Equation Solver

Jonathan Wenger

1. The way I designed a GA approach to this problem is that a chromosome was an instance of the “Solution” inner class, and each instance of a “Solution” had an x and y variable. (which are the genes of the chromosome). I started off by initializing an ArrayList to the size of initialPopulationSize, adding that many chromosomes with random x and y values between 0 and 100 (as stated in the instructions). At each generation, I crossed over and mutated different chromosomes (depending on the input of the user). I added those new genes to that ArrayList. After that, I sorted the ArrayList by fitness (highest fitness being at the front of the list, lowest being at the back), and I removed all of the lowest fitness chromosomes until I had the same size ArrayList as initialPopulationSize. I repeated this process until the first chromosome in the list was >= the threshold (as the chromosomes were sorted by fitness), or the number of generations exceeded maxGenerations, and then returned the first chromosome in the list (as that either was >= than the threshold, or it was the highest fitness chromosome after exceeding maxGenerations).
2. My fitness criterion was just plugging x and y into the formula: $6x - x^2 + 4y - y^2$, and whatever value that returned was the fitness.
3. After calculating the derivative myself and finding the highest value that this equation could be, I found that $x=3$ and $y=2$, and the equation would equal 13, so my threshold was 13.
4. After running thousands of tests to see which selection type produced the best result, I do not have a conclusive answer. Each selection type performed extremely well, but I did not find that one selection type produced better results than the other.
5. The way my implementation performed mutation was that it found two random integers between -5 and 5, and added those integers to the x and y values of that chromosome respectively. If, after being mutated, the x or y values were either less than 0 or greater than 100, that value was changed to 0 or 100 respectively.
6. The way my implementation performed crossover was it took the two chromosomes (let’s call them A and B), and made two new chromosomes out of them. One chromosome took the x value of A and the y value of B, and the other took the x value of B and the y value of A, and added those to the list of chromosomes.