Local Search

CS 440 - Assignment 1
Group: Akhil Velagapudi, Nithin Tammishetti

Task 1

The GUI for vizualizing the puzzles was written in HTML and JavaScript. The algorithms that are responsible for actually generating the puzzles are written in Go and Python. The HTML front-end communicates with the back-end through REST calls.

Task 2

We implemented the breadth first search to evaluate puzzle fitness. The algorithm keeps track of a 'visited' matrix to ensure paths are not repeated. Another matrix is used to keep track of the previous location for each point in the puzzle. The path can be obtained by backtracking through this matrix.

Examples of different evaluations.

1	1	4	2	4
1	2	2	2	4
4	3	2	2	3
2	1	3	3	1
3	4	1	2	G

0	1	2	5	X
1	2	X	3	X
2	4	X	5	3
4	3	4	4	X
X	4	3	4	X

1	3	3	1	6	4	3
6	2	1	1	5	4	2
2	4	1	3	3	1	2
2	2	2	3	1	2	6
4	2	1	4	2	2	3
6	2	1	4	4	4	6
2	4	2	4	6	3	G

0	1	5	4	2	6	7	
1	3	5	4	3	8	2	
6	7	6	5	7	8	6	
4	2	5	3	6	7	3	
7	8	7	8	7	7	7	
5	3	6	4	8	8	6	
8	8	7	4	3	8	X	

7	8	8	8	8	7	1	2	3
8	1	7	3	3	7	7	7	2
6	5	6	4	6	2	1	6	3
2	2	4	2	1	1	6	5	1
6	2	4	3	3	1	6	1	3
5	4	3	3	2	5	6	2	1
2	5	3	1	3	5	4	3	7
6	1	3	2	2	6	7	1	4
5	3	4	7	5	4	1	7	G

0	7	7	X	X	2	3	1	X
2	6	7	9	8	4	4	4	3
X	3	10	6	8	5	4	2	5
5	9	6	8	7	7	5	5	4
8	8	6	9	7	6	7	8	5
6	6	9	8	8	7	9	7	6
9	5	7	7	8	8	6	9	7
1	4	5	8	7	3	2	8	6
4	5	7	9	6	5	5	3	X

iterations: 4000
solution: 21

3	2	2	3	3
4	1	2	2	1
1	3	1	2	4
4	3	3	3	3
1	1	2	3	G

0	4	14	1	3
8	11	12	10	9
7	5	15	16	6
1	3	13	2	2
18	19	20	17	21

iterations: 4000
solution: 33

1	5	1	4	1	5	1
6	2	2	5	1	4	6
5	5	4	3	3	5	5
3	5	2	2	3	5	5
5	1	4	3	3	1	6
1	3	3	2	3	4	5
4	4	1	4	5	1	G

0	1	25	18	17	3	2
1	6	26	7	16	5	2
10	3	29	9	4	11	4
13	7	27	11	28	12	8
20	23	24	19	22	21	20
X	2	28	10	3	4	X
14	31	30	8	15	32	33

iterations: 4000
solution: 56

3	2	2	7	7	7	6	4	3
5	1	7	1	7	3	5	5	1
7	3	5	6	5	5	2	7	8
4	1	3	5	5	5	4	4	7
4	4	3	3	4	4	4	4	2
2	3	3	5	3	2	5	3	2
1	1	3	3	5	4	3	6	2
5	7	3	5	6	7	5	5	5
1	4	8	7	5	4	7	5	G

0	14	8	1	9	4	35	49	47
44	43	6	5	6	45	42	47	46
3	15	9	3	16	27	39	4	4
1	23	24	21	2	25	37	20	22
52	13	11	51	53	12	40	50	54
29	16	30	32	17	31	34	18	33
28	27	25	37	X	26	36	48	55
2	28	10	2	10	3	38	21	3
53	14	7	4	3	13	41	19	56

Task 3

The basic hill climbing approach was run for one thousand iterations for each puzzle size to generate these graphs. The algorithm was averaged over fifty iterations for each puzzle size. A graph with the algorithm running for ten thousand iterations is also included. The search with 1000 iterations was able to yield an evaluation of 88.

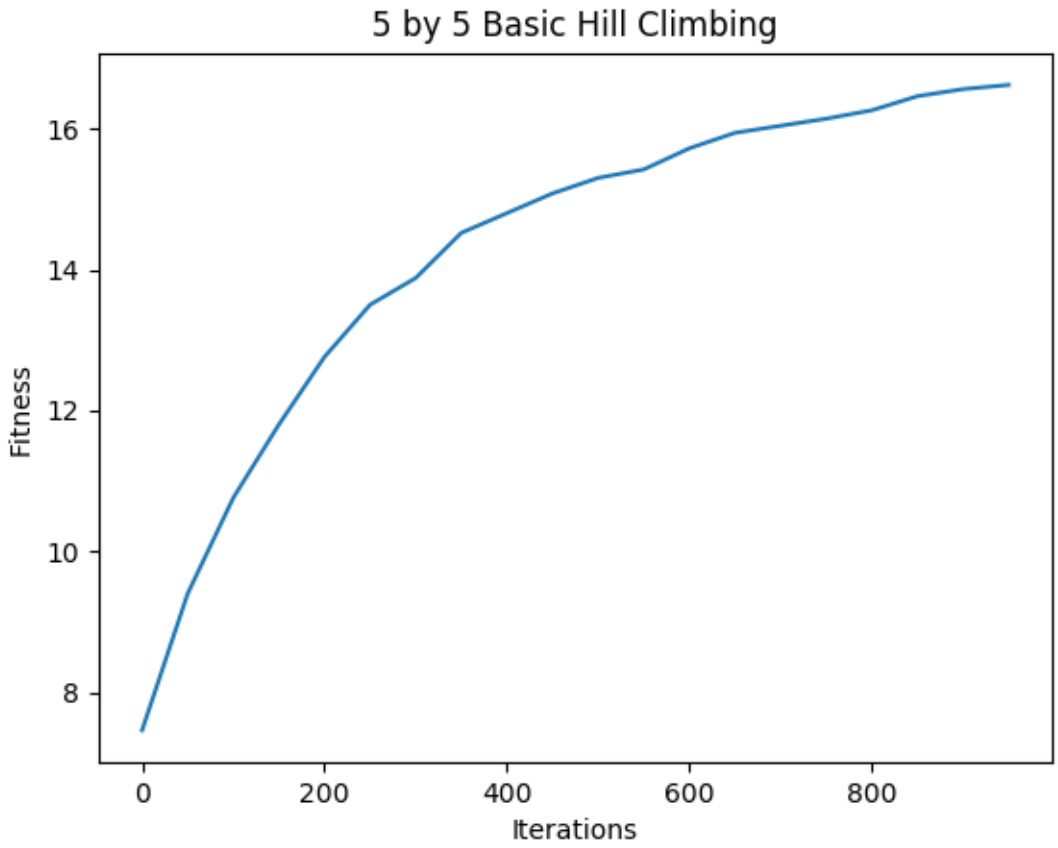

Figure 1: Basic hill climbing for 5x5 puzzle

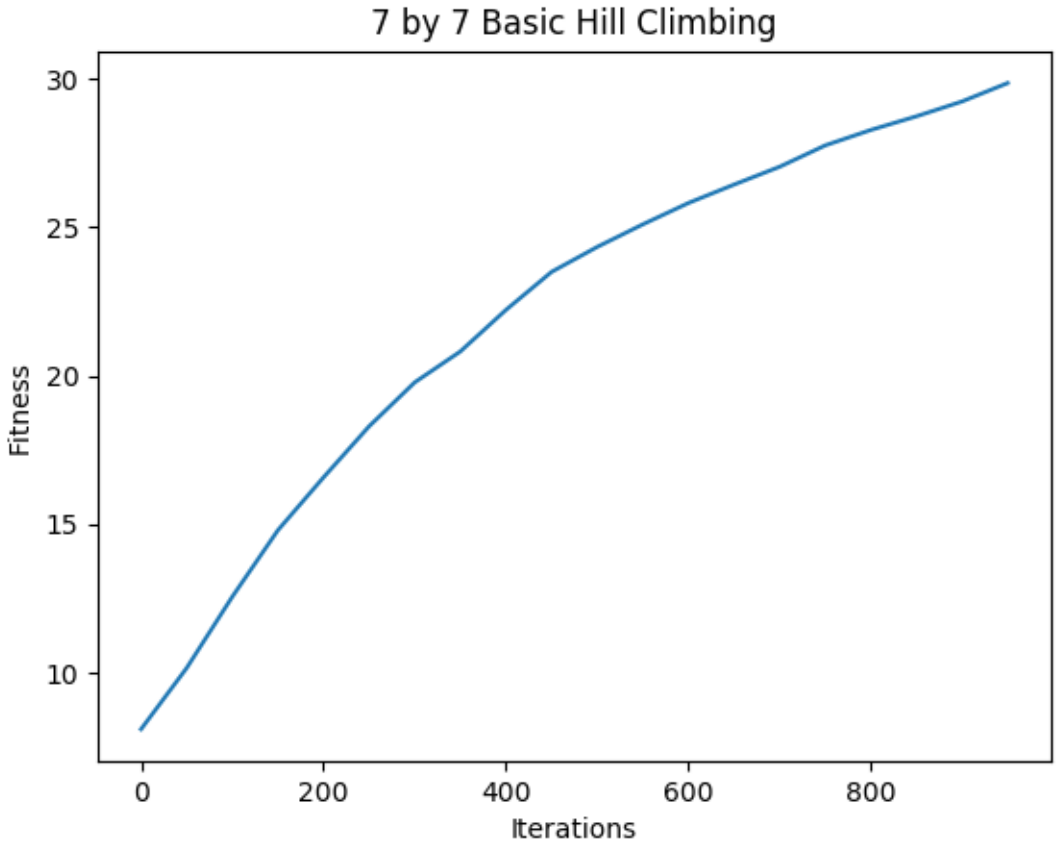

Figure 2: Basic hill climbing for 7x7 puzzle

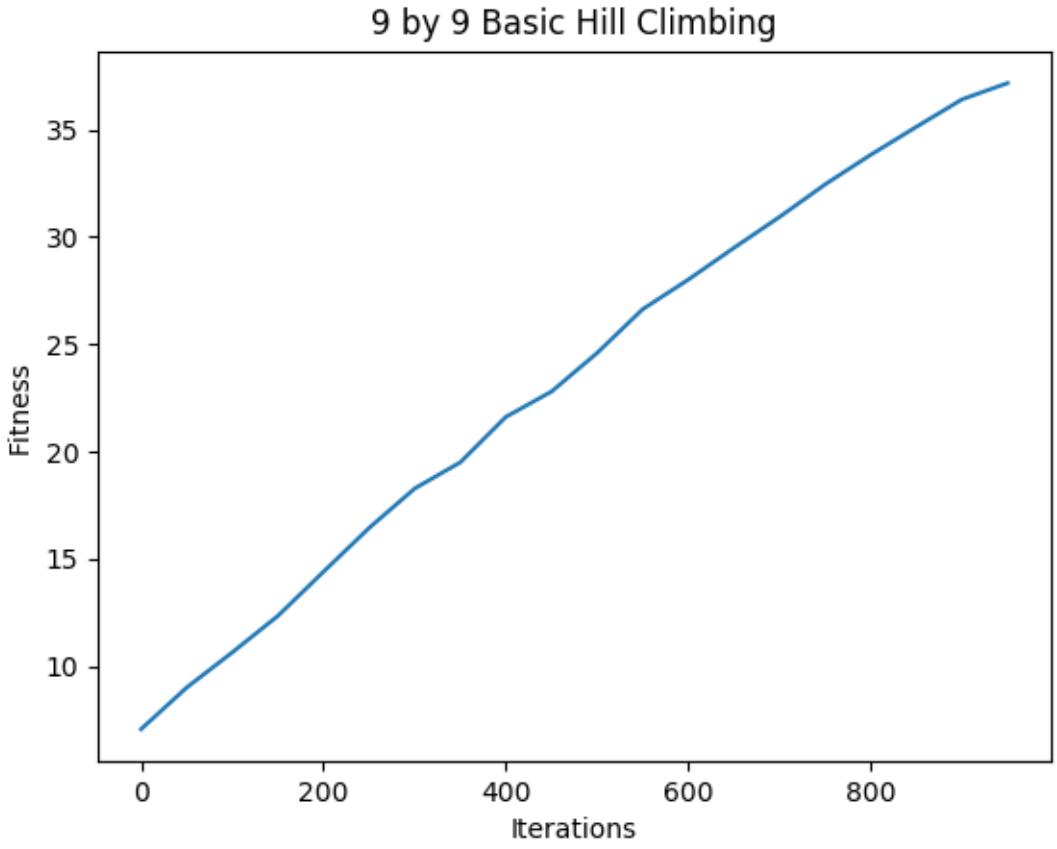

Figure 3: Basic hill climbing for 9x9 puzzle

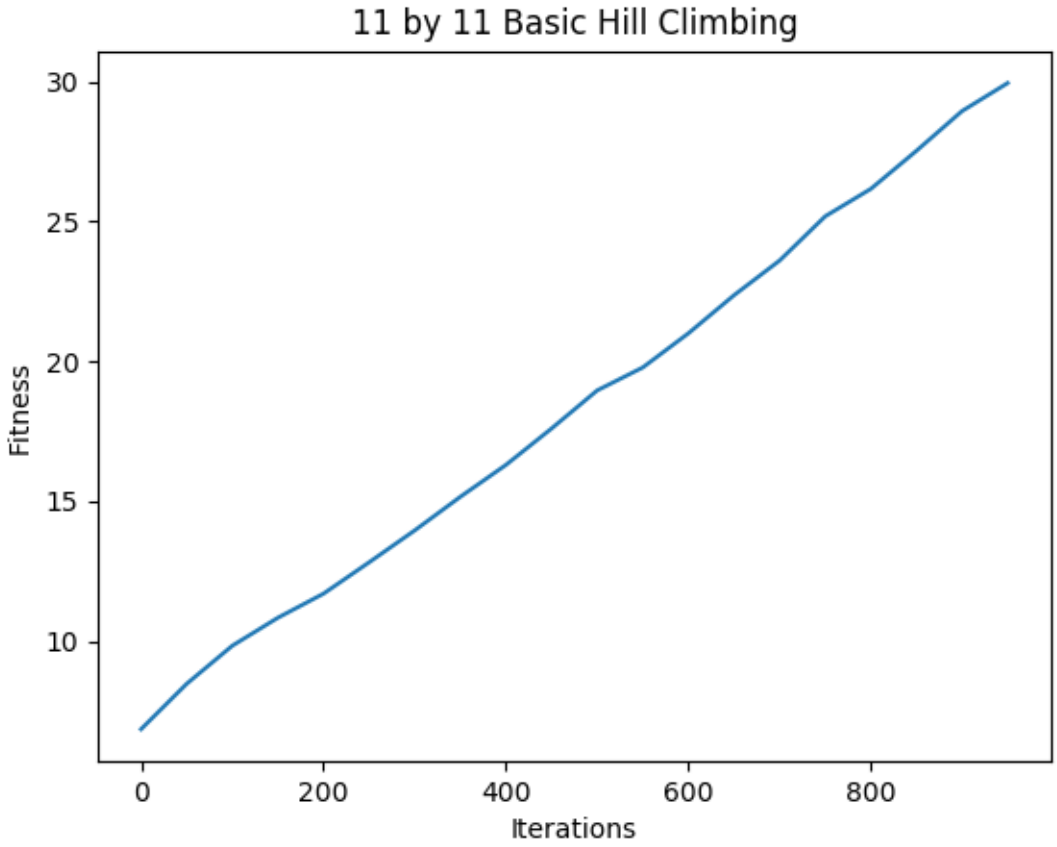

Figure 4: Basic hill climbing for 11x11 puzzle

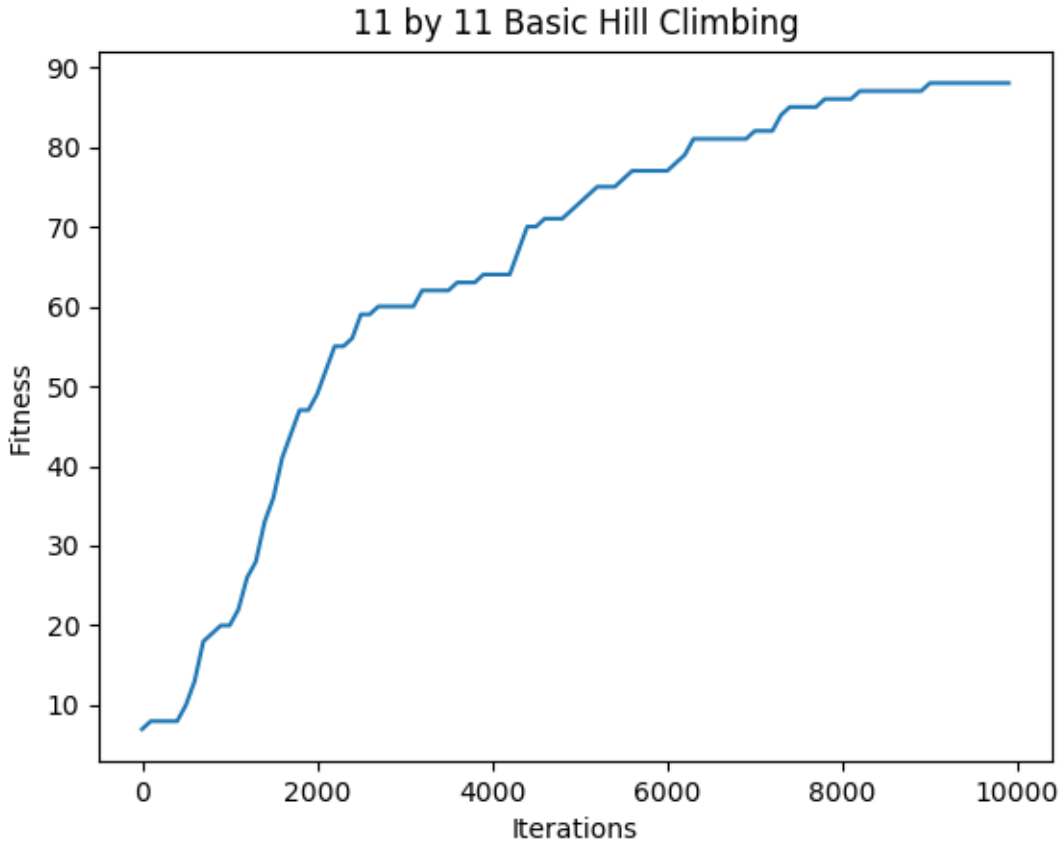

Figure 5: Basic hill climbing for 11x11 puzzle deep

Task 4

The hill climbing approach with random restarts was run for one thousand iterations for each puzzle size to generate these graphs. The algorithm was averaged over fifty iterations for each puzzle size. A graph with the algorithm running for ten thousand iterations is also included.

Since the algorithm does not run deep enough for these graphs, the performance might seem inferior. But this is expected since the restarts happen before hitting a local maximum, so the algorithm simply starts over and simply slows down the progress. The overall number of iterations remains the same since there were 2 restarts with 500 iterations each. The deeper graph was run with 4 restarts with 2500 iterations each. The 10000 iteration run was able to reach an evaluation of 87. But for very large numbers of iterations, restarts would win over basic hill climbing since the algorithm simply runs hill climbing many times so it is bound to go past local optima eventually.

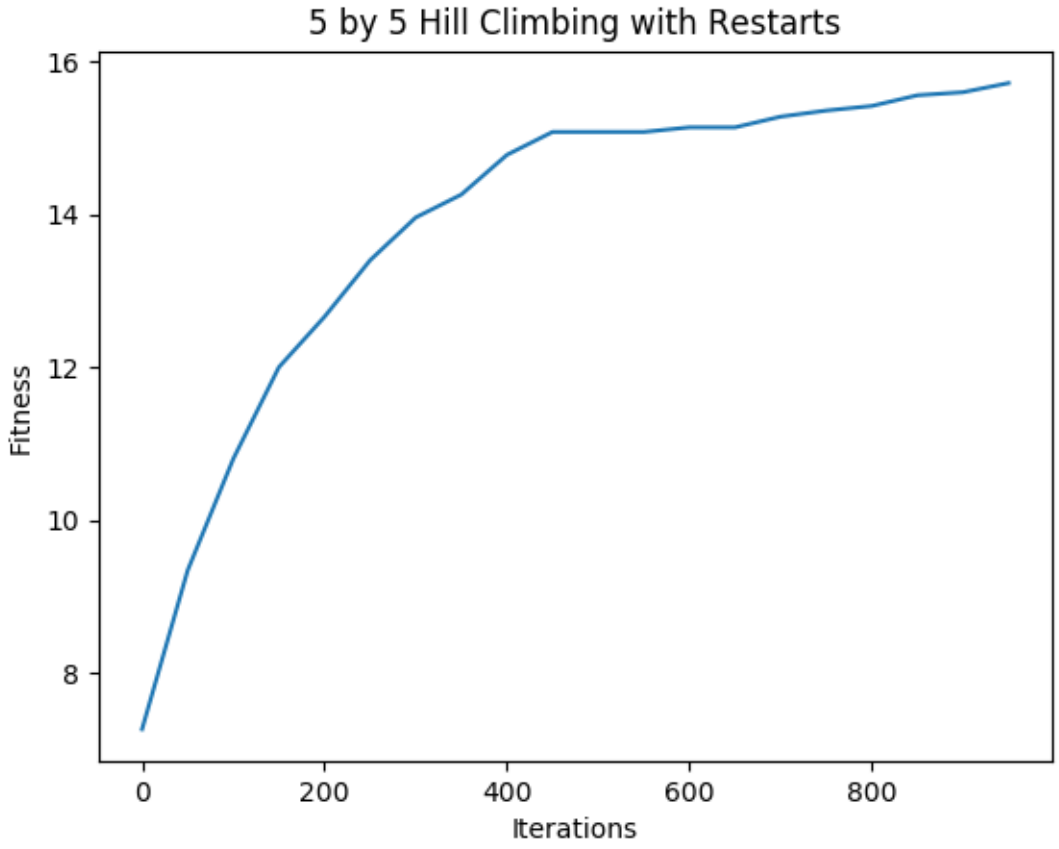

Figure 6: Hill climbing with restarts for 5x5 puzzle

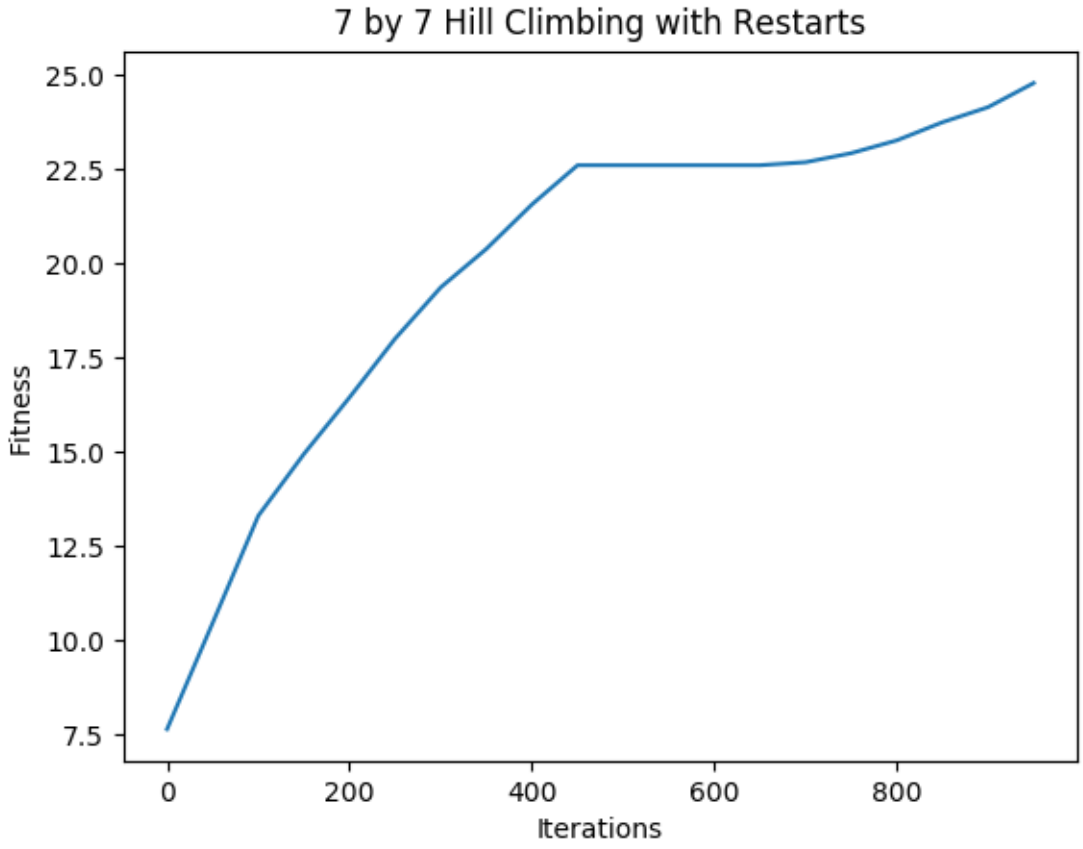

Figure 7: Hill climbing with restarts for 7x7 puzzle

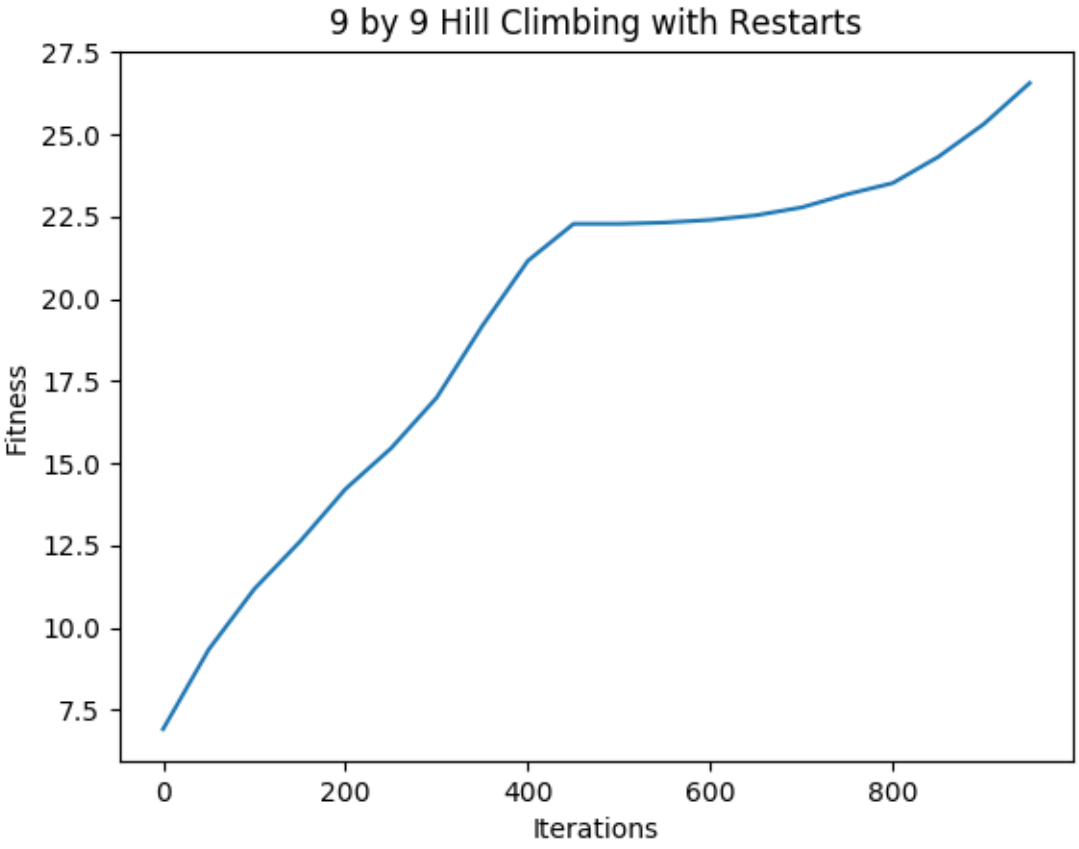

Figure 8: Hill climbing with restarts for 9x9 puzzle

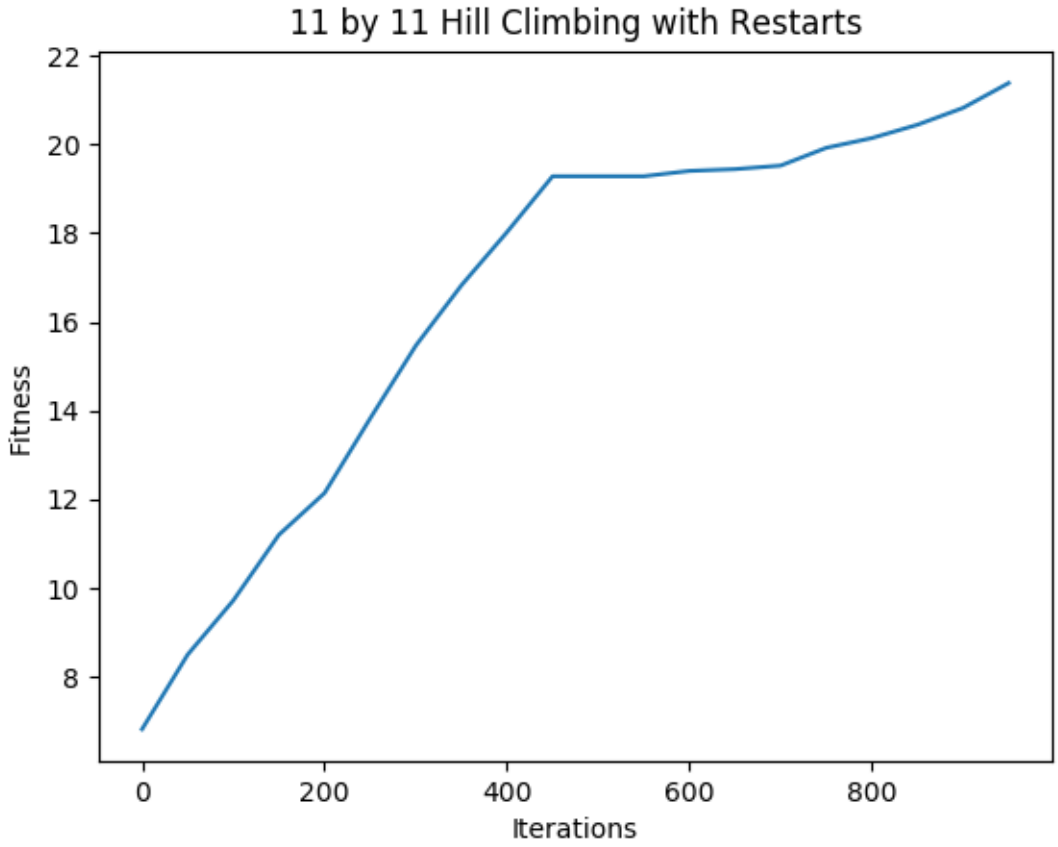

Figure 9: Hill climbing with restarts for 11x11 puzzle

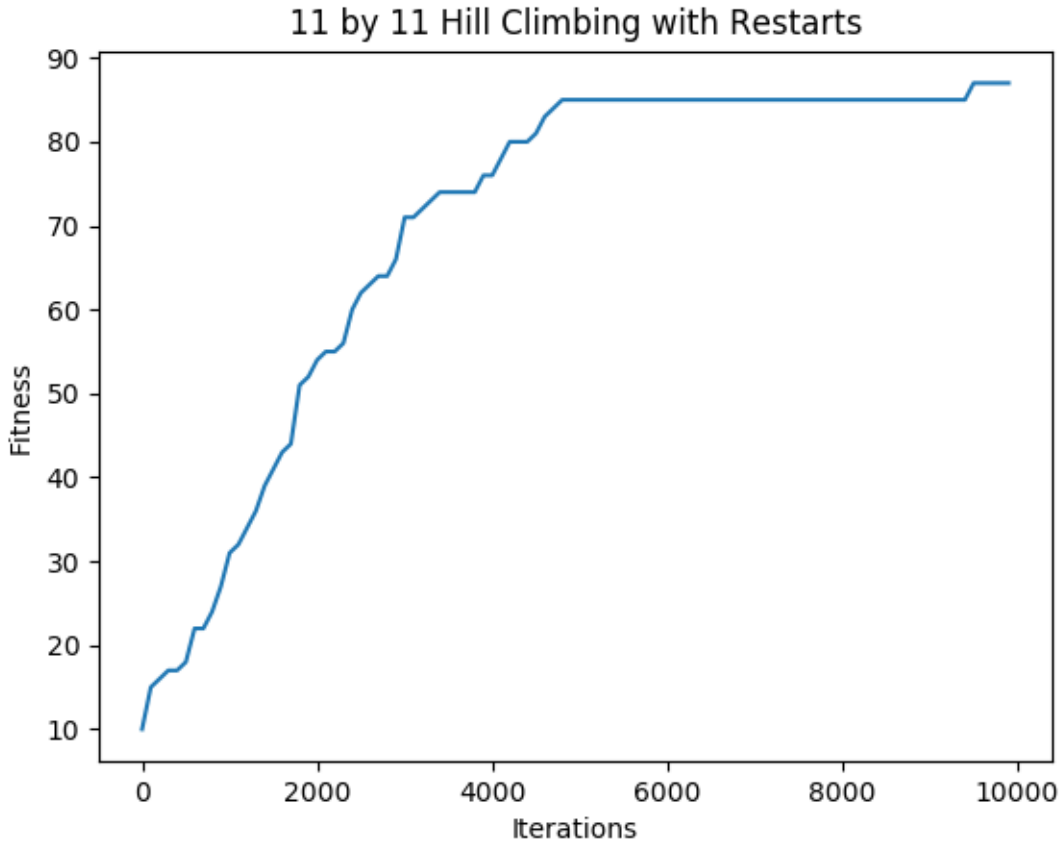

Figure 10: Hill climbing with restarts for 11x11 puzzle deep

Task 5

The hill climbing approach with random walk probability was evaluated for each puzzle size. The parameter p was chosen by first trying out a wide range of p values and observing the performance of the algorithm. Then a small range of p values was examined for fine tuning the parameter choice. The final value we chose was p = 0.0025.

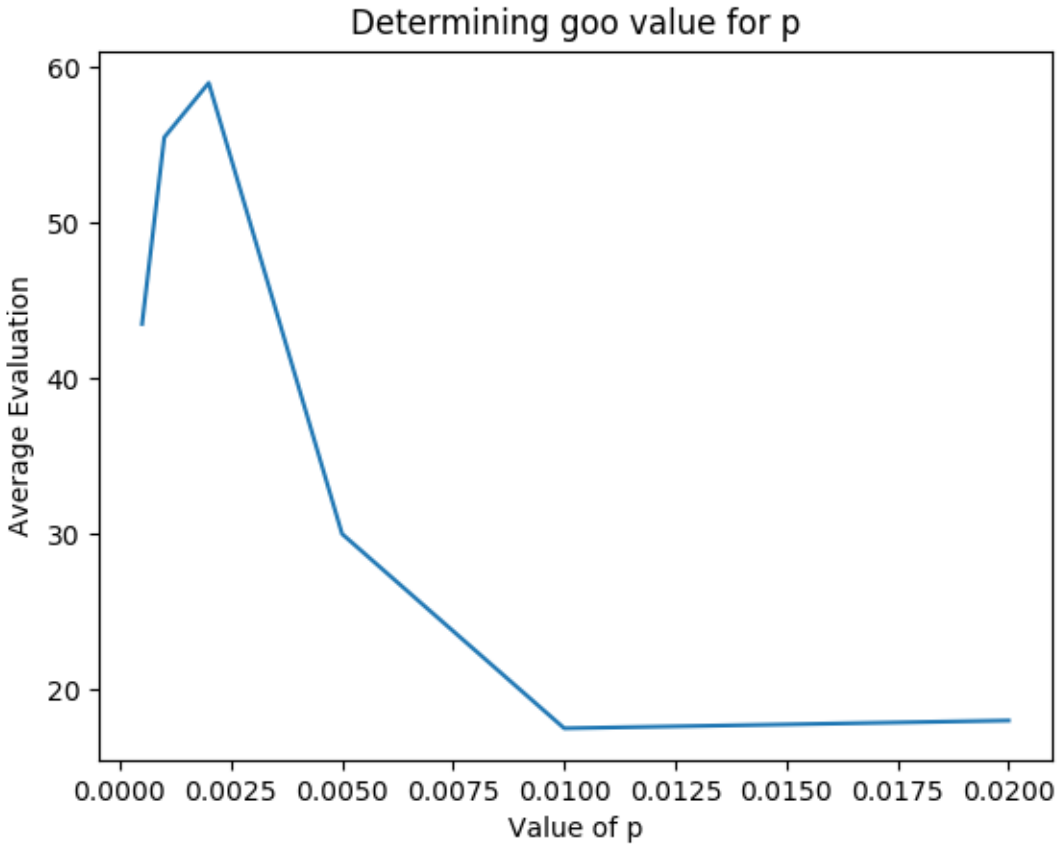

Figure 11: Trying out different p values for random walk

The algorithm was also run for 10000 iterations to compare its performance with the other approaches. This deeper search was only able to reach an evaluation of 65. This is because the algorithm potential makes hugely detrimental leaps due to its completely random nature. As can be seen from the graph, the algorithm struggles with making consistent progress due to the huge dips in fitness caused by random chance.

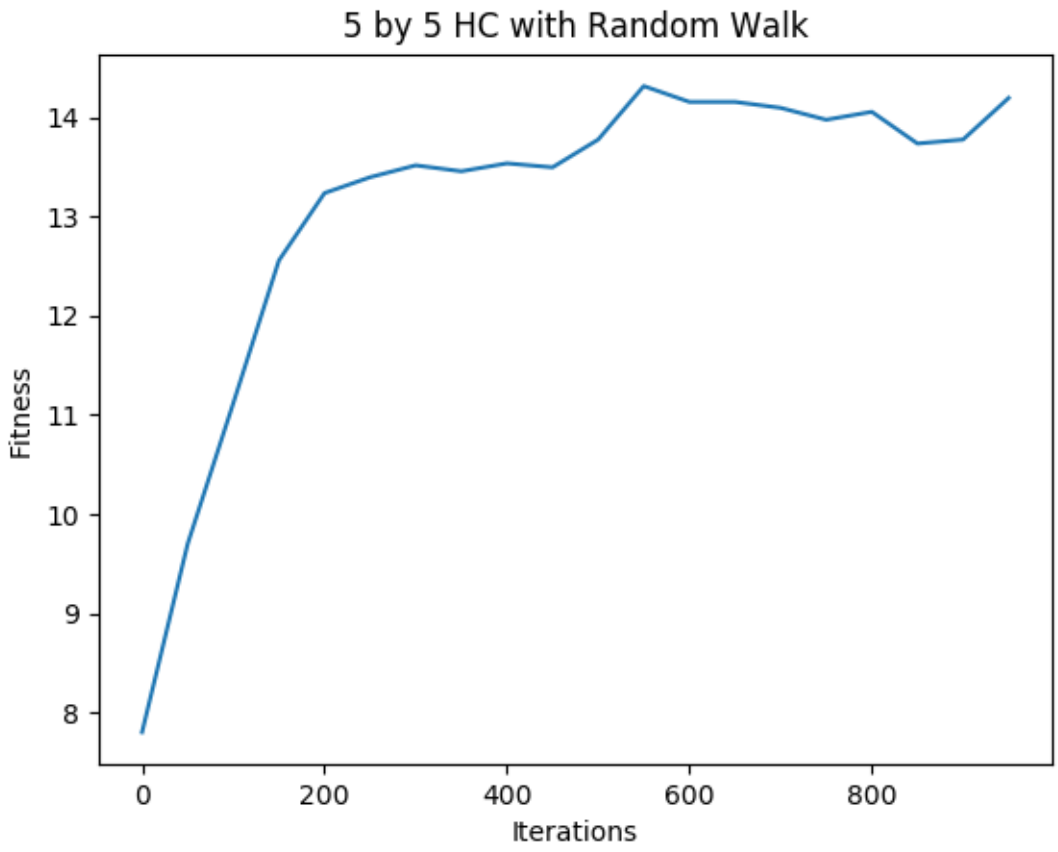

Figure 12: Hill climbing with random walk for 5x5 puzzle

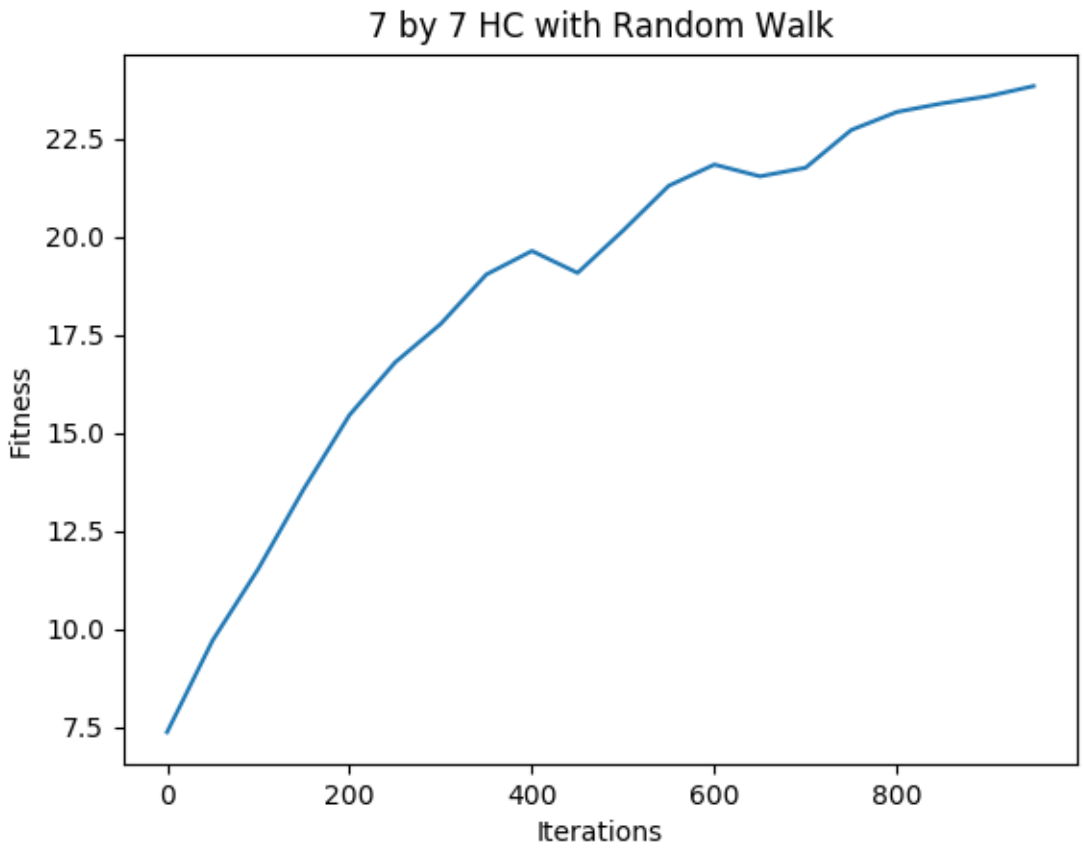

Figure 13: Hill climbing with random walk for 7x7 puzzle

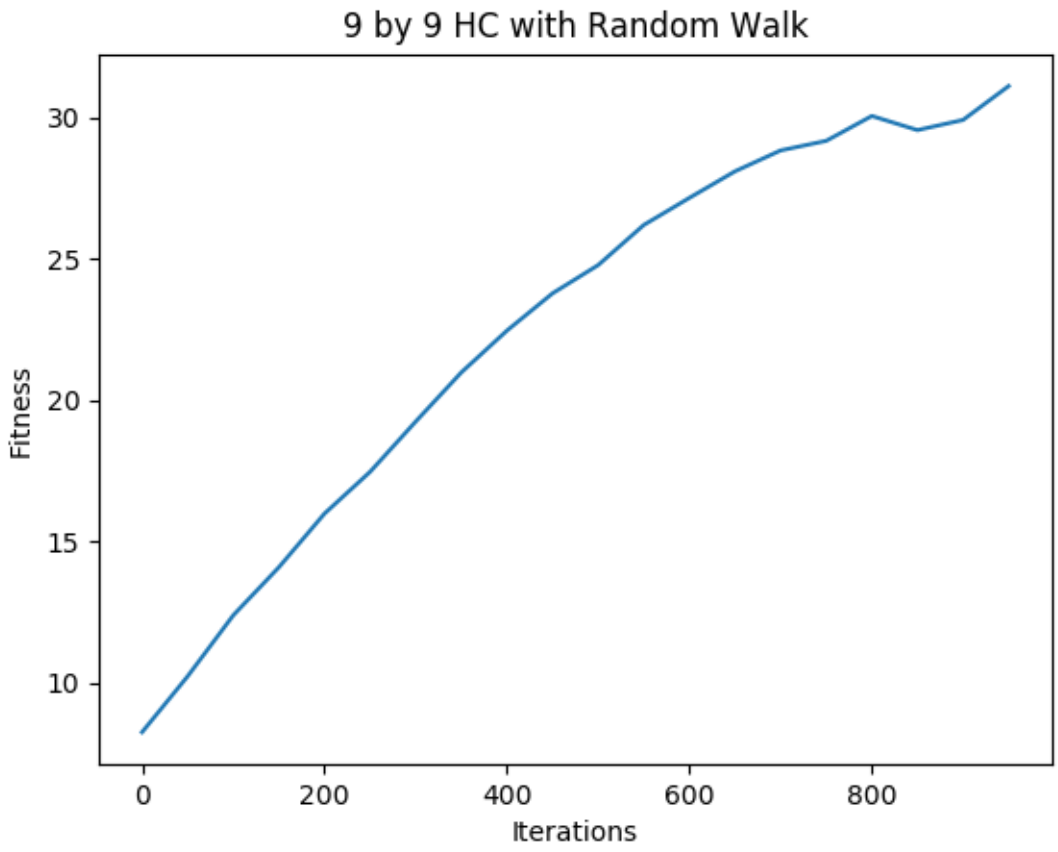

Figure 14: Hill climbing with random walk for 9x9 puzzle

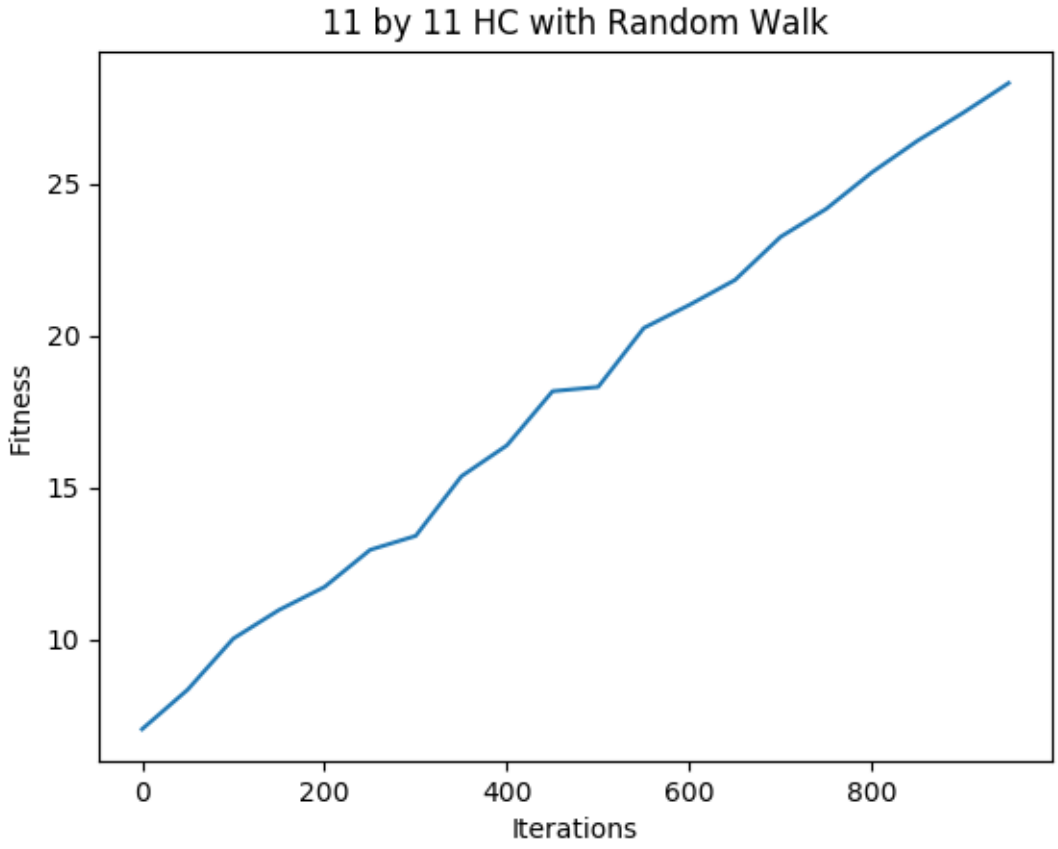

Figure 15: Hill climbing with random walk for 11x11 puzzle

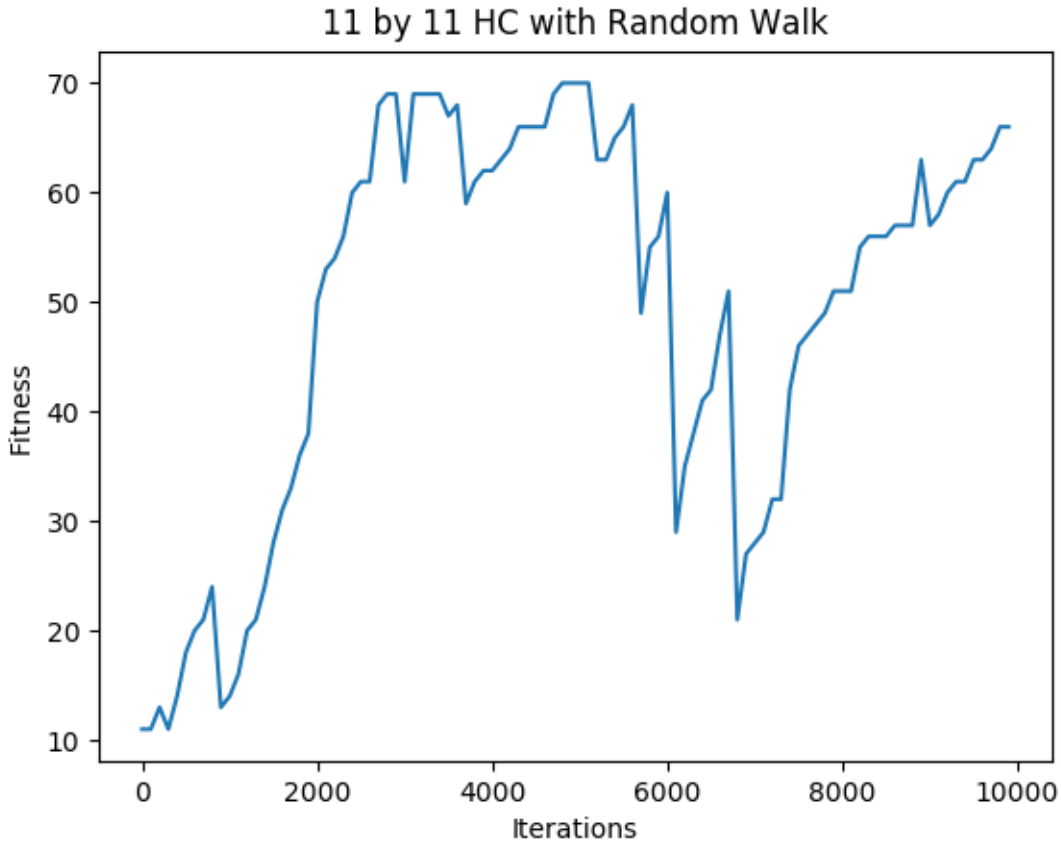

Figure 16: Hill climbing with random walk for 11x11 puzzle deep

Task 6

The last hill climbing approach involved simulated annealing, where the algorithm starts off with a high initial temperature which leans more toward random walk. As the iteration number increases, the temperature is lowered and the algorithm decomposes to hill climbing. The temperature and decay rate interact together to contol the probability of taking a step backward in the state space. At the beginning, the algorithm explores a wider state space to arrive an optimal location and then proceeds with hill climbing through a deeper search. The algorithm also takes into account the magnitude of the change in fitness when calculating the probability of taking a detrimental backwards step. So a jump that would lead to a drop in fitness of 50 is less likely to happen than a jump that would only lead to a drop of 3. So this algorithm does not suffer from the shortcomings of the random walk implementation since it is not likely to undertake hugely detrimental steps.

The choice of temperature and decay rate are dependent on the number of iterations since the temperature might decay too fast or too slow. For example having a slowly decaying algorithm with decay rate = 0.999 might work well for 10000 iterations but would not work as well for 500 iterations since it would not have enough time to even reach the hill climbing part of the algorithm. The decay rate was chosen for 10000 iterations by observing the changes in probability throughout the iterations for a given step size in fitness. Decay rate = 0.999 was chosen for the 10000 iteration run. The temperature choice was chosen by running the algorithm with variable temperatures and observing the fitness. The temperature calue we chose was 80 since its average fitness was 90.5, the highest of numbers we iterated over(30,40,50,80,120,200).

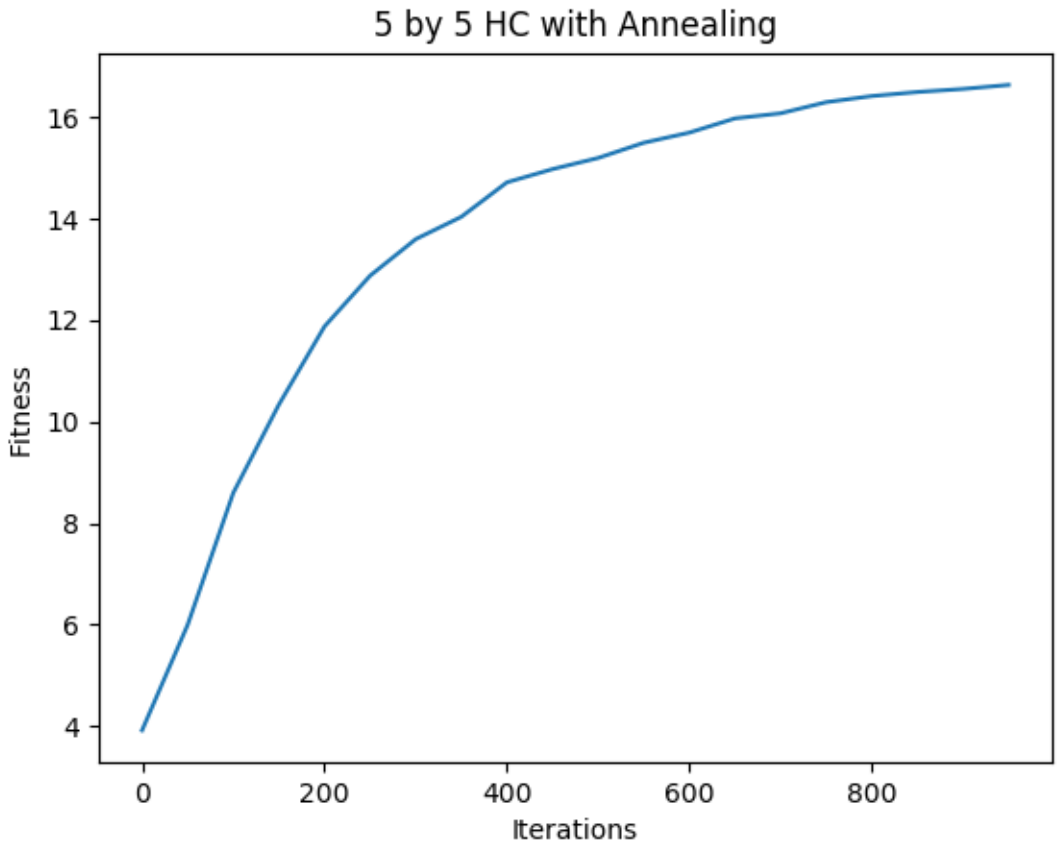

Figure 17: Hill climbing with simulated annealing for 5x5 puzzle

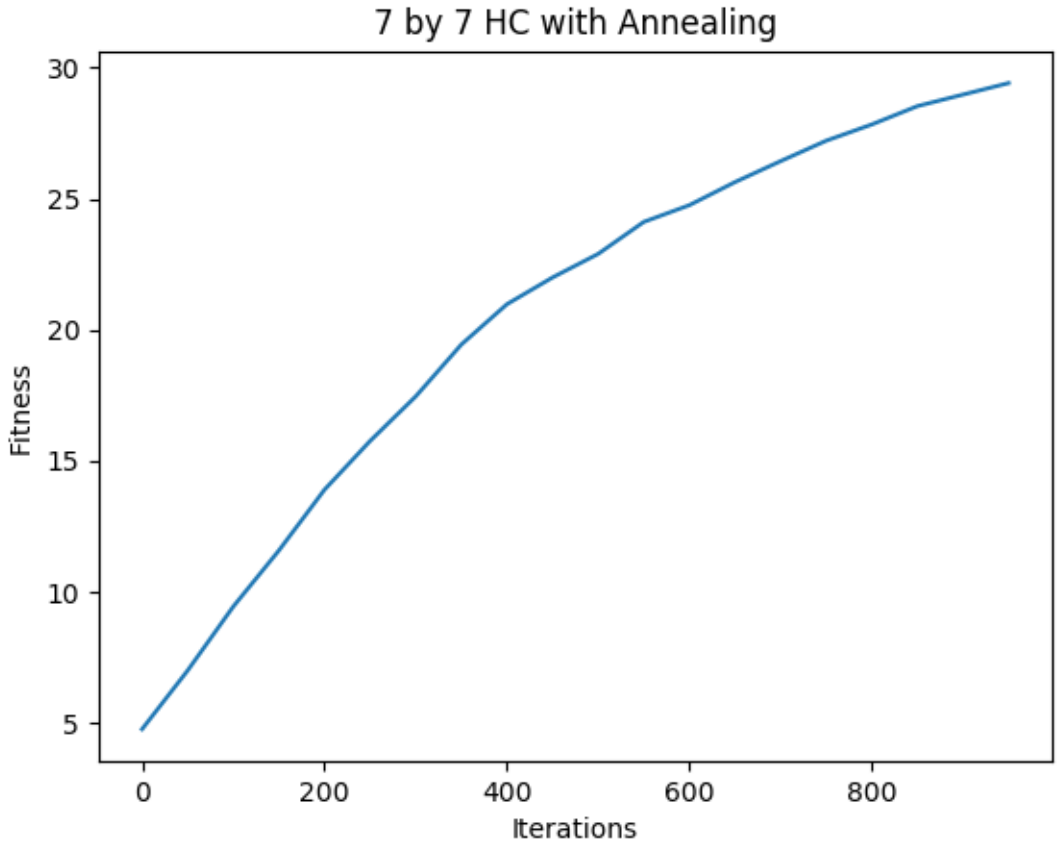

Figure 18: Hill climbing with simulated annealing for 7x7 puzzle

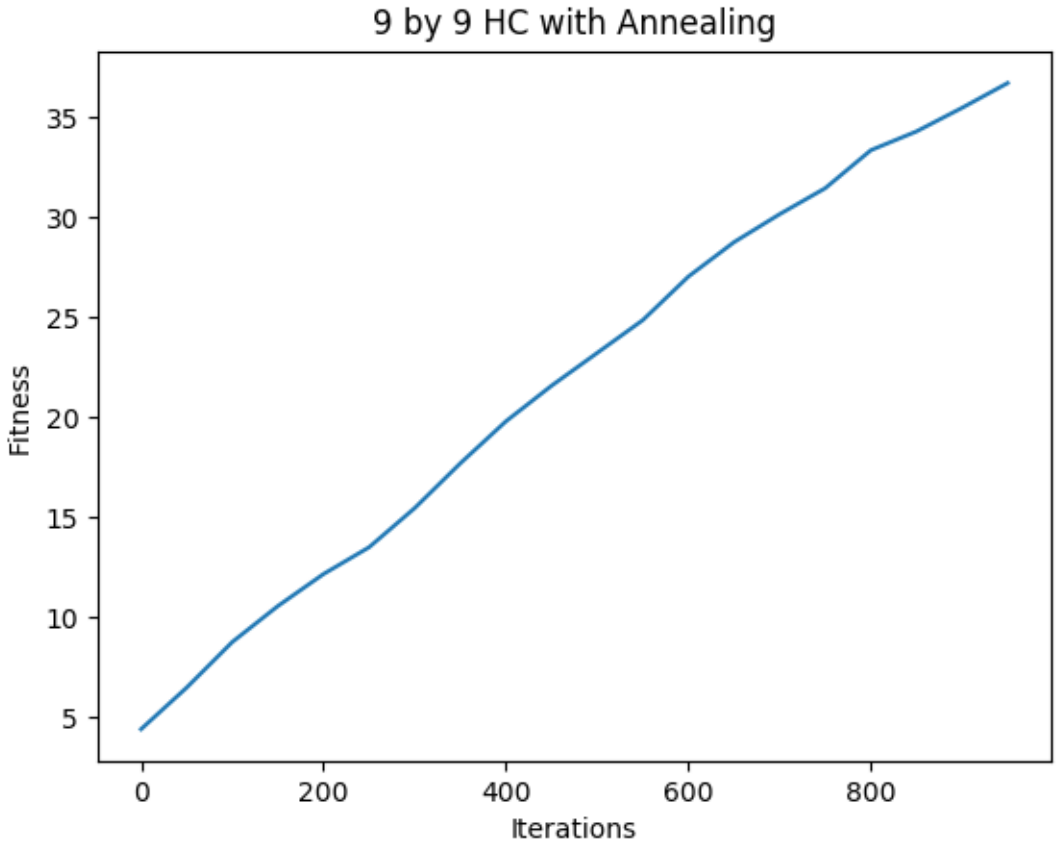

Figure 19: Hill climbing with simulated annealing for 9x9 puzzle

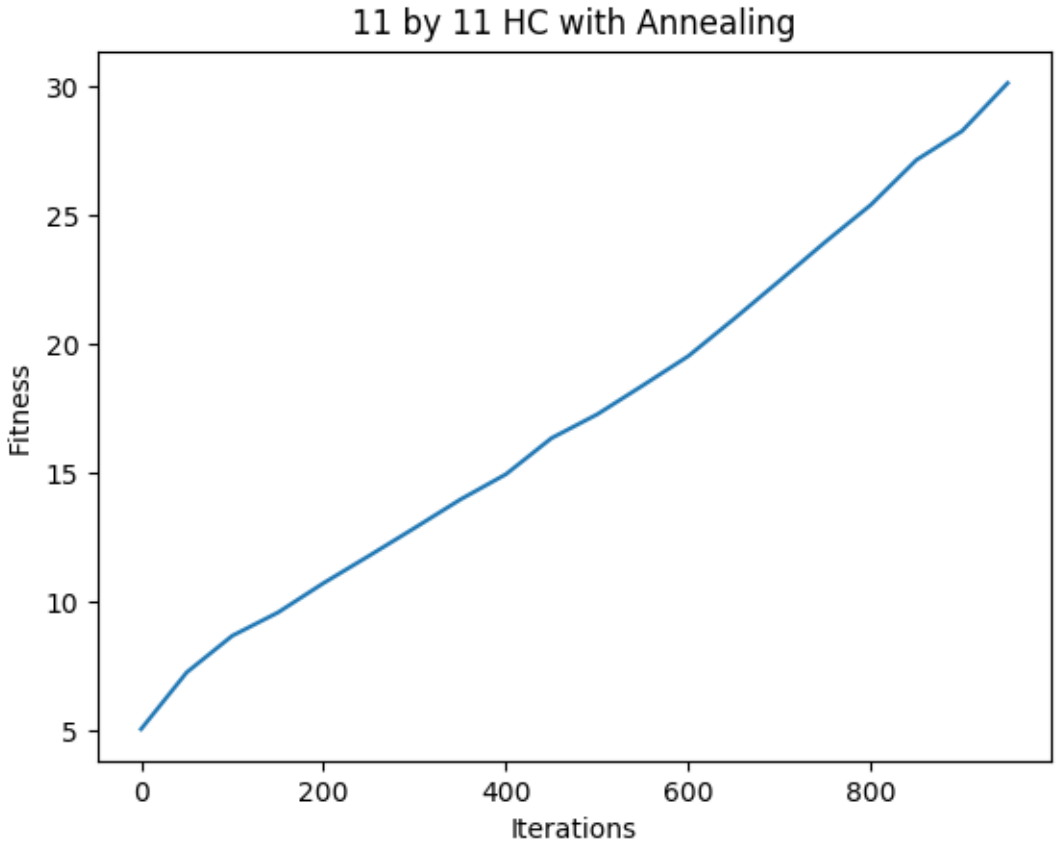

Figure 20: Hill climbing with simulated annealing for 11x11 puzzle

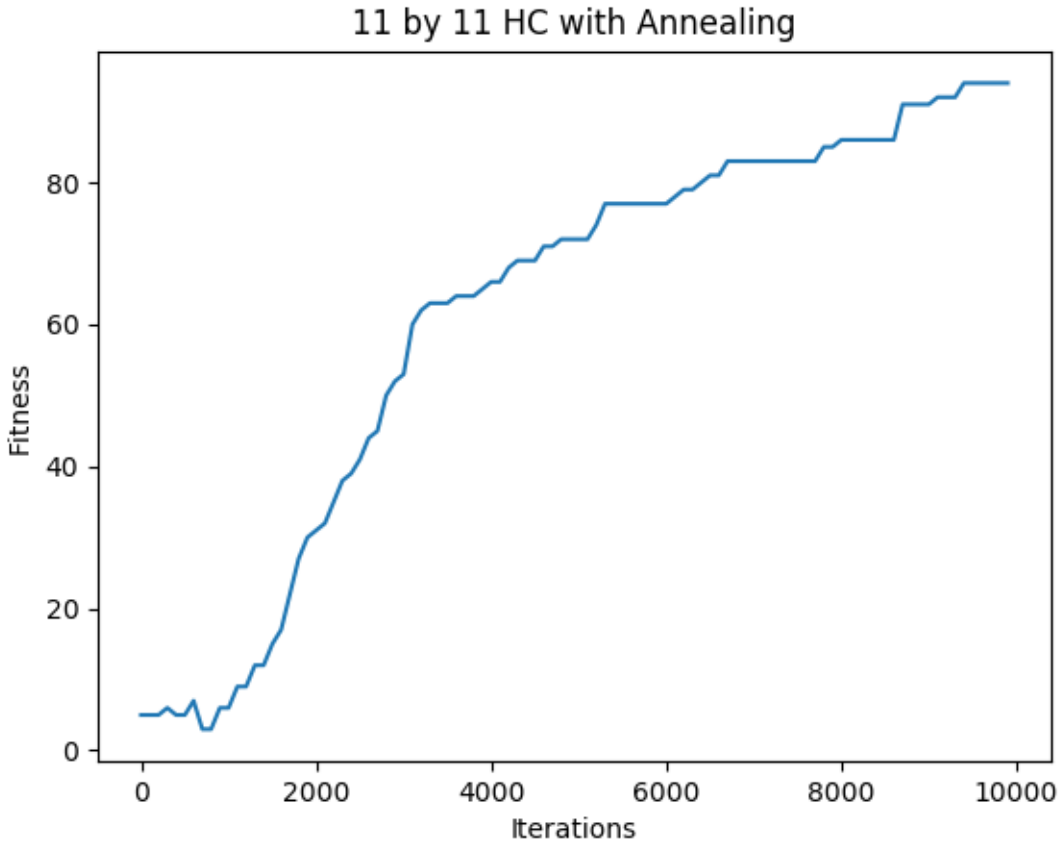

Figure 21: Hill climbing with simulated annealing for 11x11 puzzle deep

Task 7

We implemented a genetic algorithm for our population-based approach. Each puzzle is represented as a chromosome by concatenating the values in each cell left to right and top to bottom. To make this simpler, the internal implementation used for the puzzle matrix was a flat array.

Parameter	Value	Justification / Reasoning
`n` size	given (5,7,9,11, ...)	
number of generations	`4000`	Can be flexible but chosen to limit the computation time while also providing reasonably difficult puzzles. According to plots provided below, 4000 generations is well past the point of diminishing returns.
survival rate (subset of population that survives to then reproduce including elitists)	`0.3`	This factor is important in maintaining the genetic diversity of the population. A small survival rate can quickly arrive at local minimum by exploring only the very successful individuals. A higher survival rate can maintain a greater level of diversity in the population, leading to more exploration.
mutation rate (likelihood of each cell mutating after crossover)	`0.018`	Similar to survival rate, this parameter is important in determining the exploration vs exploitation of the genetic algorithm. High mutation rates are good for exploring new areas of the state space but come at cost of overlooking well-performing individuals. So, they slow down convergence to a local maximum but increase the likelihood of convergence to the global maximum.
population size	`n * n * 2`	This parameter has a positive correlation with diversity in the population. The cost of a large population is the increased computation necessary for simulating each generation. It is reasonable for this parameter to depend on the size of the state space, which is why it is represented as a function of n in our implementation.
elitism (number of individuals that survive each generation non-probabilistically)	`pop_size * surv_rate * 0.2`	Elitism is important for accelerating convergence because it allows the algorithm to not waste time re-discovering well-performing partial solutions that were discarded due to probabilistic selection. This introduces a form of greediness into the algorithm. Having too many elite individuals can hurt diversity.

Selection (w/ Elitism)

The selection model we employed is a fixed percent of the population (30%) survives and is responsible for repopulating to the original population size by creating offspring. These are the steps that take place each generation:

1. The fitness of each individual in the population is calculated
2. Elite individuals are automatically added to the survival pool
3. The fitness scores of all the individuals in the population are normalized to be positive by adding `n^2` to each score
4. A survivor is picked from the population using a random number generator with the normalized fitness of the individuals acting as weights
5. Step 4 is repeated until the survival pool is full (size of survival pool is 30% of population size)
6. Elite individuals are automatically added to the new population

7. 2 parents are chosen randomly from the survival pool and create 2 offspring (through crossover and mutation) which are added to the new population
8. Step 7 is repeated until the size of the new population is the same as the size of the initial population

Crossover

After 2 parents are chosen for crossover, a crossover point is randomly picked between 25% and 75% of the length of the chromosomes. This method was implemented instead of picking a completely random crossover point in order to ensure atleast some of the features of each parent are passed on.

After a crossover point has been determined, all of the values to the right of the crossover point are swapped between the parent puzzles. The resulting offspring then go through a mutation process before being added to the new population.

Mutation

Our original implementation of mutation simply tested each each cell in the puzzle to see if it should be mutated (using a random number generator and the cell mutation rate).

We found this to be inefficient and switched to determining whether a mutation should happen in the puzzle overall. If so, a random cell is picked and mutated to a new value.

In order to allow for multiple mutations, mutations will continue to occur as long as the random numbers being generated are less than the mutation rate. However, to avoid excessive multiple mutations, the mutation rate is reduced by a factor of 3 with each mutation in an individual.

iterations: 4000
fitness: 68

6	7	2	2	6	7	6	6	5
4	5	7	5	5	4	6	3	6
4	3	4	5	4	6	5	6	3
2	7	4	4	2	1	3	6	3
8	4	3	3	2	4	2	2	5
1	5	6	3	2	4	5	5	6
4	4	6	6	5	1	6	5	7
8	5	6	6	3	5	5	2	2
5	1	8	5	2	8	3	3	G

0	15	24	17	25	7	1	59	16
34	32	22	20	3	31	33	61	21
2	13	25	18	3	5	12	63	4
46	51	47	49	54	53	48	50	52
36	43	41	38	40	42	39	62	37
35	14	6	X	55	30	15	58	5
1	28	26	21	2	29	2	60	27
9	12	42	19	56	8	11	57	10
45	44	23	67	65	6	66	64	68

Figure 22: Puzzle with fitness of 68 generated by genetic algorithm

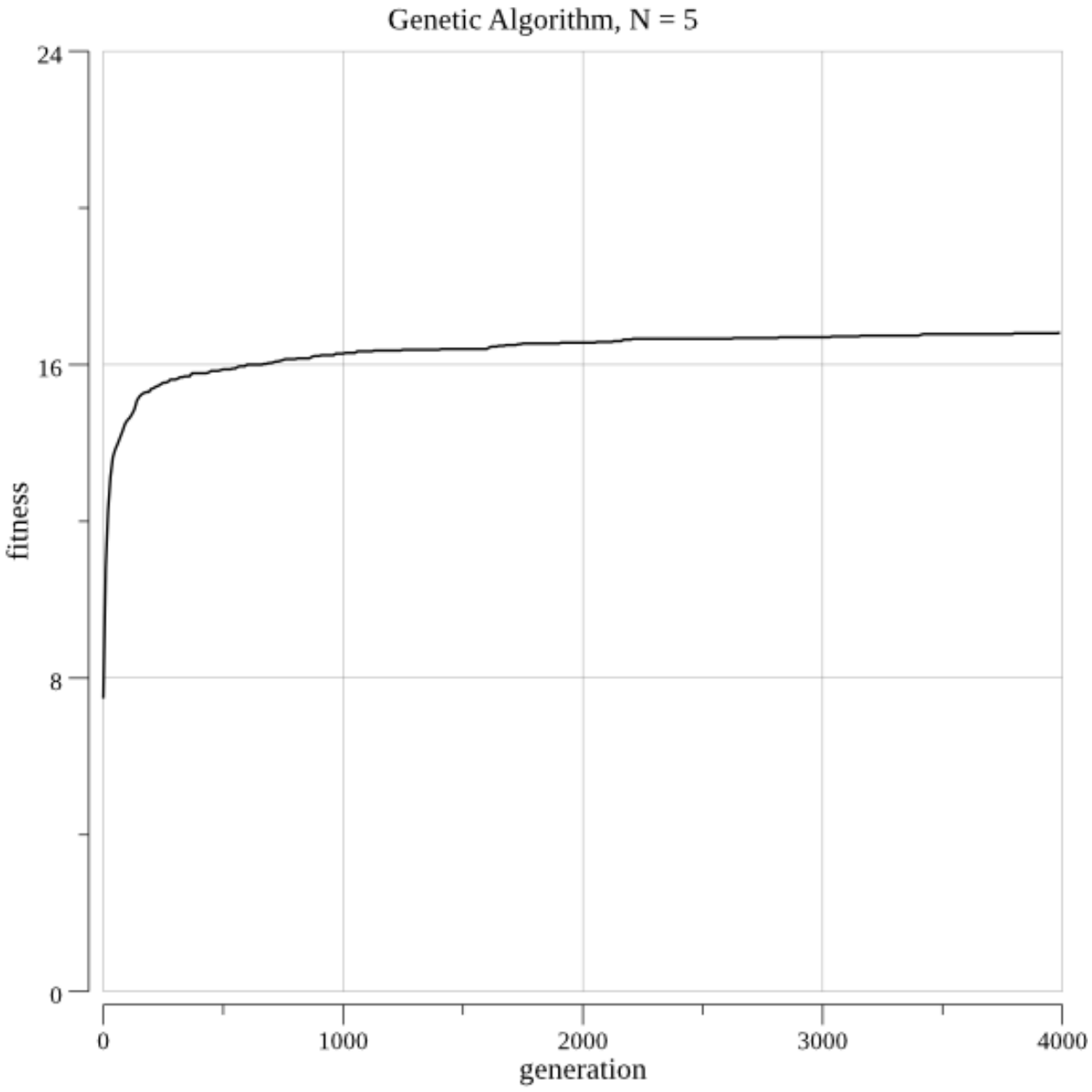

Figure 23: Fitness over number of generations for 5x5 puzzle

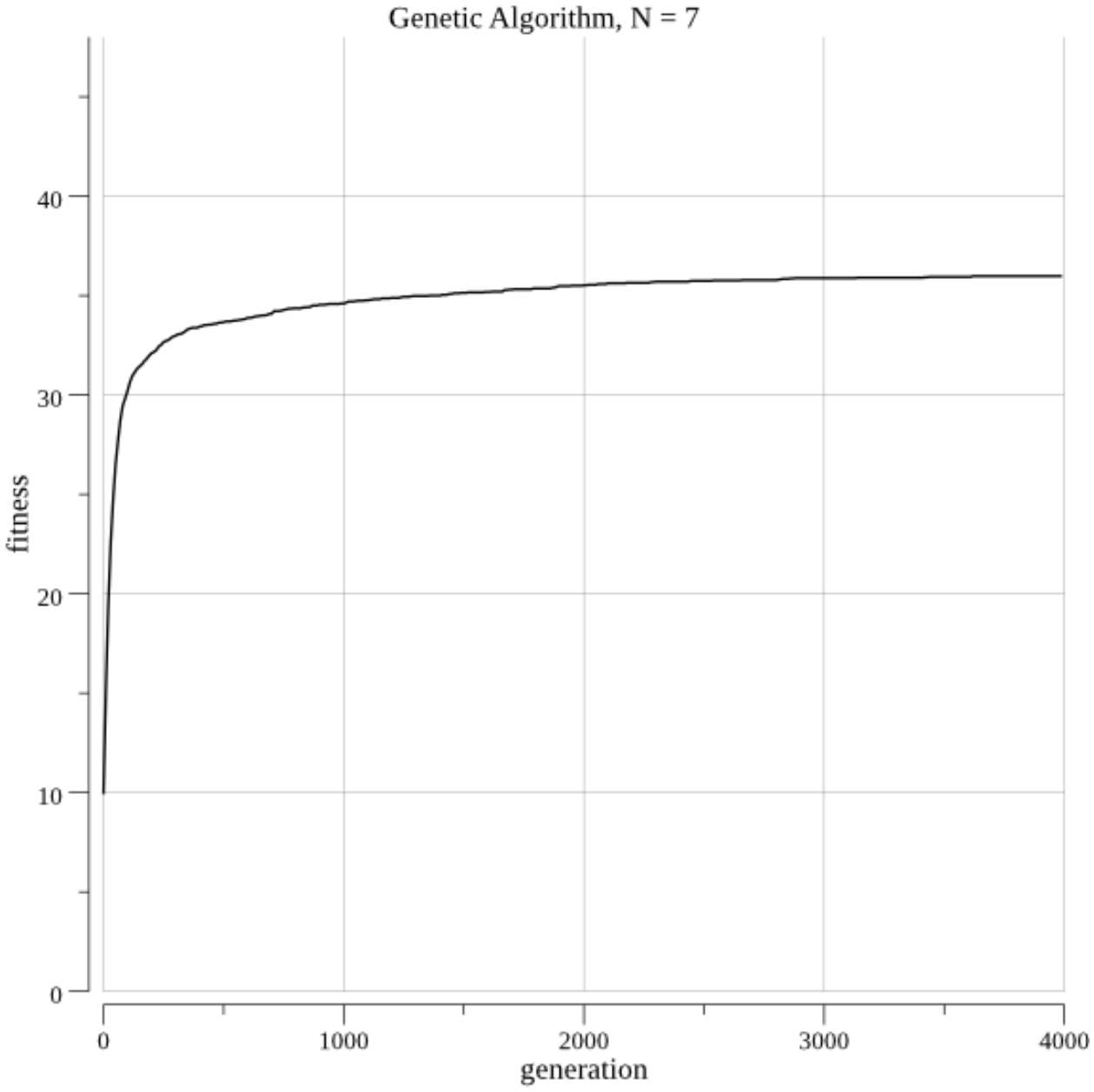

Figure 24: Fitness over number of generations for 7x7 puzzle

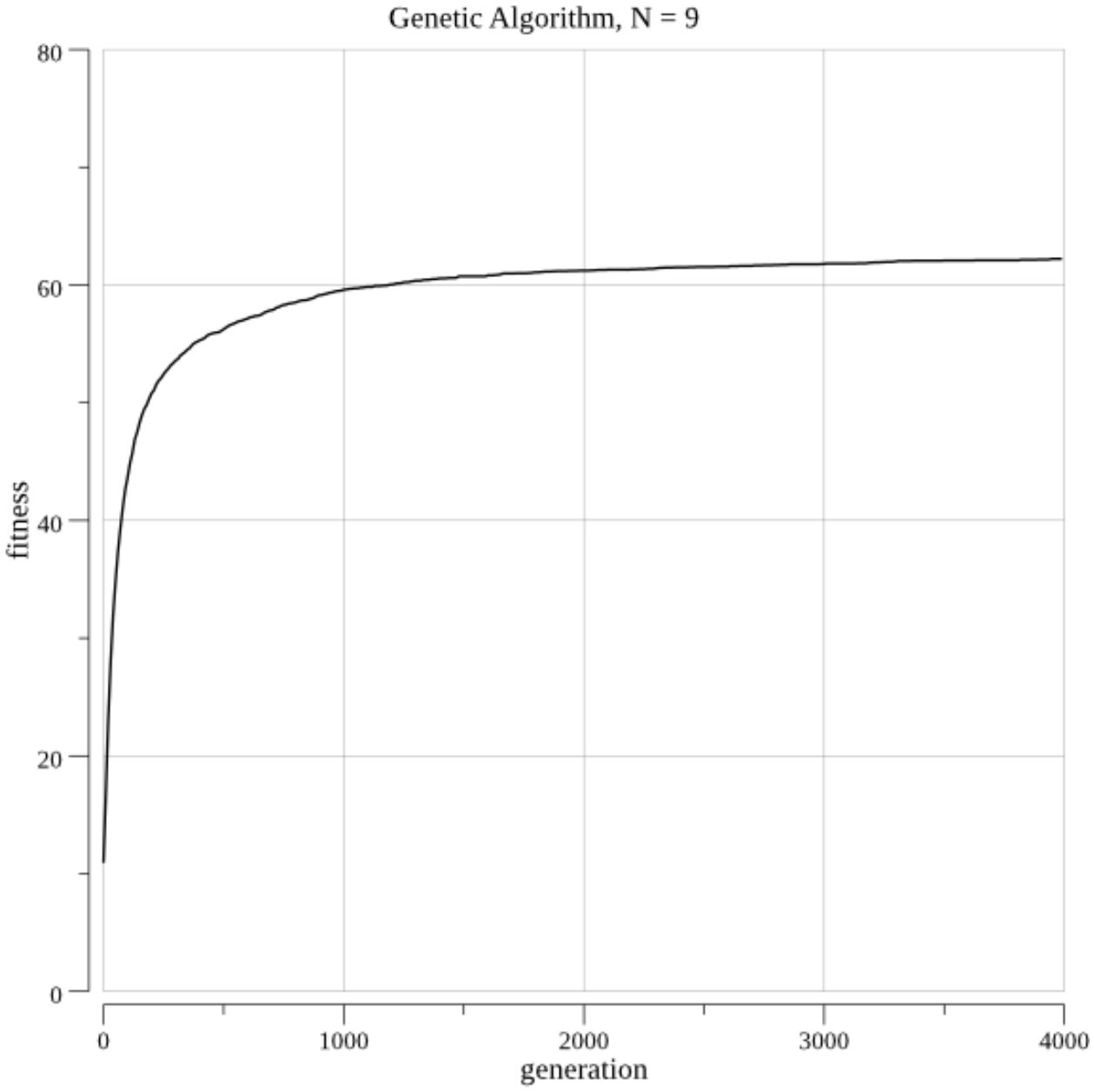

Figure 25: Fitness over number of generations for 9x9 puzzle

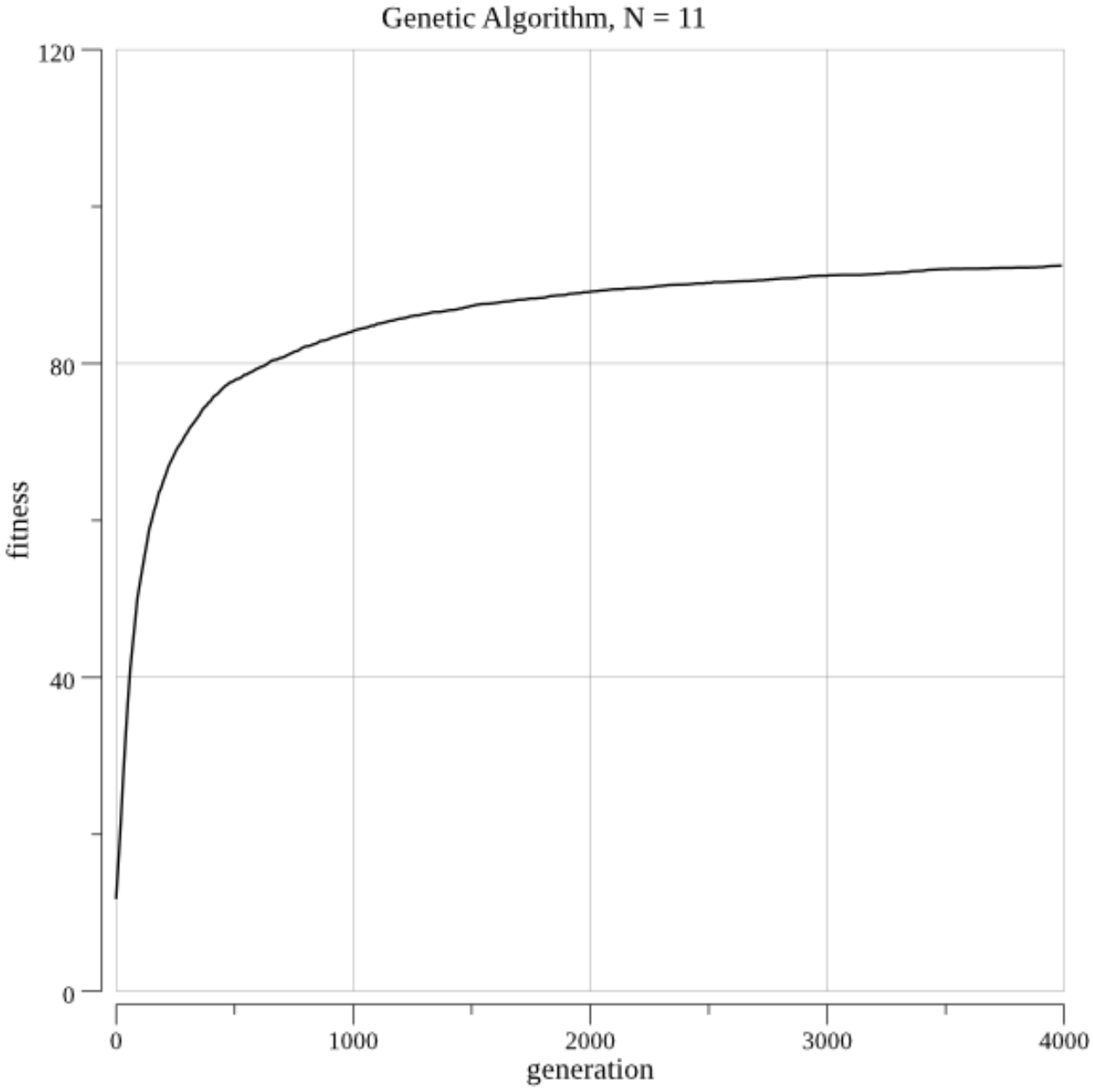

Figure 26: Fitness over number of generations for 11x11 puzzle

Task 8

n = 40 Puzzle

method: genetic algorithm
fitness: 942
generations: 160000
pop size: 800
survival rate: 0.3
mutation rate: 0.018
solution:

```
R R D U D U D U L D D U D R L R L R L L R L R L R L R L R L U R D R L R L R L R L R L R L
R L R L R L R L D U D U D R U L R U D U D R L R L U D U D U D U R L R L R L R L R D U D U
L R L R L R R L R L R L D U U D D U R D L R L R U L R D R L U L R L R L R L L R L R L R L
R U U L R L R L D U D U D U D U D U D L R L U D U D R U L R L R L L L D U D U D U R L R L R L
R D R L R L U R R L U D U D U D U D L U R L R L U R L R L R L R L R L U D R L U D U D U L D D R U
D L U R D U L D U U L L D R U L D R L U R R L R R L R L R L R D U L R L D U D U D U R L R L R
R D L R U L D U D L L D U R L R L R L R L R L U D D U D U U L R R D U D L U D U D U L U D
D U L R D D R U R L L R L R D D U R D L L R L U D R D L U L R L D D R D U D U D U L L D R
L R L R U U D U L U R D U L D D R L R L R L U D U U U D U D D U D R L R L R R R R L L D U
U D U D D D R L U D L R L L U D U U U U U D U U D U U D U D L U D R R R D L R L L R U R L U
D D R L L D U U D U D D D U R L R R R D L U D R U D U U R R L R L D D R R R L L R R R D U D
D U L D D D U D U D U R U L U L D R R L L U R L D U D U R L R L L R L R L R L R D L U R L
L U U R R R L U L D R L R L L R R R L L D U R R R L L R L R R R R L R L L R R U U D D L R
R R L D U U D D U D U D D R L R L R U D U D D D L U U D D D U U R D R R L U R D L L R U D
L L L R R L L L D D U D R L D L R L R R L L R L R R L U D U D L R U R D D L R L D U D U D
L L R U L D L R R U D U D D U L D U U D R L L R L R L L R R D D U D D D R D L R R L L R U
L D D U U D D U U D U R L L L R L L R R L L R L U R L L R R L R L R D U D U D D U D U R L
R L D U D U D U U U D U D D U R L L R R L U R R D U D U U D U L R L R L R R L U D R R U L
R L D D R U D U U D U R L R L R U D D U R L L R L R R D L L L U R R L L R L U R R L R L R
R R L U U L L R D U U U U D D U U U D D L R R L U R U L R L U U R U D D U U D U L L R R R
L L D R L R D U U D D U D U D D U D R R D L R R L U U U D R L D R L R L L R L R R R
```

6	16	13	25	15	13	20	38	34	4	38	3	21	31	28	35	25	33	30	31	5	25	15	22	12	16	32	5	33	33	26	37	25	12	4	9	4	31	9	19
5	24	14	25	33	5	5	19	2	29	16	2	12	36	20	18	14	26	1	15	30	4	23	31	37	19	19	8	1	8	29	15	36	37	24	11	1	6	32	38
16	17	8	13	8	21	28	4	16	33	7	33	11	7	35	6	32	20	22	13	31	10	23	26	29	32	6	14	29	28	33	31	24	7	11	15	14	33	38	12
9	4	14	3	9	30	14	13	21	27	5	34	36	19	2	36	19	18	11	25	35	14	34	5	2	32	35	33	3	14	8	30	1	2	36	36	30	10	6	31
7	11	31	24	29	5	34	16	5	25	9	34	5	27	10	28	29	15	32	34	12	30	34	11	8	6	16	33	1	8	18	16	28	14	8	7	6	22	3	37
14	2	12	30	21	18	33	4	28	16	31	1	13	27	33	3	34	33	10	30	16	14	31	26	27	29	29	23	14	33	10	2	29	33	34	15	35	5	33	6
34	22	5	35	10	18	23	16	27	3	12	26	17	32	11	25	21	28	32	30	23	1	10	31	33	11	27	15	29	31	33	33	8	7	34	17	15	22	31	27
8	9	30	6	23	34	14	28	11	31	32	5	30	3	27	13	30	23	1	18	13	32	24	1	17	31	31	2	25	23	20	7	7	26	30	29	26	31	19	39
35	30	12	6	34	12	21	18	25	17	25	28	10	27	29	27	18	14	10	25	26	30	30	25	1	4	25	13	2	31	30	27	12	1	11	26	35	17	33	37
32	22	2	6	30	8	26	31	13	11	2	14	13	19	5	5	10	2	8	29	29	28	11	8	23	25	7	13	18	1	25	5	32	25	10	13	20	34	34	24
28	34	26	34	6	32	3	4	31	4	19	26	29	3	21	7	5	9	9	22	11	14	11	18	24	22	11	6	11	7	7	2	32	12	9	32	12	15	22	36
20	2	9	8	10	34	28	23	16	21	28	22	23	4	26	11	1	28	19	21	24	4	23	7	12	25	18	8	24	12	1	10	3	6	21	30	20	21	5	29
28	2	16	31	5	26	7	31	3	22	22	5	12	21	22	9	22	23	10	12	5	5	24	25	27	26	4	4	22	14	18	22	18	1	34	1	12	13	1	38
11	38	32	16	15	33	24	20	30	1	16	3	24	13	22	22	12	24	26	23	23	9	26	2	13	23	19	22	21	18	8	12	15	31	29	29	34	35	28	7
23	28	34	4	24	30	25	24	10	24	8	23	24	14	9	4	11	18	22	22	2	6	18	17	24	22	7	4	18	27	21	25	8	10	29	34	23	34	22	20
32	3	28	16	26	14	24	29	29	16	10	13	10	24	6	11	21	4	12	21	5	19	11	20	5	21	18	5	21	15	26	22	1	2	20	4	36	30	20	22
13	29	33	25	27	24	31	21	27	30	25	16	10	14	2	3	6	10	13	6	17	23	18	17	8	23	13	2	17	26	17	26	21	17	8	29	4	28	27	17
36	32	30	32	35	23	25	16	5	16	28	5	23	25	23	21	19	2	7	5	11	7	21	18	18	19	26	5	24	23	3	24	32	27	30	34	33	35	24	23
5	7	31	35	29	29	21	4	24	24	7	4	23	24	8	21	10	5	7	15	18	9	19	9	17	20	5	24	18	5	13	18	29	1	25	31	25	22	34	33
28	7	12	27	16	30	25	17	28	3	7	25	16	15	10	20	5	20	3	1	15	9	15	21	14	2	14	22	22	23	22	2	2	30	29	35	24	31	16	33
19	23	19	23	9	29	29	19	16	30	21	10	19	22	13	1	22	9	6	3	11	5	7	10	23	22	21	20	18	26	25	22	3	5	31	22	1	12	31	34
30	32	8	27	10	8	33	10	23	27	25	12	20	20	14	7	16	16	6	1	20	1	14	22	22	12	4	19	27	4	24	30	29	32	31	28	27	19	11	23
28	30	5	26	28	21	8	30	27	26	4	27	5	26	15	9	14	14	12	18	15	6	15	17	8	8	26	20	23	25	13	28	32	24	22	24	23	29	2	38
23	28	24	13	1	7	16	20	20	11	25	18	7	5	2	19	19	7	9	9	13	23	21	17	1	6	12	19	7	13	6	27	15	4	1	34	15	3	25	4
34	27	1	34	4	30	14	32	1	29	11	24	21	19	16	2	22	12	21	10	3	20	15	8	23	2	25	4	27	26	12	26	30	32	29	26	31	26	37	32
19	15	25	16	19	11	28	27	28	23	1	20	8	3	9	25	23	21	25	21	6	21	23	17	17	20	23	26	22	3	15	28	31	32	19	12	11	36	7	33
10	5	37	23	30	30	8	1	7	16	21	20	19	24	24	12	5	10	9	12	20	9	10	3	9	11	5	5	23	8	4	31	10	5	31	33	8	17	19	21
1	7	28	26	21	23	10	29	6	11	21	6	17	9	20	11	17	26	27	2	7	21	22	7	11	7	2	25	10	17	13	20	16	6	25	20	9	18	35	2
38	15	30	14	6	20	22	14	11	13	28	23	17	22	27	2	8	24	28	9	14	21	16	20	11	14	15	10	5	7	20	22	23	7	29	17	15	14	15	9
33	23	14	7	4	5	22	32	23	3	24	18	15	17	6	12	27	21	28	22	13	10	11	1	13	26	22	28	7	19	15	29	14	8	32	21	7	7	12	17
28	23	30	4	5	25	27	7	6	28	22	6	20	11	17	25	30	24	30	20	11	5	19	30	18	11	10	30	8	23	9	9	8	24	34	31	17	35	11	33
25	32	17	31	28	28	7	7	9	19	12	20	6	17	16	28	21	22	27	10	30	9	28	25	19	16	29	26	29	18	12	7	17	32	11	12	15	17	34	8
11	22	25	32	27	7	21	16	25	17	30	13	24	3	6	29	20	14	16	24	16	16	2	25	26	8	29	5	10	7	26	17	13	6	28	5	15	24	20	11
2	36	11	15	35	23	19	32	17	14	6	16	23	31	27	21	14	4	26	27	14	5	10	32	20	17	32	20	31	29	26	12	29	32	9	13	10	28	21	11
33	8	6	13	19	27	29	19	21	29	31	29	1	16	6	1	34	4	3	15	33	25	32	28	28	17	13	21	28	29	17	32	28	25	32	18	5	2	17	34
16	6	17	19	32	23	8	18	28	17	4	14	15	13	34	29	23	1	26	3	22	31	19	30	2	30	35	32	19	6	30	16	8	8	21	18	7	10	23	13
12	1	27	10	32	1	34	14	25	17	31	17	23	30	9	25	29	31	34	6	14	7	18	2	12	31	12	17	36	2	5	33	19	28	1	18	23	28	31	29
29	7	5	5	19	19	25	17	7	10	7	21	32	30	30	12	32	24	32	18	2	17	29	31	35	21	18	2	11	18	32	15	18	15	6	1	7	31	13	7
2	13	20	27	25	23	23	21	11	20	6	18	10	29	25	29	9	31	22	29	30	19	33	25	37	25	32	29	22	35	15	23	6	13	25	5	15	33	22	17
15	32	28	33	17	36	25	33	29	8	34	17	38	18	4	26	18	18	8	8	22	25	38	13	38	34	13	20	14	7	33	39	23	14	10	37	38	36	28	G

Figure 27: 40x40 puzzle generated by genetic algorithm with fitness 942

0	915	181	297	862	813	1	631	137	812	474	136	623	212	137	71	65	916	162	863	810	307	680	914	695	811	2	679	144	685	861	140	630	624	860	X	629	276	861	809
32	530	529	787	525	31	532	9	X	534	32	533	105	406	139	X	528	329	523	524	144	790	672	263	61	531	8	664	788	789	529	527	X	273	140	665	708	526	535	X
374	482	725	294	480	X	257	903	334	179	477	425	481	256	487	617	63	478	218	X	232	618	130	259	225	X	332	371	51	X	X	373	333	X	258	231	709	479	X	X
301	348	X	300	594	228	177	221	450	302	777	137	103	595	X	75	387	X	220	873	178	X	147	X	X	343	4	88	698	115	751	384	596	597	X	229	303	934	752	X
785	638	241	237	154	784	X	633	236	813	637	431	632	237	499	318	383	240	216	123	235	121	682	634	500	889	636	139	X	687	631	558	236	155	635	X	630	688	X	X
168	347	X	277	358	436	254	549	345	374	85	430	108	847	169	616	57	191	344	X	357	548	151	50	900	109	X	494	345	113	356	X	276	167	X	549	346	275	X	255
1	928	X	301	543	32	511	12	517	510	768	392	440	156	544	73	X	195	519	172	414	547	548	13	138	545	6	X	254	441	367	X	278	178	2	518	546	393	166	644
465	348	729	X	596	463	269	267	466	X	319	424	337	408	141	X	55	193	827	467	270	310	145	265	266	598	541	597	X	598	352	782	730	271	X	268	X	468	488	464
283	287	721	293	289	291	X	X	829	285	778	X	720	X	372	617	389	292	828	170	170	313	149	261	887	888	286	371	829	791	658	288	581	760	390	284	306	713	290	X
160	555	728	299	552	367	560	550	116	300	318	428	319	158	498	317	559	X	604	125	272	117	674	556	554	320	558	497	791	790	366	557	159	115	553	X	558	413	551	X
326	145	726	141	481	230	843	140	315	509	410	141	322	409	504	314	392	328	606	175	508	139	143	751	325	890	329	138	327	323	750	509	X	139	X	140	324	142	X	316
753	641	724	642	591	162	745	805	835	748	164	420	819	747	592	833	832	403	830	168	754	751	445	909	836	752	834	167	X	581	749	750	580	166	746	161	837	831	165	163
251	248	X	249	595	458	253	869	933	372	775	423	622	254	627	612	245	845	X	868	764	619	621	371	183	765	620	370	252	688	621	371	626	625	250	666	628	247	246	247
376	642	79	X	471	81	668	85	112	X	83	377	106	410	171	77	X	245	X	472	756	909	670	908	694	186	84	86	X	691	669	X	644	179	80	667	107	78	82	643
281	207	209	239	765	205	139	141	X	355	767	203	816	211	499	313	386	X	221	314	763	312	764	138	280	656	X	137	766	208	354	138	279	356	204	206	210	278	X	X
397	639	62	295	64	460	145	395	507	400	X	682	X	403	506	508	393	402	345	296	507	120	681	635	683	401	509	X	X	684	63	399	398	399	X	400	396	394	X	401
339	349	122	133	129	131	124	222	237	126	504	135	439	340	502	884	503	585	521	525	236	718	128	845	886	121	559	136	134	132	350	130	335	X	X	123	710	125	166	127
39	43	37	41	51	49	45	47	932	762	33	422	321	933	35	315	53	X	605	867	761	364	370	48	696	111	331	369	50	116	368	46	38	44	X	42	40	36	34	52
375	640	188	114	111	376	486	203	335	114	769	202	109	198	503	200	X	770	829	872	X	X	153	910	691	830	X	137	X	690	X	X	113	112	113	110	201	199	115	X
93	483	897	98	899	91	95	902	100	276	774	137	102	342	898	608	385	241	607	871	234	308	901	896	899	895	X	90	94	97	99	96	140	97	141	92	101	242	900	657
X	434	722	12	590	10	2	10	X	106	915	433	104	4	677	676	120	X	X	866	271	119	675	912	435	532	9	678	914	583	369	105	582	913	11	3	711	712	19	107
60	98	801	61	592	X	62	804	145	146	802	427	624	405	140	613	59	243	799	865	358	X	144	428	800	797	330	372	141	796	61	97	60	99	X	803	145	798	X	63
28	22	386	24	26	30	609	13	15	141	768	17	320	20	610	834	388	293	608	526	758	840	X	915	586	893	31	370	29	25	365	23	27	140	859	16	19	14	18	21
376	380	849	589	588	437	378	447	449	854	440	429	438	852	500	X	382	584	853	439	855	306	379	377	585	586	850	448	440	381	660	587	583	856	858	383	305	857	851	X
188	20	848	846	589	33	842	221	451	35	771	435	623	846	X	845	386	844	219	869	843	140	673	844	267	892	924	893	21	845	X	695	847	357	189	34	629	674	19	220
121	144	849	298	542	455	68	490	452	286	770	771	841	X	501	70	66	X	830	122	842	120	131	502	X	454	331	850	939	794	X	772	287	169	69	X	453	143	X	67
338	484	700	361	359	698	485	489	490	703	339	391	336	411	486	491	390	586	702	488	413	363	335	917	695	704	362	492	697	689	364	337	334	696	360	699	705	412	487	701
569	570	387	786	577	783	562	490	571	574	780	782	321	576	572	883	563	330	346	567	575	568	577	911	885	578	884	492	784	322	353	781	579	564	573	882	491	566	785	565
310	929	873	238	472	654	648	X	931	646	473	652	818	159	138	614	930	239	161	874	762	309	647	260	931	655	651	495	649	795	657	X	645	650	653	160	600	394	311	656
340	927	61	663	541	461	512	416	516	814	84	419	815	715	515	662	62	538	522	568	415	514	418	918	919	342	540	663	513	420	661	515	X	341	60	514	559	920	539	417
447	877	180	445	597	435	176	446	878	178	X	434	439	341	447	615	367	194	163	174	442	366	444	913	182	894	367	X	441	599	365	443	181	177	X	X	212	179	X	175
688	229	898	296	593	227	841	805	X	401	773	693	840	213	505	74	391	X	215	864	145	839	146	51	226	689	363	493	52	692	214	694	594	228	297	X	838	277	695	X
X	X	731	880	735	718	744	X	738	749	476	136	719	714	737	319	715	892	742	169	734	717	733	264	137	891	3	732	741	793	882	736	183	739	743	881	716	713	742	740
687	273	385	825	155	229	234	48	236	275	779	421	821	255	678	611	384	537	826	171	233	538	823	262	693	235	7	422	50	686	351	374	824	272	234	822	304	274	536	49
113	571	872	446	278	462	486	806	111	373	776	821	820	197	142	610	64	196	609	870	143	121	129	279	326	110	807	X	253	112	367	372	277	114	871	285	706	X	374	808
356	926	242	146	598	457	561	268	X	180	772	426	103	715	141	72	244	602	603	243	755	364	145	49	692	269	181	87	255	600	355	289	X	651	759	601	599	480	901	245
187	925	727	590	153	456	457	903	879	922	86	142	188	155	905	X	54	190	217	173	757	904	152	906	X	112	923	89	143	382	X	383	335	759	758	189	154	921	924	X
282	876	730	879	119	457	563	223	877	X	X	138	107	407	140	878	56	244	343	875	759	118	148	14	224	120	X	140	760	141	X	141	139	342	759	882	211	876	119	402
186	288	187	X	470	459	144	142	696	356	475	432	817	157	170	316	184	192	520	124	169	307	150	907	62	185	5	496	143	114	659	695	182	168	X	883	306	469	183	171
X	934	723	X	553	719	140	11	932	X	165	937	104	404	939	76	58	242	940	936	760	311	671	916	60	122	941	141	938	792	366	139	X	935	59	230	707	933	X	942

Figure 28: 40x40 puzzle generated by genetic algorithm with fitness 942 (BFS process)

n = 20 Puzzle

method: genetic algorithm
fitness: 323
generations: 40000
pop size: 400
survival rate: 0.3
mutation rate: 0.018
solution:

```
D R D U D U D U D U R D U D U D U D L U R L R L R L R U D U L R L U D U R L D U U D U D R
L U R L R L D R U R L D D U D U R D R D L R L R L R R U L D U U D U D U U L R L U D R U U
U D L R L R L R L U D U U D D U U D D U R L R L L R L R L R L R D U D U D U L R R L R L R
U D U D D L U D U D U D D U D D D U D L U R L L L R L R L R U D U L R L R R R U D U D D L
U R U D U D L R U L U R D U D L L R L R L R L R L L R L R L U D U R L L U R R D U D D L R
U L L R L D U U D U D U R L R D L R R R L R L D U U D U L R L D U L R L U D R L R L L R L
R U D U L R U R L R D U U D D U D U D L U U D R L U R L R R L L R D L R L U R L D U D D U
D D U R R D U D
```

6	16	11	6	13	18	16	15	16	3	1	11	15	18	17	1	15	15	18	17
14	13	15	17	1	3	9	2	9	14	17	6	16	18	17	1	16	8	12	18
19	10	16	3	4	7	15	16	3	7	8	17	17	15	5	16	6	17	14	13
8	5	7	8	12	7	4	14	12	16	9	16	13	11	13	8	13	4	16	6
14	18	14	9	14	15	2	2	2	14	12	9	10	15	3	4	3	14	15	11
10	2	16	10	11	12	2	11	14	7	2	3	11	5	5	10	4	11	10	1
9	14	4	13	12	3	10	3	12	11	13	6	8	12	6	13	16	8	3	5
8	13	16	10	15	4	10	7	7	9	6	7	11	9	8	9	14	10	18	4
12	10	16	11	2	9	2	1	10	10	1	11	10	2	10	15	4	5	17	12
19	16	13	9	6	7	7	10	11	7	5	9	10	7	11	14	12	17	1	13
16	8	9	12	13	3	5	10	9	3	2	11	7	13	9	1	14	4	11	16
17	17	11	10	6	3	13	5	3	5	6	9	4	7	8	9	7	14	11	19
12	5	17	10	10	12	1	6	7	5	7	3	3	10	12	15	9	13	10	1
17	2	7	16	12	13	9	11	5	10	10	7	7	8	12	10	15	8	10	11
5	13	13	1	12	14	12	2	4	8	14	9	12	9	11	9	8	13	14	11
18	18	11	6	1	1	4	13	15	11	6	14	14	9	15	9	14	7	18	19
8	12	11	2	6	16	14	6	11	11	10	7	14	7	14	10	8	7	14	19
13	11	17	5	7	11	17	13	14	16	8	15	14	7	16	9	2	13	10	19
16	3	9	6	16	17	5	17	17	15	4	2	17	17	5	15	9	16	17	15
10	12	10	12	18	18	17	14	2	19	13	17	10	11	8	13	13	13	13	G

Figure 29: 20x20 puzzle generated by genetic algorithm with fitness 323

6	16	11	6	13	18	16	15	16	3	1	11	15	18	17	1	15	15	18	17
14	13	15	17	1	3	9	2	9	14	17	6	16	18	17	1	16	8	12	18
19	10	16	3	4	7	15	16	3	7	8	17	17	15	5	16	6	17	14	13
8	5	7	8	12	7	4	14	12	16	9	16	13	11	13	8	13	4	16	6
14	18	14	9	14	15	2	2	2	14	12	9	10	15	3	4	3	14	15	11
10	2	16	10	11	12	2	11	14	7	2	3	11	5	5	10	4	11	10	1
9	14	4	13	12	3	10	3	12	11	13	6	8	12	6	13	16	8	3	5
8	13	16	10	15	4	10	7	7	9	6	7	11	9	8	9	14	10	18	4
12	10	16	11	2	9	2	1	10	10	1	11	10	2	10	15	4	5	17	12
19	16	13	9	6	7	7	10	11	7	5	9	10	7	11	14	12	17	1	13
16	8	9	12	13	3	5	10	9	3	2	11	7	13	9	1	14	4	11	16
17	17	11	10	6	3	13	5	3	5	6	9	4	7	8	9	7	14	11	19
12	5	17	10	10	12	1	6	7	5	7	3	3	10	12	15	9	13	10	1
17	2	7	16	12	13	9	11	5	10	10	7	7	8	12	10	15	8	10	11
5	13	13	1	12	14	12	2	4	8	14	9	12	9	11	9	8	13	14	11
18	18	11	6	1	1	4	13	15	11	6	14	14	9	15	9	14	7	18	19
8	12	11	2	6	16	14	6	11	11	10	7	14	7	14	10	8	7	14	19
13	11	17	5	7	11	17	13	14	16	8	15	14	7	16	9	2	13	10	19
16	3	9	6	16	17	5	17	17	15	4	2	17	17	5	15	9	16	17	15
10	12	10	12	18	18	17	14	2	19	13	17	10	11	8	13	13	13	13	G

Figure 30: 20x20 puzzle generated by genetic algorithm with fitness 323 (BFS process)