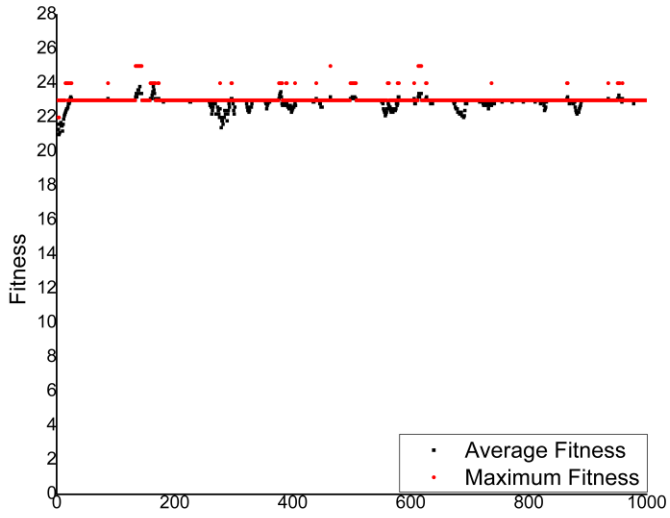
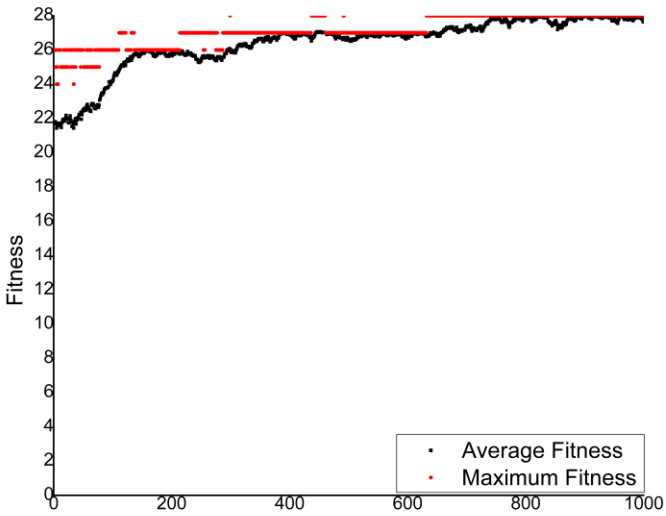
Austen Hsiao
Programming 2 CS541

Results

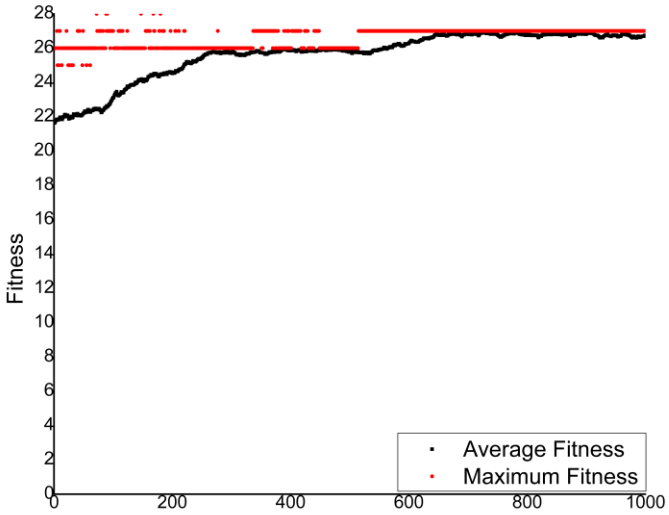
10 Population/ 1000 Generations



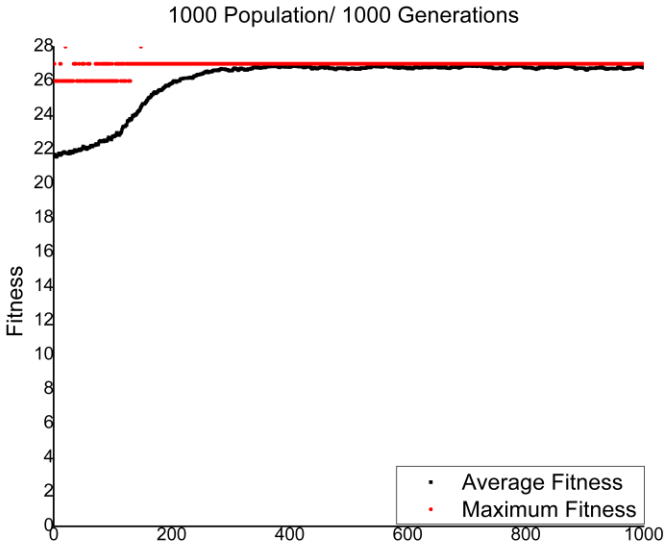
100 Population/ 1000 Generations



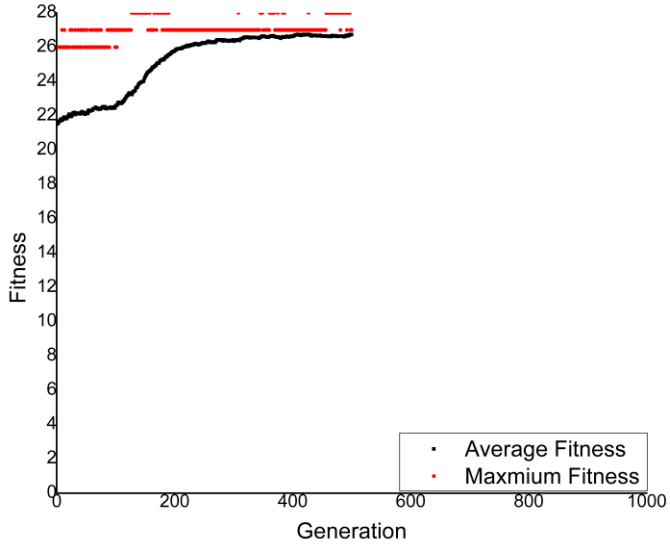
500 Population/ 1000 Generations



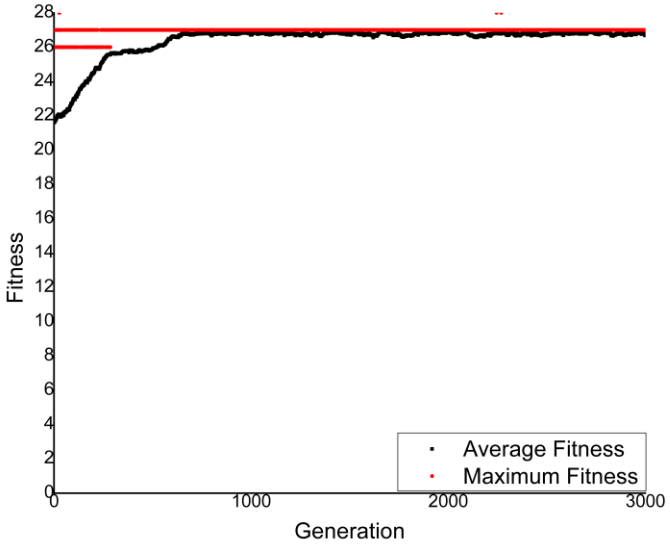
1000 Population/ 1000 Generations



1000 Population/ 500 Generations



1000 Population/ 3000 Generations



Each graph of fitness v generation shows average- and maximum fitness for each generation with different population numbers and/or different iteration numbers.

Implementation

My fitness function uses the maximum number of distinct pairs of queens as the max possible fitness and subtracts the number of non-distinct pairs of queens. Every chromosome is generated as a list of 8 values between 0-7. Every index represents a column on a chess board and the number at the index stores the row number.

Selecting a parent uses the fitness as a weight for selecting randomly. We create "populationNumber" number of children by randomly selecting parents then splicing them together at a random crossover point. There is a 1% chance for a mutation to occur at a random position.

Summary

Constrained Iterations

When the number of iterations is kept at 1000, a population of 10 causes no appreciable change in the average fitness, hovering around 23.

With a population of 100, each subsequent generation quickly increases from a minimum ~22 to a maximum of ~28 within 800 generations.

When the population is increased to 500, the rise is less pronounced and reaches an average maximum of ~27 but within 600 generations. This may be caused by randomly generated less-fit parents, which makes it harder to generate perfect chromosomes later on.

With a population of 1000, the maximum average fitness still does not reach 28 as in the case of a population of 10. The maximum of ~27 is reached in ~300 generations—faster than the population 500 case.

Constrained Populations

Constraining the size of the population does nothing for the overall trend, but gives the genetic algorithm more/less time to reach a maximum.

In all cases, the average fitness converged at a maximum of ~27. There are some cases where the maximum fitness is not reached, and I think this is caused by a bad set of starting chromosomes that makes it hard for the algorithm to get to a solution.

Examples—these examples show one set of parents and subsequent child throughout the algorithm. The average fitness is also reported for the set of children.

10 Population- 1000 Generations

Generation 1: [6, 3, 6, 5, 3, 4, 5, 6] + [3, 7, 5, 1, 7, 3, 1, 6] => [6, 3, 6, 1, 7, 3, 1, 6]; Ave. Fitness: 21.1
Generation 200: [6, 4, 5, 1, 7, 3, 1, 6] + [6, 4, 5, 1, 7, 3, 1, 6] => [6, 4, 5, 1, 7, 3, 1, 6]; Ave. Fitness: 22.0
Generation 400: [6, 4, 7, 1, 7, 3, 6, 6] + [6, 4, 7, 1, 7, 3, 6, 6] => [6, 4, 7, 1, 7, 3, 6, 6]; Ave. Fitness: 23.0
Generation 600: [6, 4, 7, 1, 7, 3, 6, 6] + [6, 4, 7, 1, 7, 3, 6, 6] => [6, 4, 7, 1, 7, 3, 6, 6]; Ave. Fitness: 23.0
Generation 800: [6, 4, 7, 1, 7, 3, 6, 6] + [6, 4, 7, 1, 7, 3, 6, 6] => [6, 4, 7, 1, 7, 3, 6, 6]; Ave. Fitness: 23.0
Generation 999: [6, 4, 7, 1, 7, 3, 6, 6] + [6, 4, 7, 1, 7, 3, 6, 6] => [6, 4, 7, 1, 7, 3, 6, 6]; Ave. Fitness: 23.0

100 Population- 1000 Generations

Generation 1: [2, 2, 3, 3, 0, 4, 3, 0] + [0, 0, 7, 7, 0, 5, 1, 2] => [2, 2, 3, 3, 0, 4, 1, 2]; Ave. Fitness: 21.33
Generation 200: [7, 3, 0, 5, 3, 1, 6, 4] + [7, 5, 2, 5, 0, 4, 6, 4] => [7, 3, 0, 5, 3, 1, 6, 4]; Ave. Fitness: 23.58
Generation 400: [7, 5, 0, 5, 3, 1, 6, 4] + [7, 5, 0, 5, 3, 1, 6, 4] => [7, 5, 0, 5, 3, 1, 6, 4]; Ave. Fitness: 24.92
Generation 600: [7, 5, 0, 5, 1, 1, 6, 4] + [7, 5, 0, 5, 3, 1, 6, 4] => [7, 5, 0, 5, 1, 1, 6, 4]; Ave. Fitness: 24.87
Generation 800: [7, 5, 0, 5, 3, 1, 6, 4] + [7, 5, 0, 5, 3, 1, 6, 4] => [7, 5, 0, 5, 3, 1, 6, 4]; Ave. Fitness: 25.0
Generation 999: [7, 5, 0, 5, 3, 1, 6, 4] + [7, 4, 0, 5, 3, 1, 6, 4] => [7, 5, 0, 5, 3, 1, 6, 4]; Ave. Fitness: 25.41

500 Population- 1000 Generations

Generation 1: [3, 4, 5, 0, 3, 0, 3, 5] + [6, 4, 5, 1, 4, 3, 7, 1] => [3, 4, 5, 1, 4, 3, 7, 1]; Ave. Fitness: 21.514
Generation 200: [4, 1, 3, 6, 2, 7, 5, 0] + [4, 1, 3, 6, 2, 7, 5, 0] => [4, 1, 3, 6, 2, 7, 5, 0]; Ave. Fitness: 27.516
Generation 400: [4, 1, 3, 6, 2, 7, 5, 0] + [4, 1, 3, 6, 2, 7, 5, 0] => [4, 1, 3, 6, 2, 7, 5, 0]; Ave. Fitness: 27.744
Generation 600: [4, 1, 3, 6, 2, 7, 5, 0] + [4, 1, 3, 6, 2, 7, 5, 0] => [4, 1, 3, 6, 2, 7, 5, 0]; Ave. Fitness: 27.648
Generation 800: [4, 1, 3, 6, 2, 7, 5, 0] + [4, 1, 3, 6, 2, 7, 5, 0] => [4, 1, 3, 6, 2, 7, 5, 0]; Ave. Fitness: 27.848
Generation 999: [4, 1, 3, 6, 2, 7, 5, 0] + [4, 1, 3, 6, 2, 7, 5, 0] => [4, 1, 3, 6, 2, 7, 5, 0]; Ave. Fitness: 27.754

1000 Population- 1000 Generations

Generation 1: [4, 6, 3, 6, 6, 1, 1, 1] + [2, 5, 3, 5, 1, 7, 3, 3] => [4, 6, 3, 6, 1, 7, 3, 3]; Ave. Fitness: 21.575
Generation 200: [1, 1, 7, 2, 0, 6, 1, 5] + [3, 5, 7, 2, 0, 6, 1, 5] => [1, 1, 7, 2, 0, 6, 1, 5]; Ave. Fitness: 24.991
Generation 400: [3, 1, 7, 2, 0, 6, 1, 7] + [3, 1, 4, 2, 0, 6, 1, 5] => [3, 1, 7, 2, 0, 6, 1, 5]; Ave. Fitness: 26.815
Generation 600: [3, 1, 7, 2, 0, 6, 1, 5] + [3, 1, 7, 2, 0, 6, 1, 5] => [3, 1, 7, 2, 0, 6, 1, 5]; Ave. Fitness: 26.82
Generation 800: [3, 7, 4, 2, 0, 6, 1, 5] + [3, 7, 4, 2, 0, 6, 1, 5] => [3, 7, 4, 2, 0, 6, 1, 5]; Ave. Fitness: 27.421
Generation 999: [3, 7, 4, 2, 0, 6, 1, 2] + [3, 7, 4, 2, 0, 6, 1, 5] => [3, 7, 4, 2, 0, 6, 1, 5]; Ave. Fitness: 27.738