Computational Intelligence - Excercise 2
Tel-Hai Academic College

Mor Tel Tsur — 308465582
Tom Shalit — 204182992

1 Non-Trivial GA steps

In our solution to the N-Queens Problem using a Genetic Algorithm, we decided to use a representation of the problem using permutations, thus avoiding rows and columns collisions. Our fitness function calculates diagonal collisions, and we search for $f_{min} = 0$, meaning 0 diagonal collisions. In our code we also refer to the following issues:

1.1 Crossover Permutations

We read a couple of articles regarding solving the N-Queens problem, and implemented 3-way tournament [1] and another suggested permutation crossover function [2], and after many tests using these crossovers, we eventually used "Order One" permutation, in which when given 2 parents, copy a random sub-chromosome from first parent, and "complete" the missing numbers in the permutation by the order they appear in the second parent [3].

1.2 Mutation Operator

After experimenting with 3 types of mutations [3]:

1. Inversion(x) - Pick two alleles at random and then invert the sub-string between them.

2. Insert(x) - Pick two allele values at random, and move the second to follow the first, shifting the rest along to accommodate.

3. Swap(x) - Pick two alleles at random and swap their positions. In our solution, we randomly choose between 1-3 swaps as a single mutation.

According to the results, we decided to continue using ***Swap*** mutation.

1.3 GA Parameters

1. Population Size $\mu = 100$. A decision made after trying different population sizes: 50, 100, 200, 1000

2. Probability for Crossover $P_c = 0.37$. To be honest we did not play with this one much, just used the initial value in original GA code from the lab.

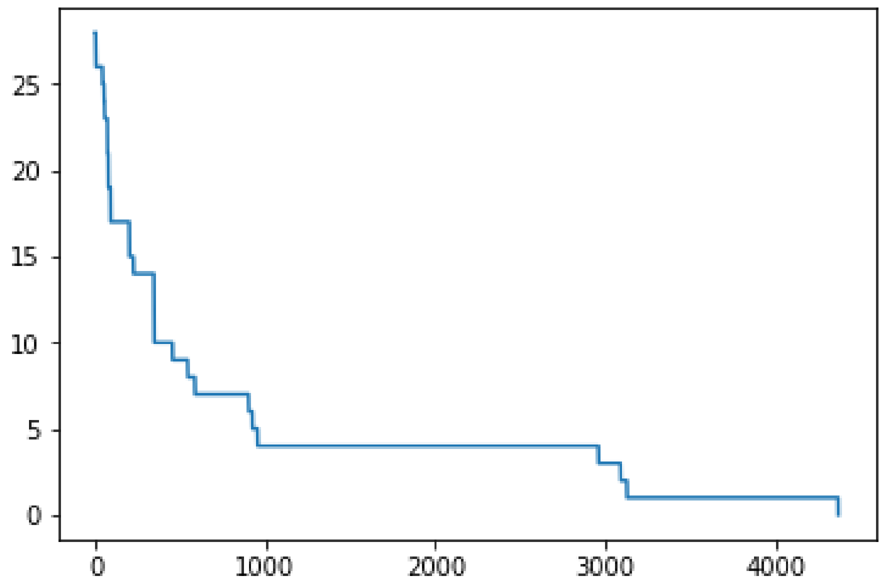
3. Probability for Mutation $P_m = 1$. The mutation operator was recorded to be the meaningful help to actually solve the problem [1], and therefore is always being used.

2 Results

A table describing 2 different solutions for each of the different board-dimensions required:

N	Iterations	Solution
64	787500	[27 15 21 36 40 59 24 29 43 48 58 49 3 39 11 22 1 18 47 50 33 51 62 31 8 23 35 2 34 5 17 14 44 0 55 60 6 57 53 63 45 37 46 12 61 38 4 20 26 9 30 28 54 10 19 16 42 25 52 41 32 56 13 7]
64	437000	[42 5 46 41 19 30 52 18 56 9 43 38 8 21 23 13 63 0 34 62 11 1 40 59 50 44 49 2 27 10 33 16 61 57 54 25 29 3 45 31 17 55 36 7 22 14 58 20 32 51 4 48 53 39 60 6 35 28 15 47 12 37 24 26]
32	219200	[17 2 27 11 15 10 5 1 28 13 20 29 26 8 4 22 31 14 9 7 23 3 6 21 24 0 19 12 30 25 16 18]
32	116100	[30 9 0 23 14 31 18 6 27 5 13 26 28 15 3 1 4 17 29 12 21 25 7 2 19 8 16 11 20 10 24 22]
16	62900	[10 4 11 8 2 12 14 7 0 3 5 15 9 6 13 1]
16	56800	[6 12 0 5 13 9 1 14 7 3 15 11 8 10 2 4]
8	1500	[4 1 3 5 7 2 0 6]
8	800	[4 6 1 5 2 0 3 7]

The best run over $N = 64$ is described in the following graph:



References

[1] L. B. Marko Bozikovic, Marin Golub, "Solving n-queen problem using global parallel genetic algorithm," *EUROCON Ljubljana, Slovenia*, 2003.

[2] S. N. Uddalok Sarkar, "An adaptive genetic algorithm for solving nqueens problem," *Department of Electrical Engineering, Jadavpur University, Kolkata, India*, 2017.

[3] O. Shir, "Lecture 6, computational intelligence," *Tel-Hai College*, 2018.