

Assignment 1

Kyle Reagle and Yigit Gungor

September 26, 2017

Important Notes:

After documenting everything, we noticed that the GUI displays the wrong puzzle (in red), but it displays the shortest path to every cell correctly (in blue), as well as the correct evaluation function. This is an issue for task 3, 4, and 5 because they all use the same hill climbing function, but we are unsure how to fix this issue. Task 6 and 7 seem to display the correct puzzles. Despite this issue, the plots in this document show that the algorithms work as intended. We also realized that BFS only needed to be called once per iteration, so run times should be at least n times faster than this document shows.

Task 2

Figure 1: Solvable and unsolvable 5x5 puzzle

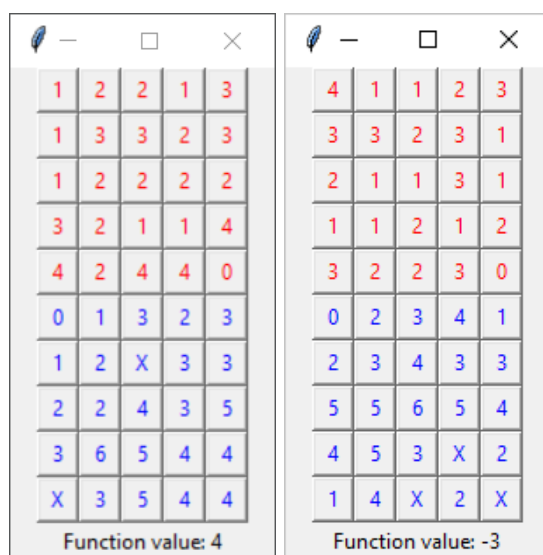



Figure 2: Solvable and unsolvable 7x7 puzzle

	—	□	×
2	5	3	6
5	2	3	5
1	5	4	2
6	5	3	1
2	4	4	2
3	5	1	2
2	2	6	5
0	7	1	3
2	X	5	5
1	2	4	5
2	X	2	4
4	6	5	5
5	7	X	6
3	X	3	4
Function value: 5			



	—	□	×
5	3	5	2
1	1	3	2
2	3	2	1
1	2	3	3
2	5	4	2
5	3	5	3
2	4	3	6
0	X	X	X
X	X	X	X
X	X	X	X
X	X	X	X
X	X	X	X
1	X	X	X
X	X	X	X
Function value: -45			

Figure 3: Solvable and unsolvable 9x9 puzzle

 tk	—	□	×
7	7	7	1
3	6	3	5
6	7	5	2
6	2	2	4
1	1	6	4
4	4	5	2
1	2	2	6
6	4	7	5
7	6	8	4
0	X	5	5
2	6	X	3
X	6	5	5
4	5	X	6
3	4	5	5
4	5	4	X
7	8	X	4
1	7	6	4
X	9	X	4
Function value: 7			


 tk	—	□	×
6	1	4	4
3	7	5	2
7	1	4	1
2	3	5	4
4	3	3	2
6	4	1	3
5	3	4	3
5	4	7	4
3	4	3	4
0	6	6	X
2	4	6	3
X	X	8	7
7	5	8	4
3	6	7	6
5	4	7	8
1	6	7	7
X	4	8	5
4	5	6	5
Function value: -11			

Figure 4: Solvable and unsolvable 11x11 puzzle

tk	—	□	×
5	7	3	6
9	9	1	8
7	8	4	1
4	4	6	1
3	2	7	5
8	2	2	1
2	1	4	6
9	5	5	4
2	6	1	3
6	9	3	8
10	7	4	1
0	9	X	5
6	X	X	6
6	4	5	5
7	5	5	4
5	X	X	5
1	4	4	5
4	3	4	4
6	4	5	X
5	7	6	5
7	8	6	5
6	5	5	X
Function value: 5			

tk	—	□	×
2	3	10	9
1	1	5	1
9	1	7	2
10	6	8	4
3	7	5	7
6	6	1	6
10	5	2	4
8	8	8	6
6	4	8	5
10	9	8	2
7	7	10	2
0	5	1	10
6	5	5	13
1	6	3	9
9	6	12	11
7	12	7	8
8	10	9	10
7	X	6	8
8	X	X	12
X	11	7	10
X	4	4	11
12	10	2	9
Function value: -14			

Task 3

Figure 5: 5x5 puzzle hill climbing results at 100, 1000, and 10000 iterations. Run times were 0.2, 2.1, and 20.4 seconds respectively

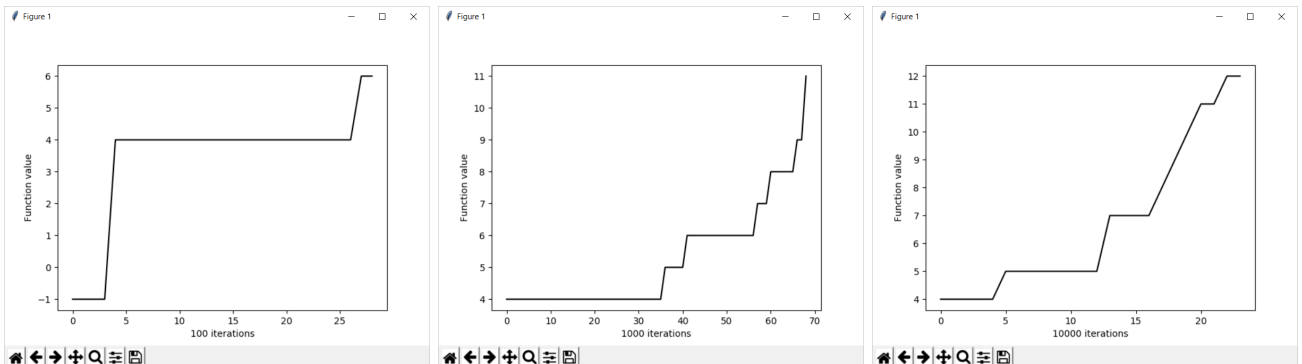


Figure 6: 7x7 puzzle hill climbing results at 100, 1000, and 10000 iterations. Run times were 1.1, 9.9, and 96.7 seconds respectively

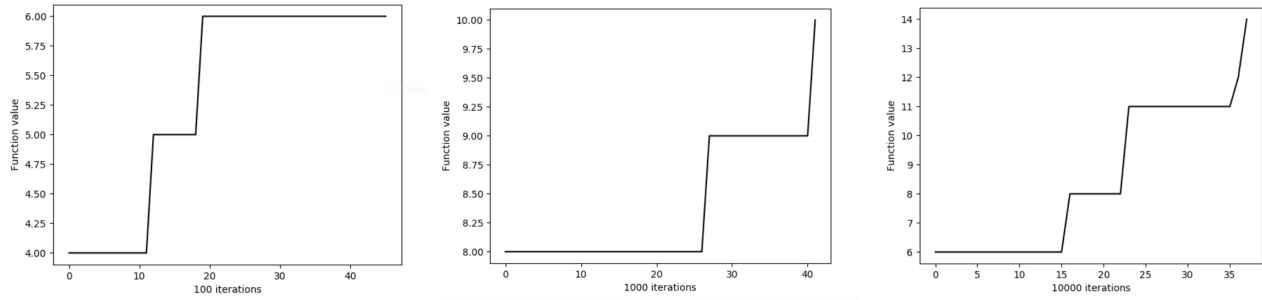


Figure 7: 9x9 puzzle hill climbing results at 100, 500, and 1000 iterations. Run times were 3.2, 17, and 33.4 seconds respectively

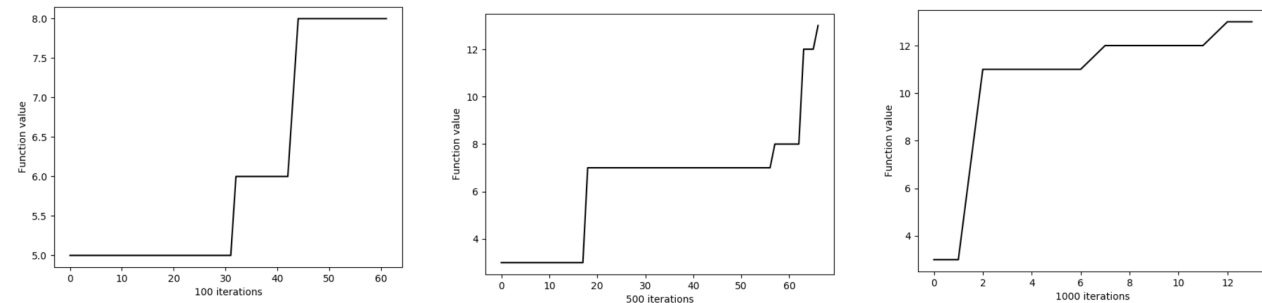
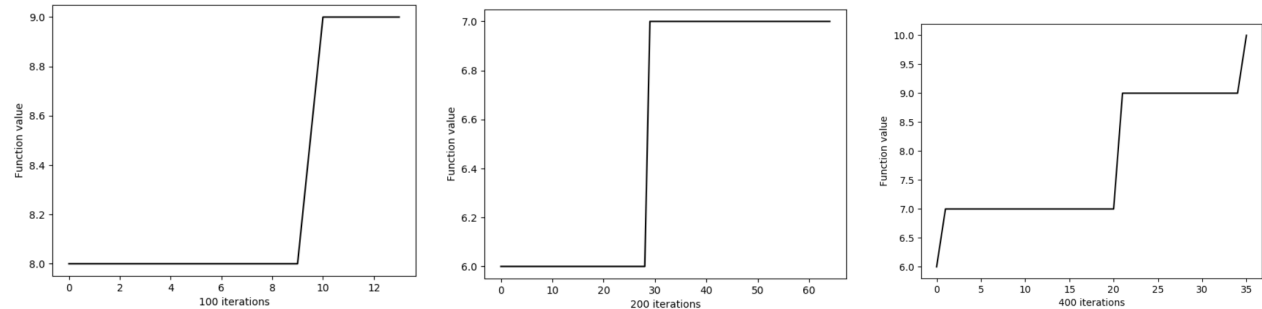


Figure 8: 11x11 puzzle hill climbing results at 100, 200, and 400 iterations. Run times were 8.7, 16.1, and 34.9 seconds respectively



Task 4

Figure 9: 5x5 puzzle hill climbing with random restarts results at 200x5, 2000x5, and 200x50 iterations. Run times were 2.1, 21, and 19.4 seconds respectively

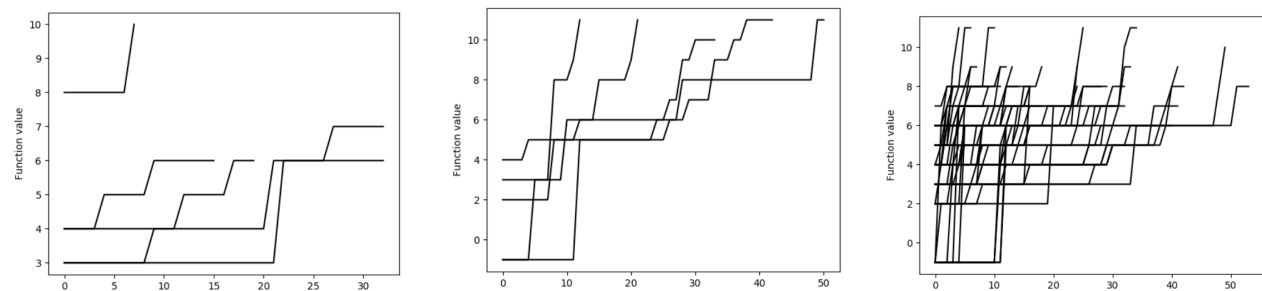


Figure 10: 7x7 puzzle hill climbing with random restarts results at 200x5, 2000x5, and 200x50 iterations. Run times were 9.4, 93.4, and 95.9 seconds respectively

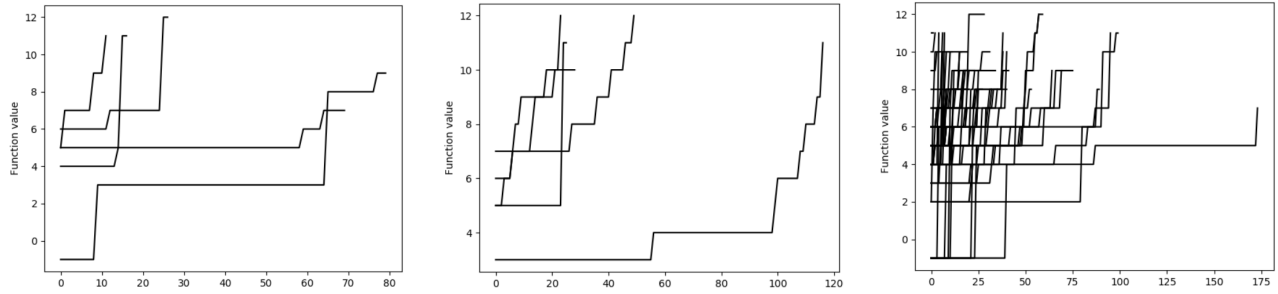


Figure 11: 9x9 puzzle hill climbing with random restarts results at 100x5, 100x10, and 1000x3 iterations. Run times were 14.6, 32.2, and 92.6 seconds respectively

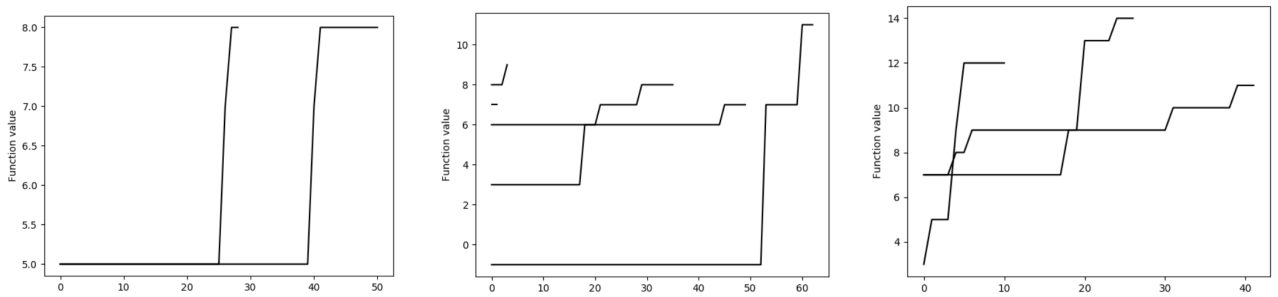
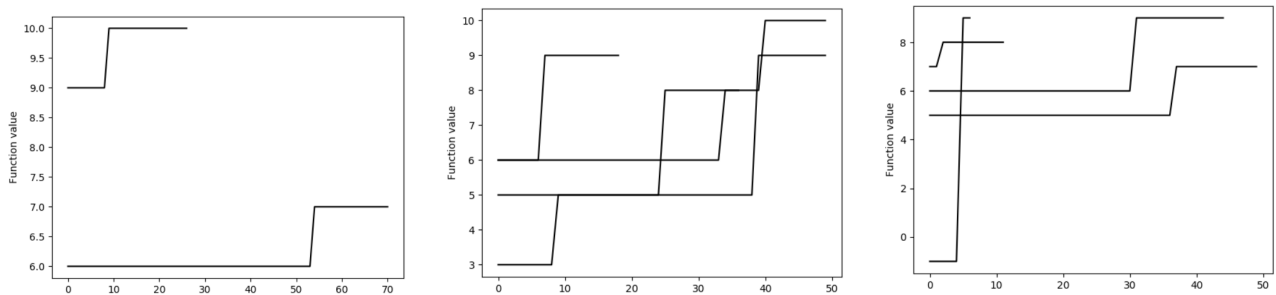


Figure 12: 11x11 puzzle hill climbing with random restarts results at 100x2, 50x4, and 100x4 iterations. Run times were 18.8, 17.2, and 32.8 seconds respectively



Task 5

Figure 13: 5x5 puzzle hill climbing with random walk results at 5000 iterations. Probability of accepting a downhill move was 0.2%, 1%, and 10% respectively. Run times were all about 10 seconds.

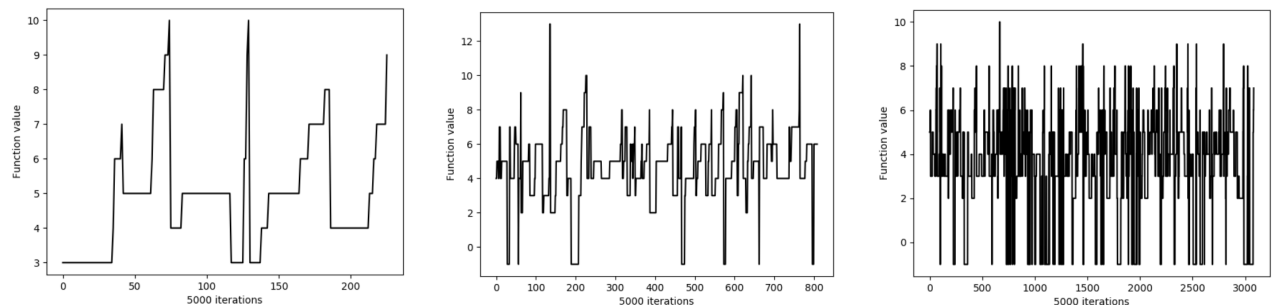


Figure 14: 7x7 puzzle hill climbing with random walk results at 1000 iterations. Probability of accepting a downhill move was 1%, 5%, and 10% respectively. Run times were all about 9 seconds.

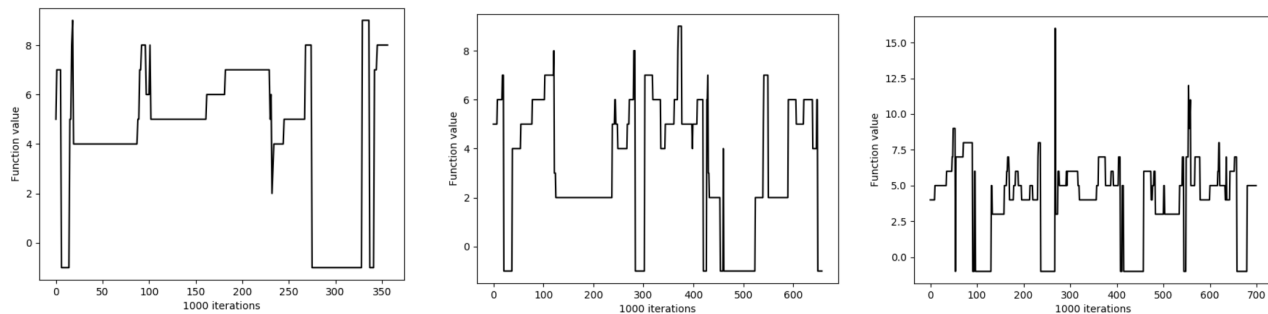


Figure 15: 9x9 puzzle hill climbing with random walk results at 1000 iterations. Probability of accepting a downhill move was 1%, 5%, and 10% respectively. Run times were all about 30 seconds.

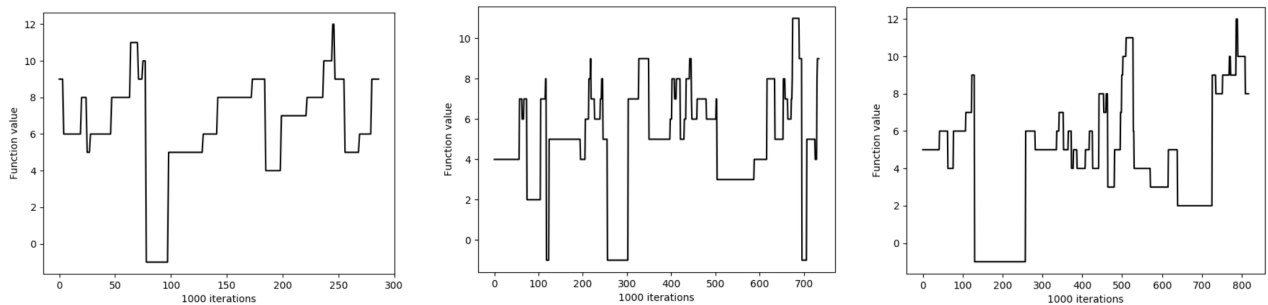
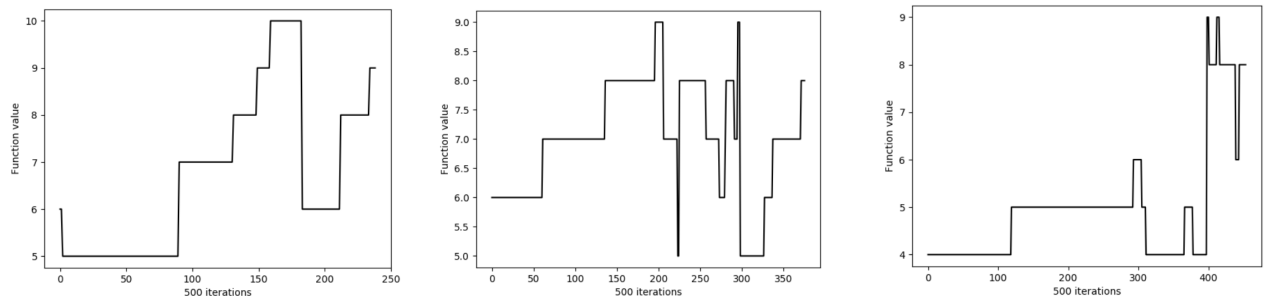


Figure 16: 11x11 puzzle hill climbing with random walk results at 500 iterations. Probability of accepting a downhill move was 1%, 5%, and 10% respectively. Run times were all about 45 seconds.



Task 6

Figure 17: 5x5 puzzle simulated annealing results at 5000 iterations. Initial temperatures were 10000, 1000, and 100, and decay rates were 0.9, 0.9, and 0.99 respectively. Run times were all about 10 seconds.

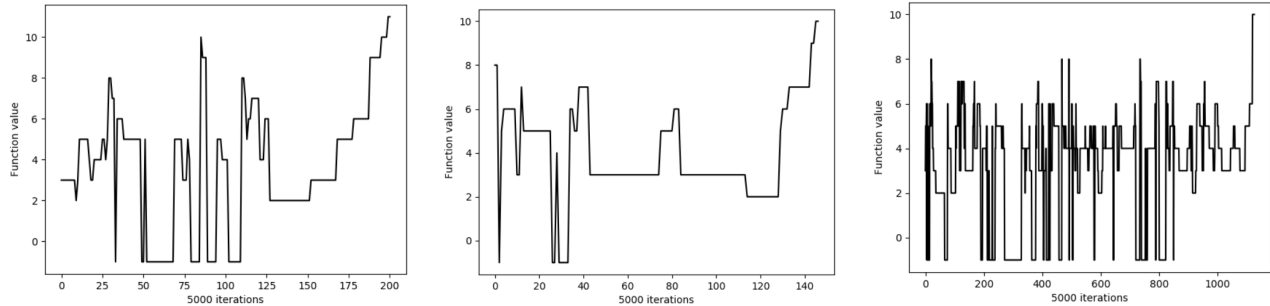


Figure 18: 7x7 puzzle simulated annealing results at 5000 iterations. Initial temperatures were 1000 and decay rates were 0.9, 0.75, and 0.99 respectively. Run times were all about 10 seconds.

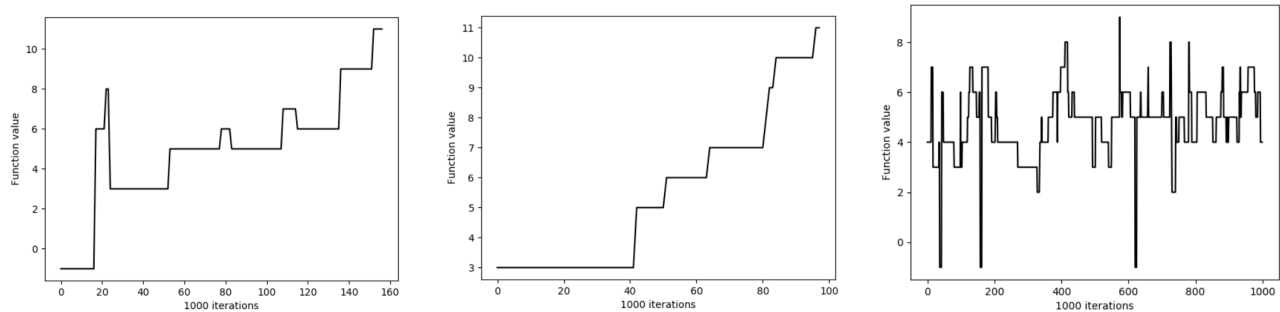


Figure 19: 9x9 puzzle simulated annealing results at 5000 iterations. Initial temperatures were 1000, 1000, 100 and decay rates were 0.9, 0.75, and 0.75 respectively. Run times were all about 15 seconds.

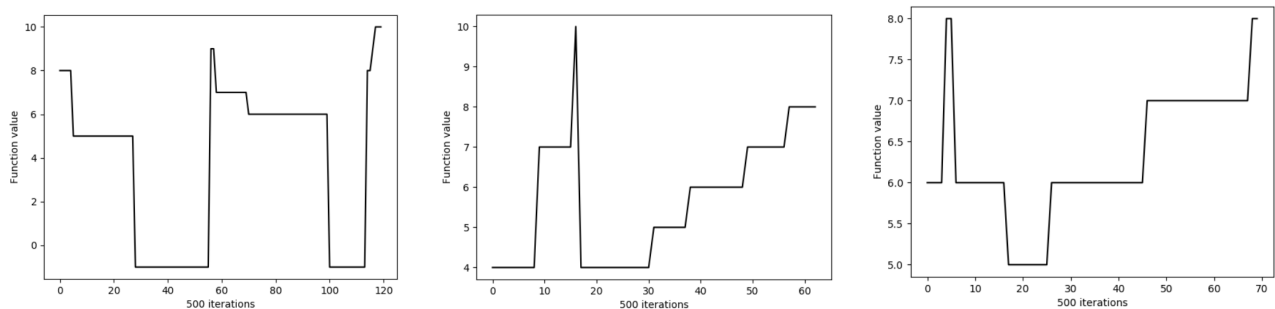
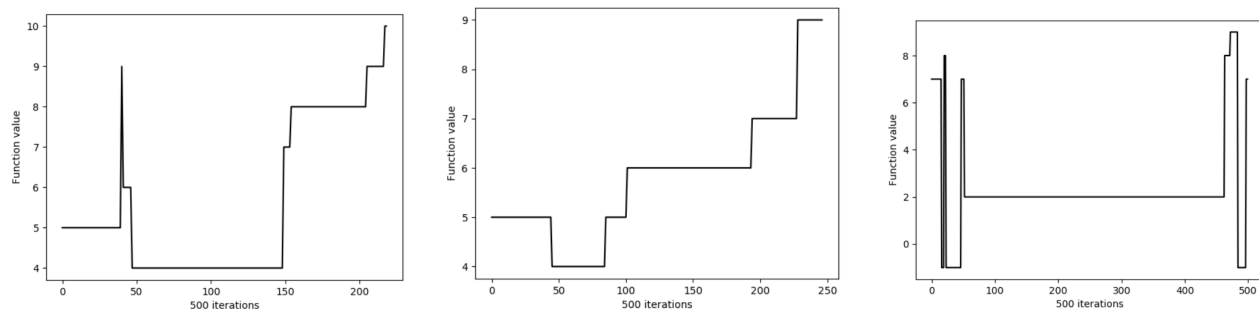


Figure 20: 11x11 puzzle simulated annealing results at 5000 iterations. Initial temperatures were 1000, 100, 1000 and decay rates were 0.9, 0.9, and 0.99 respectively. Run times were all about 42 seconds.



Task 7

For the population based approach, we decided to use the genetic algorithm as described in the textbook. An initial population was generated, in which population size was an input. The value of every puzzle in the population was evaluated, and those values were used to determine the probability of becoming a parent. Then parents were randomly chosen based on their respective probabilities. To generate children, the parents were spliced at a random row in the puzzle, and the bottom halves of the parents were swapped to create two children. The children were then mutated based on the probability of mutation given by user input. This entire process was repeated for a given number of generations, and the hardest puzzle generated through the entire process was displayed in the GUI. After lots of testing, it seems that higher mutation rates reduce the probability of making more fit populations, and sometimes high mutation rates cause the program to get stuck in an infinite loop.

Figure 21: 5x5 puzzle genetic algorithm results at a population of 50 and 100 generations. Chances of mutation per child were 0%, 1%, and 10% respectively. Run times were all about 10 seconds.

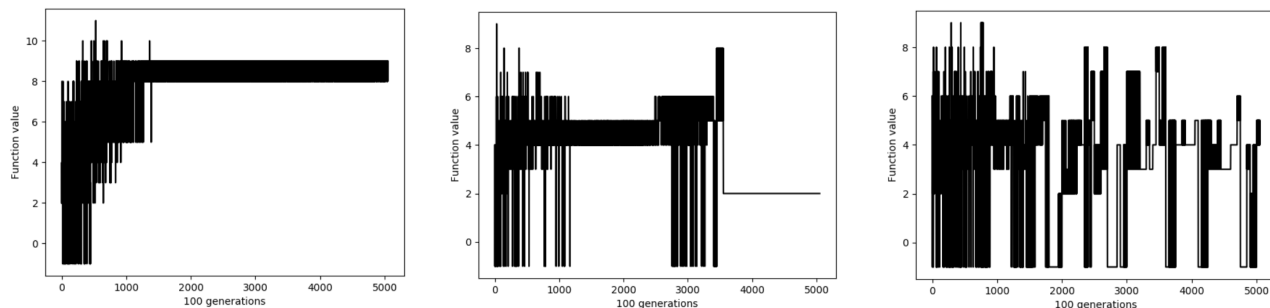


Figure 22: 7x7 puzzle genetic algorithm results at a population of 10, 50, 100, and 500, and 500, 100, 50, and 10 generations respectively. Chances of mutation was 1%. Run times were all about 50 seconds.

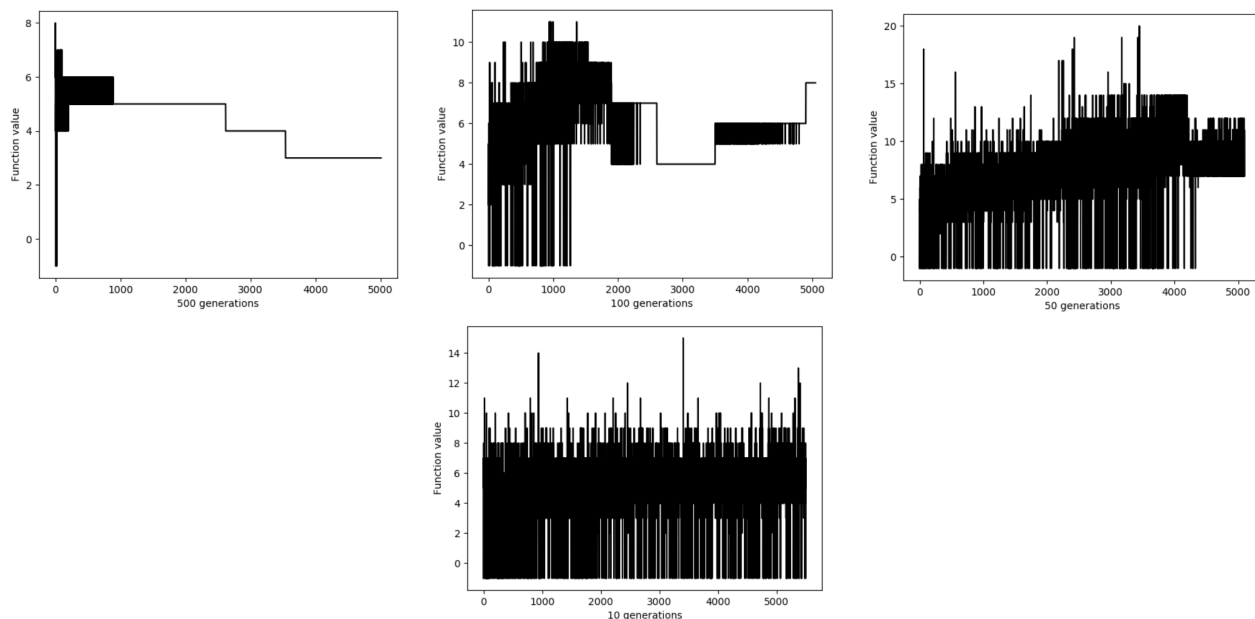


Figure 23: 9x9 puzzle genetic algorithm results at a population of 100, 10, and 4, and 10, 100, and 250 generations respectively. Chances of mutation was 1%. Run times were all about 30 seconds. These plots are getting weird.

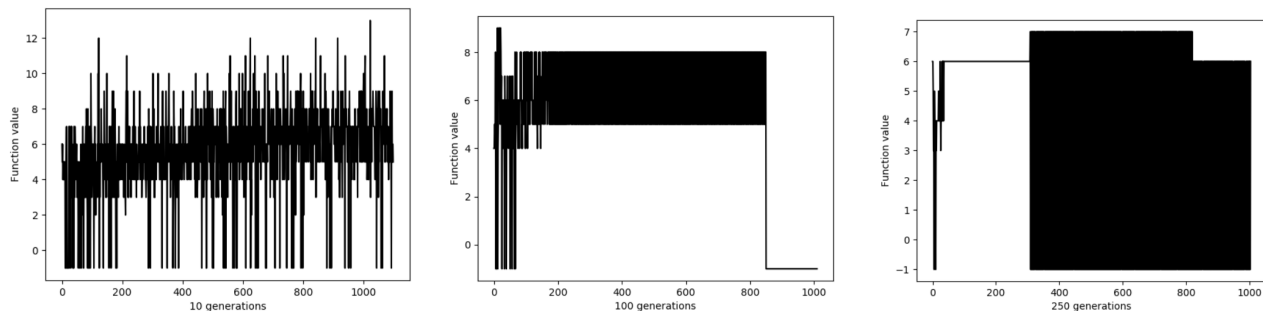


Figure 24: 11x11 puzzle genetic algorithm results at a population of 10, and 10, 25, and 50 generations respectively. Chances of mutation was 1%. Run times were 9.7, 23.2, and 43.4 seconds respectively.

