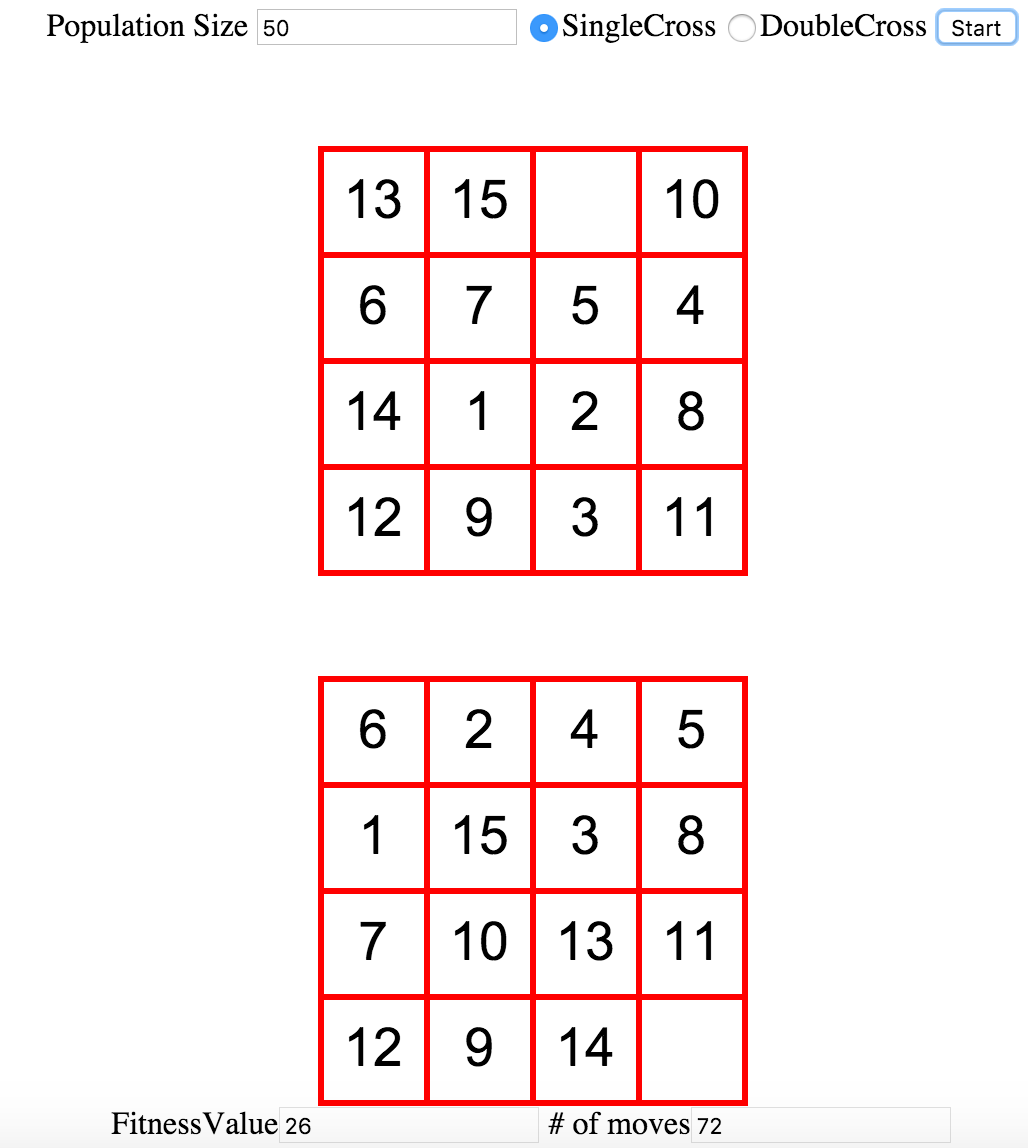


**DoubleCross(WoC)**

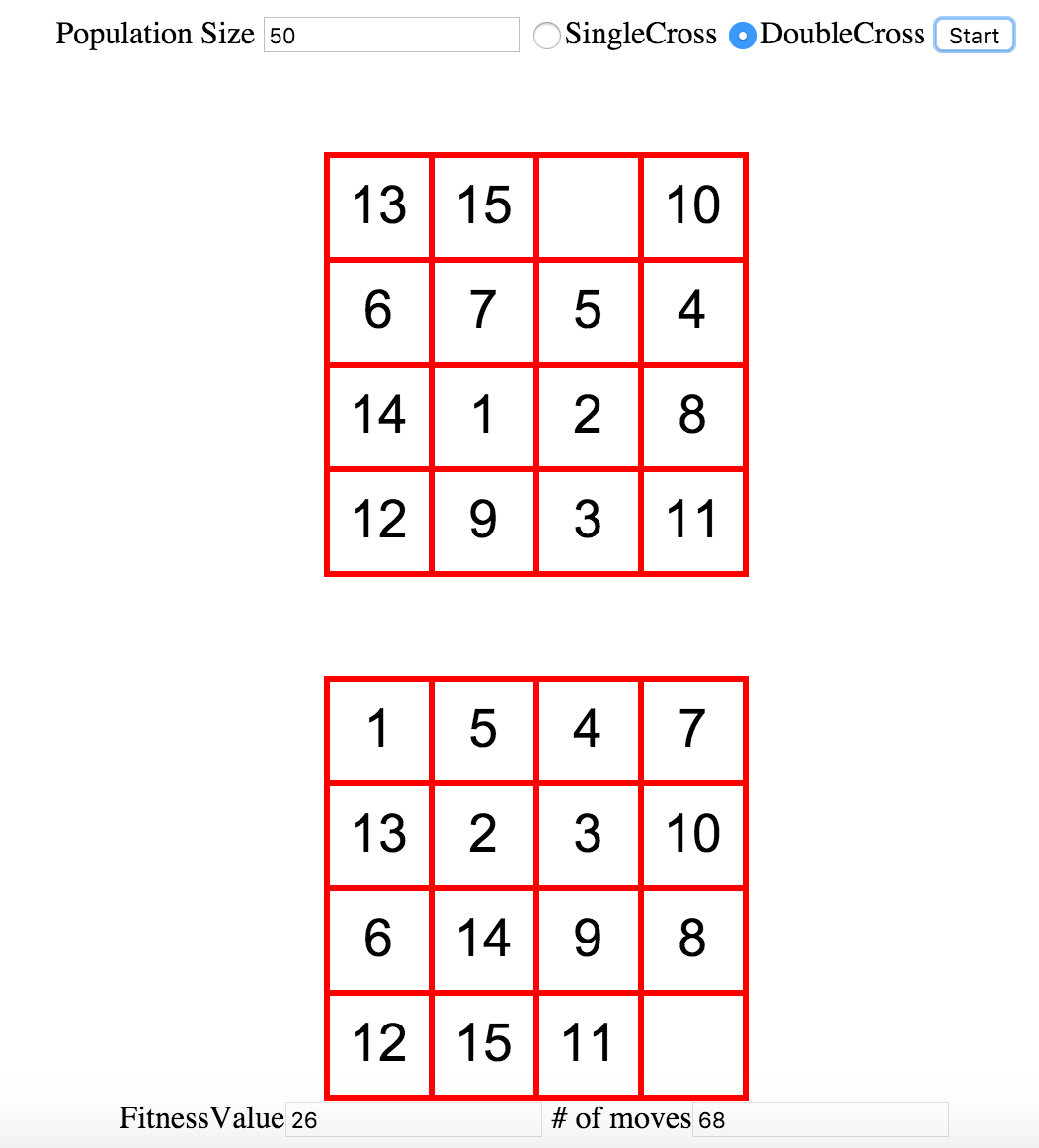


**Population = 50**

**SingleCross(WoC)**

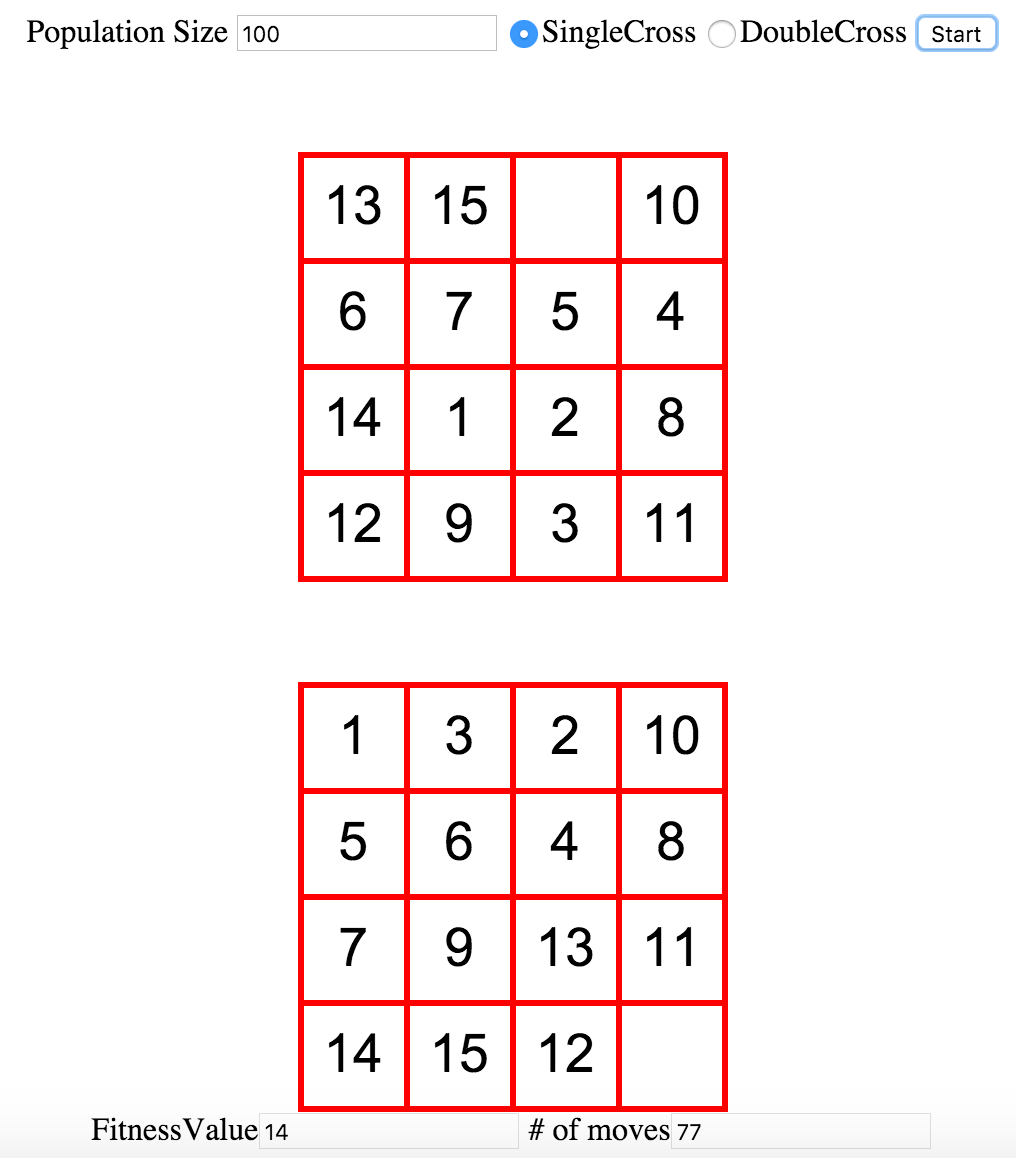


**DoubleCross(WoC)**

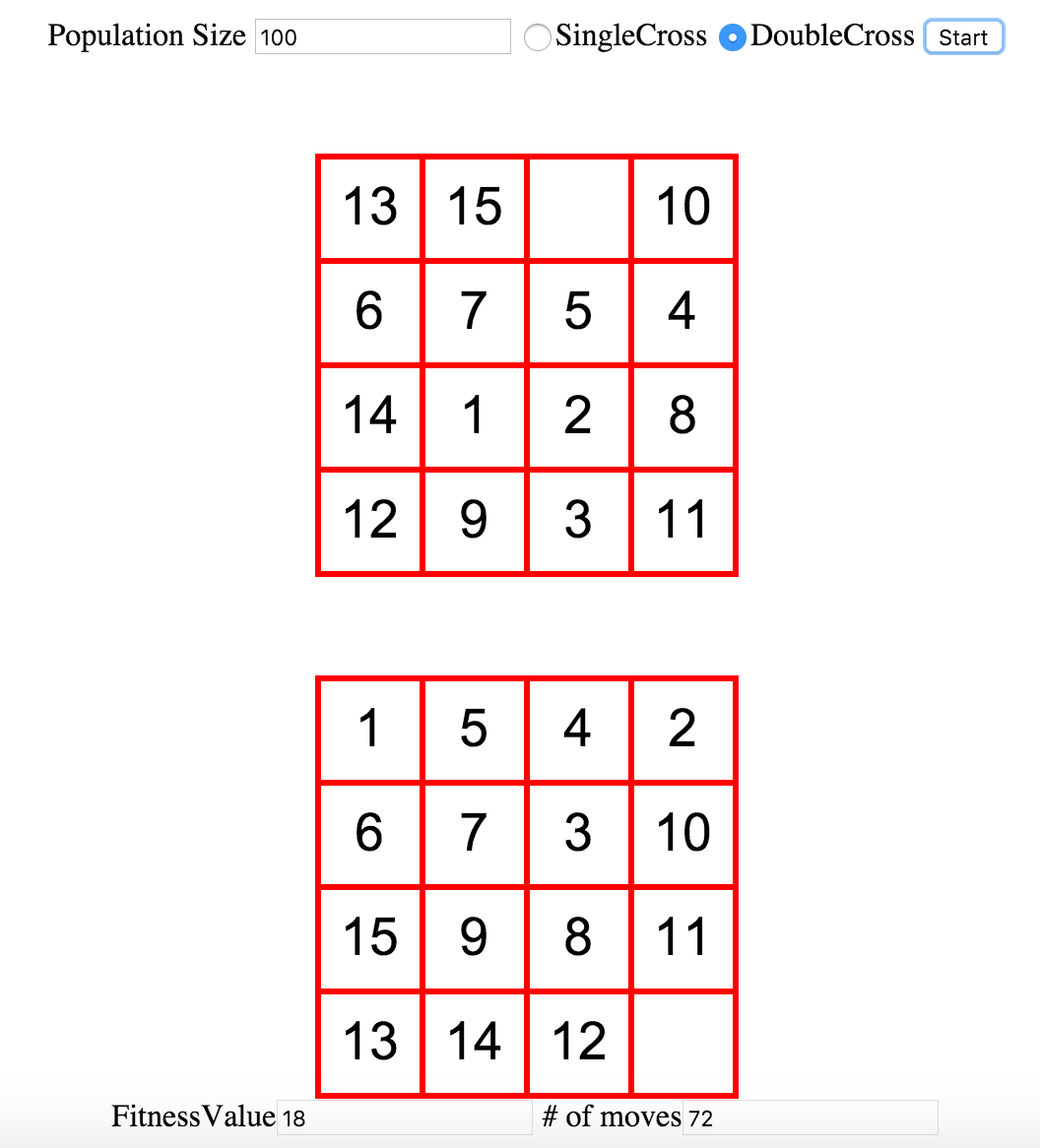


**Population = 100**

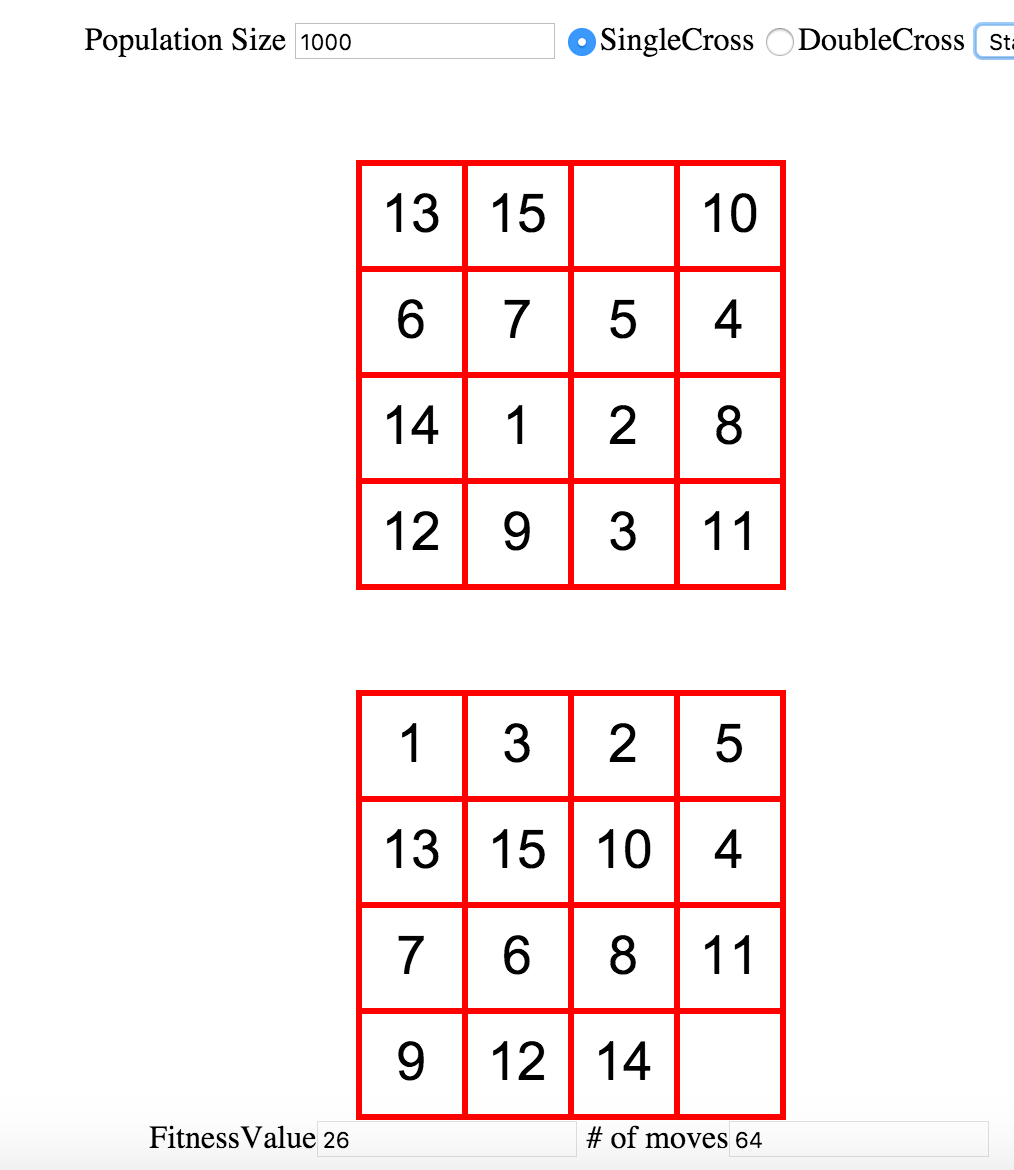
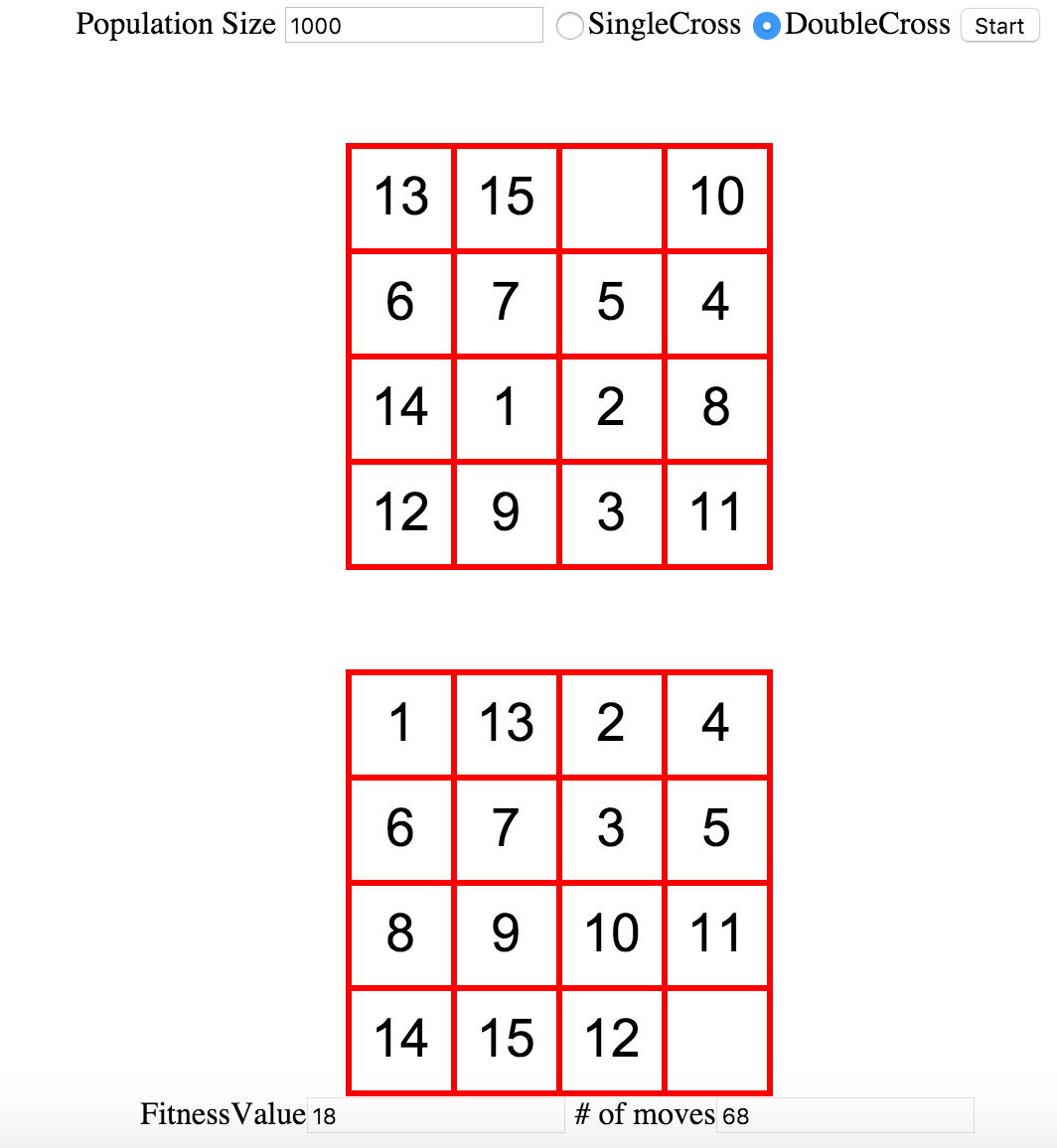
**SingleCross(WoC)**

****

**DoubleCross(WoC)**

****

**Population = 1000**

**SingleCross(WoC)****DoubleCross(WoC)**

In exception of a few outliers, the results seem to improve as we increase the population’s size with the best results coming from the algorithms with a population of size 1000. Also the DoubleCross genetic algorithm with wisdom of crowds seems to perform better regardless of population size.

This has by far been on of the more interesting projects I have had to work on for this class. This project really forced me to think out of the box when it came to finding a problem to tackle and implementing a decent way to solve that problem. Though I am happy with the current project there are places where it can be improved. In the future, I would update my project so that at each level it would find a solution that fixed the 15-puzzle if it exists. I would also try to expand the project so it covers and N-puzzle.