Ministry of Education and Science of the Russian Federation
Peter the Great St. Petersburg State Polytechnic University
Institute of Computer Sciences and Technologies
Graduate School of Cyber-Physical Systems and Control

Practice Task – Ch 6

Genetic Algorithm
Discipline: Intellectual Computing
3 April 2017

Student Group: 13541/8

_______________________ Christopher W. Blake

Professor

_______________________ Kuchmin A.Y

St. Petersburg
2017

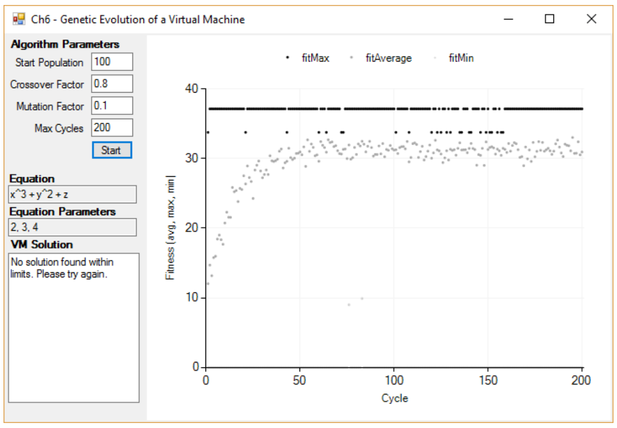
Contents

Introduction

Chapter 6 of "AI Application Programming" by M. Tim Jones is about the genetic algorithm, which is used for optimization. In this case the usage of basic stack operations (duplicate, swap, multiply, add, over), to mimic a mathematical equation. The equation to is described by a chromosome and various genes (the stack operations). Each gene represents one operation in the stack and can be replaced by other gene possibility (operation). By swapping the genes, different configurations of the stack are created.

The genetic algorithm occurs by introducing a fitness function and forms of chromosomal manipulation. The fitness function is used to judge how well the new object (and its genes) fulfills a desired task. Those with higher results have higher possibilities of reproduction and continue their genes. Two forms of chromosomal manipulation are utilized in this program: crossover and mutation. Through these manipulations over cycles of new generations, an optimized solution becomes present in the population.

A sample C# program has been created to show this methodology and genetic development. The program allows the user to specify the initial population, crossover probability, mutation possibility, and max number of generation cycles. After the simulation is run, the maximum and average fitness vs generation cycle are displayed.



Algorithm Parameters
Start Population: 100
Crossover Factor: 0.8
Mutation Factor: 0.1
Max Cycles: 200

Equation
x^3 + y^2 + z

Equation Parameters
2, 3, 4

VM Solution
No solution found within limits. Please try again.

Figure 1: C# Sample Program

Background

Chromosomes and Genes

Each chromosome with genes in the genetic algorithm is actually a stack of operations to perform on another stack of numbers. For example, the below chromosome includes 6 genes (the below stack includes 6 operations) to represent the equation a^2 + b^2 on a number stack of [a, b].

Chromosome = [duplicate, multiply, swap, duplicate, multiply, add] (a^2 + b^2)

Fitness Function

A fitness function is used to judge how well the solution fulfills the task. In the above example, the fitness would be the highest because it properly mimics the equation. However, adding additional fitness steps allows the better approximations to remain within the population. For example, the fitness value is increased according to the below rules for this program.

If no error:	+100 points
If numeric stack ends with 1 number:	+200 points
If numeric stack ends with +1 numbers:	-10 points per additional number
If numeric stack is correct value:	+500 points

Crossover

Crossover is the mixing of two chromosomes to create two new chromosomes. A crossover point is selected, and the genes are copied from each parent, creating two new children. See figure 2. In the program, the crossover factor controls the probability of chosen parents producing children via crossover. The actual crossover point is random.

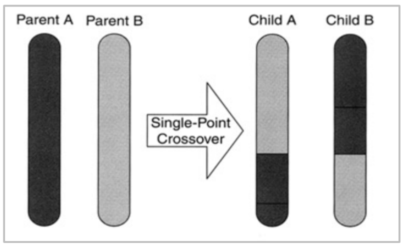


Figure 2: Genetic Crossover

Mutation

Mutation is the random changing of one or more genes within the chromosome. See figure 3. In the program the mutation factor controls the probability of a produced child of having a mutation. The gene selected for mutation is random.

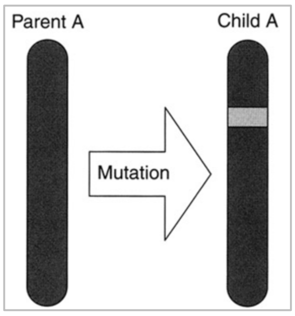


Figure 3: Genetic Mutation

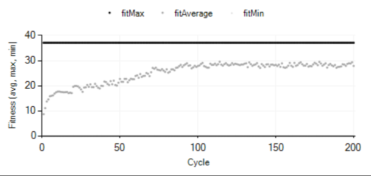
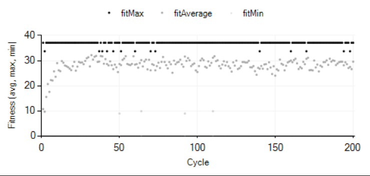
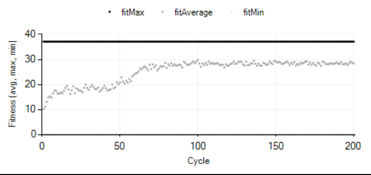
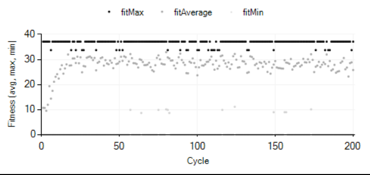
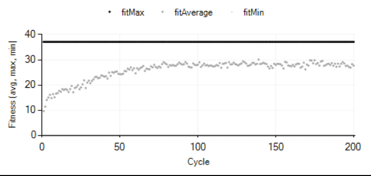
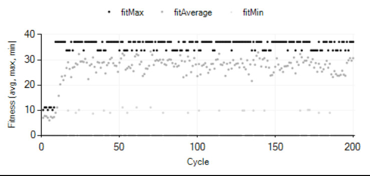
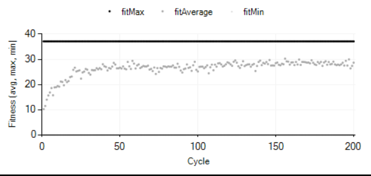
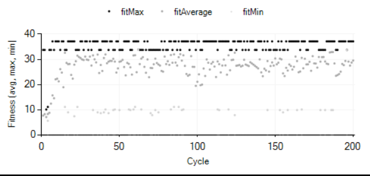
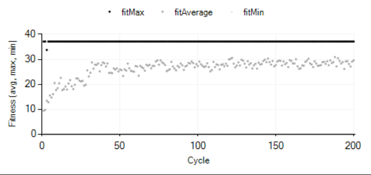
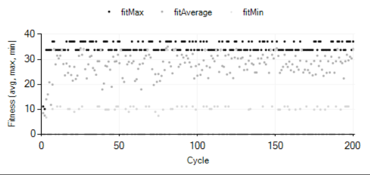
Results

The initial population is important on the calculation requirements and number of generations cycles until a solution is found. As such a few plots have been produced to show the effect of changing population.

The charts to the right show that the initial population significantly affects the averate fitness level. As less initial population is used the presence of the randomization function becomes apparent. With low populations, such as 10 and 20, there is little or no obvious fitness improvement. However, with a large population, the fitness clearly increases with each generation.

However, it should be noted, using large populations is calculation intensive. As such a middle value such as 100 is likely sufficient.

Table 1: Fitness vs Generation Cycle for different population sizes

500 	50 
400 	40 
300 	30 
200 	20 
100 	10 

Conclusion

A genetic evolution approach has been applied to generating stack operations that estimate a mathematical equation. Afterwards, the effect of initial population was investigated on fitness results vs generation. It is shown that a medium value such as 100 is sufficient to show improvement as well as avoid higher computation requirements.