Genetic Algorithm: The Eight Queens Problem

Bingling Fu

Final Draft: April 14, 2018

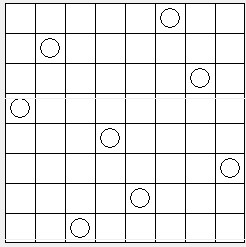
**Problem**

I created a genetic algorithm to find a solution to the Eight Queens Problem. The problem poses the situation: The Eight Queens question is a question on the background of chess: how can eight queens be placed on an 8×8 chess board so that no queen can directly eat other queens? In order to achieve this goal, neither of the two queens can be in the same horizontal, vertical or diagonal line.

**Design**

*Genetic code and expression:* With multi-value coding, the length of the chromosome depends on the number of queens. The position of each gene in the chromosome indicates the number of rows in the checkerboard, and the gene value indicates the number of columns.

It can be seen as a gene. This code can naturally solve the constraint that only one queen can be placed in a row. If every element is not repeated, it can be regarded as an arrangement of 0 n-1. It is natural to ensure that each column has only one queen. Therefore, it is necessary to pay attention to the uniqueness of list in cross mutation and generation of individuals. For example, the solution is 46031752, it means like that:

****

*Fitness function:* Since I am evaluating for the optimal solution, By comparing the Queen's position, comparing the difference between the number of rows and the number of columns of the two pieces, Math.abs(y2 - y1) == (x2 - x1) The two queens attack each other once, and the value decreases by 1. When the value of 1 At 28, there is no mutual attack. The chromosome at this time is the optimal solution. In general, the greater the fitness, the better. The fitness function is bestfitness=value. For multiple degrees of fitness, the average fitness is increased and the accuracy is increased.

*Crossing Over:* Since the individuals of the new population are randomly generated, the adjacent (and optionally randomly paired) individuals are chosen as parents to cross. Different from the bit-coded crossover, in order to ensure that the peers do not conflict with each other, the numbers in the new individual cannot be duplicated. First random generation of a single intersection position, Intercepted genes are exchanged and then stitched together

*Mutation:* Randomly generate two gene positions, exchange genes at two positions and mutate individuals. I am doing swap mutations of randomly selected bases. For example:

Before mutation 76543210

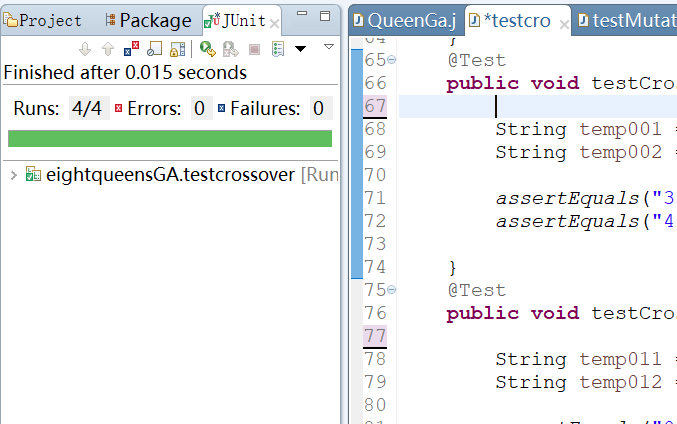
After mutation 71543260

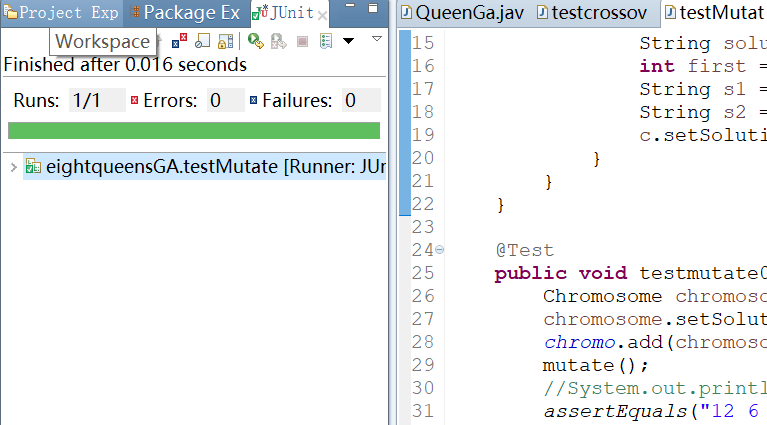
Which version of mutation is performed on a child is selected at random.

*Evolution:* The population is stored as a doubly linked list.

**Results**

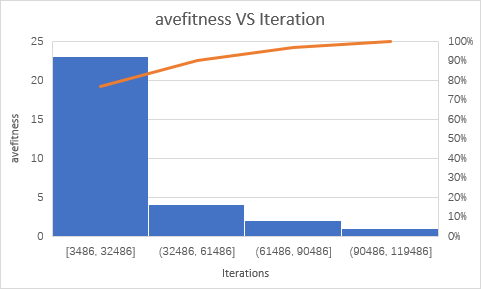
My project builds and passes my unit test cases:





*Conclusion:*

One effect is the size of the population at runtime and the solution. As the population increases, the algorithm takes longer. With the iteration increased, the average fitness increased. there was a positive relationship between number of generations and the average of fitness.



The cost time with the iteration show the linear relationship. The correlation is 0.9695.