Development of a Genetic Algorithm Model: GAP

Chan Wai Nelson

Department of Computing
Hong Kong Polytechnic University
csnchan@polyu.edu.hk

Abstract

This paper proposes a model for computer-assisted-learning in genetic algorithms, and a prototype GAP has been developed. GAP is an acronym that stands for Genetic Algorithms Platform. GAP allows student to input his/her own evaluation function, population size, number of generations, and etc. Standard cross-over methods are provided by GAP such as one-point, two-point and uniform cross-over. In addition, it can accept student defined cross-over method to produce the fittest offspring. GAP is mainly used for the purpose of demonstrating the feasibility and operations of the model. Its study and development are funded by the Educational Development Group of the Hong Kong Polytechnic University.

1 Introduction

In the context of searching for optimal solutions, real-world problems are so complex that robust methods for exploring complex solution spaces are desired. In the late 1960 at the University of Michigan, Holland [4] noted that there are two important elements in the learning process of a living species. One of them is learning by adaptation and the other is learning by evolutionary adaptation over a number of generations. To relate this learning process with Darwinian striving for survival, Holland proposed a learning strategy, called the searches reproductive plans. It is based on simulation of natural genetic inheritance and survival of the fittest. The so-called searches reproductive plans, later known as genetic algorithms (GA), are used as a problem-solving and function optimization technique. It uses a set of genetic operators to transform a population through successive generations for searching the fittest offspring (i.e. the optimal solution). Genetic algorithms are particular useful in situations where no easy algorithm is known, or the effort of finding optimal solutions is tremendous. The type of applications that GA can be applied falls into a wide spectrum ranging from scheduling, game playing, travelling salesman problem, strategy acquisition to multiple-fault diagnosis. Goldberg [3] has discussed vigorously about GA and their applications. More examples have been given by Forest [2] and Goldberg [1].

It is the purpose of this paper to propose a novel architecture of Genetic Algorithm Platform (GAP) for use of computer-aided learning in genetic algorithms. Its functional components will be described in detail, and some of the experience in developing such a prototype will also be discussed.

2 How GA Works

To begin with, we would like to investigate the relationship between a genetic algorithm and the application problem it attempts to solve. There are quite a few components worth to be considered, but we would concentrate only on two major components. They are the encoding mechanism and the evaluation function. Both these components are to be defined by a user and they vary from application to application.

There exists many encoding mechanisms. GAP employs the technique of bit manipulation. Bit manipulation mechanism is a representation of the application problem using chromosome bit strings. Thus, a chromosome is essentially a bit string consisting of 0s or 1s. For example, 10011101, can be considered as a chromosome bit string of length eight bits. Let us illustrate this encoding technique with a simple application problem of finding the cube root of $1331 = 11^3$. For this particular application, a chromosome is an integer whose cube is close to or nearly close to 1331. Such a chromosome is to be generated by the GA according to some user defined attribute. Therefore, the set of chromosomes under consideration can be taken as a certain range of integers whose cubes are close to 1331. The range, say from 1 to 20, is a possible choice. If we convert the upper bound 20 into its binary representation 10100, then the length of a chromosome bit string is 5, which is therefore a user defined attribute for the chromosomes under consideration.

The evaluation function, as defined by the user, is the mechanism that will link a particular application at hand to genetic algorithm. When a chromosome is input to this function, the corresponding numerical output is a measure of fitness of the chromosome. GA will start a new generation based on the fitness of chromosomes. This process is very similar to the natural selection. A possible choice of the evaluation function for our simple problem is:

$$f\ (\text{chromosome}) = 2000 - \text{abs}\ (\ \text{chromosome} \wedge 3 - 1331\ ) \qquad (1)$$

Assuming the GA has generated the following set of chromosomes, we now apply this particular evaluation function to calculate their fitness values.

Table 1. Evaluation of the fitness of chromosomes according to function f in (1).

Chromosome	Integer Equivalent of the Chromosome	Output f
00011	3	696
00001	1	670
10100	20	- 4669
11111	31	- 26460
11010	26	- 14245
01111	15	- 44
01010	10	1669
01100	12	1603
01011	11	2000

It is easy to see that the optimal fitness (i.e. 2000) is associated with the chromosome corresponding to the desired cube root 11 of 1331.

3 Steps of a Simple Genetic Algorithm

The above example illustrates the relationship between a chromosome and its evaluation function. To obtain the optimal solution, GA must be able to perform search on a set of potential solutions. This can be achieved by allowing information formation and exchange within the initial set of chromosomes. In addition, this set of chromosomes will undergo an 'evolution', so that only those 'good' chromosomes survive and become a new generation of chromosomes. The evolution will stop only when a certain pre-specified condition is satisfied. The description of our simple GA is shown in Figure 1.

Figure 1. Simple GA procedure.

```
procedure  Simple_GA
begin
  Initialize_Population;
  Evaluate_Population;
  repeat
    Select_Parent_Chromosome;
    Perform_Reproduction;
    Evaluate_Population;
  until  ( number of generation is exceed or optimal solution is found)
end
```

4 Architecture of a Simple GA

In order to implement the simple genetic algorithm, a number of modules need to be constructed. These include Population Module, Reproduction Module and the Evaluation Module.

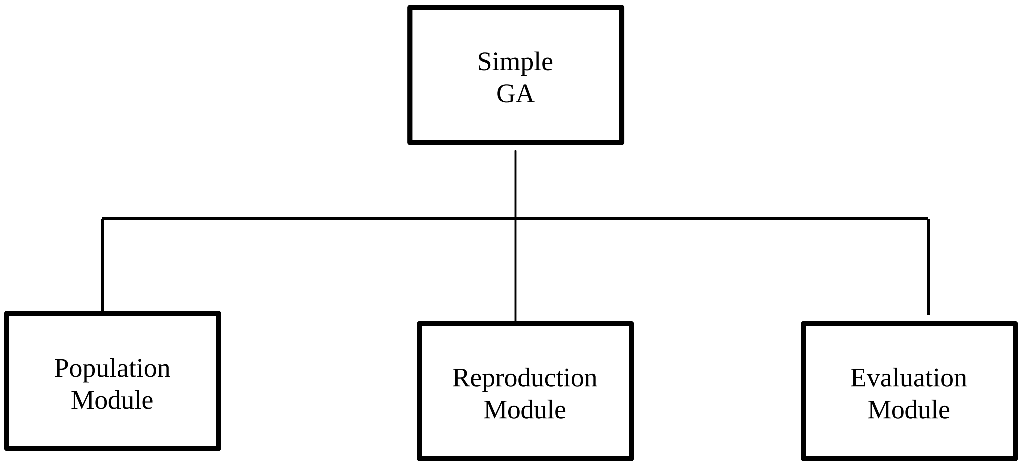
Population Module.
The Population Module is to reproduce and maintain a set of good chromosomes according to a certain rule of "survival of the fittest". Its major functions include initialization of the first generation, removal of some or all of the existing members (i.e. "bad" chromosomes) from the current population, selection of a group of parents for reproduction according to their fitness, and replacement of old generations with new generations.

Reproduction Module
Reproduction Module is to engage in reproduction process according to a reproductive plan dictated by crossover and mutation. Crossover recombines the bit value(s) in two parent chromosomes so as to reproduce two offspring. Mutation introduces random deviation into the population by altering one or more bits of an individual chromosome.

Evaluation Module
The goal of Evaluation Module is to rank a chromosome using a user-defined mathematical fitness function. Upon completion of evaluation, the value returned by the function is the fitness of the chromosome.

The structure of a simple GA model is depicted below:

Figure 2. A simple GA model.



Generally speaking, the Population Module will select two good parents and pass them on to the Reproduction Module. During each reproduction process, the Reproduction Module will apply crossover and mutation genetic operators to the selected parents. The two offspring so reproduced will become part of the population. Upon completion of the reproduction process, the Evaluation module will calculate the individual fitness of each chromosome and disqualify those unfitted chromosomes accordingly. Finally, a new generation of chromosomes is thus born and is ready for further reproduction process.

5 Construction of a Prototype: GAP

We have developed a prototype GAP (Genetic Algorithm Platform) for the purpose of demonstrating the feasibility and usability of the simple GA model. For the time being, we have implemented some basic functions within the Population Module, Reproduction Module and Evaluation Module. These functions are:

Population Module

This particular module enables one to specify the initial chromosome population, the type of fitness value (real or integer), user-defined evaluation function, the type of crossover method, and the rates of crossover and mutation. The parent selection method used in this module is based on the roulette wheel selection scheme.

Reproduction Module

We have successfully implement a number of crossover methods for this module. These are one-point crossover, two-point crossover and uniform crossover. For the mutation method, we use only bit mutation.

Evaluation Module

GAP has a built-in evaluation function that is capable of solving travelling salesman problem. Moreover, GAP can also accept user-defined evaluation function in the form of an algebraic function.

The following table illustrates different outputs generated by the GAP in finding the cube root of 1997 after 100 generations:

Table 2. Finding the root 1997.

Chromosome Range	Population Size	Crossover Rate	Mutation Rate	GAP Solution
1 - 100	40	0.6	0.01	12.8071
1 - 100	40	0.6	0.02	12.5742
1 - 100	40	0.6	0.03	12.5712
1 - 20	40	0.6	0.01	12.6045
1 - 20	40	0.6	0.02	12.6005
1 - 20	40	0.6	0.03	12.5280
9 - 20	40	0.6	0.02	12.5572
9 - 20	40	0.6	0.02	12.5923
9 - 15	40	0.6	0.01	12.5929
9 - 15	40	0.6	0.02	12.5929
9 - 15	40	0.6	0.005	12.5929

Note the cube root of 1997 is approximately equal to 12.592908...

6 Conclusions

The prototype GAP for simple GA has been successfully developed and tested. There are two interesting points that are worth to mention. The first point is related to the length of the chromosome bit string, which is to be specified by the user. The second point is related to the rate of convergence of GAP to the optimal solution.

For the first point, if a user carelessly specifies a range of chromosome which is too far away from the expected solution, then GAP may not converge when roulette wheel selection scheme is used. The reason is simple, for the evaluation function may return a very large negative fitness value, which will subsequently be used to form the running total of the fitness. As a result, a particular chromosome may be repeatedly chosen in a particular generation. To remedy this situation, GAP only select those parents whose fitness values are positive. In case it cannot find any chromosome of positive fitness value in a generation, it will terminate automatically without further processing.

For the second point, in order for GAP to converge to the optimal solution, a certain regulation mechanism should be built-in. The method is by restricting further the fittest chromosomes to a smaller selected range after GAP has completed the user defined generations. This is particularly handy when the length of chromosome is more than 20 bits. With this modification, GAP converges more quickly to 12.5929 in 50 generations for the above particular problem.

Acknowledgments

The research and development of the prototype GAP was the work of a project funded by the Educational Development Group of the Hong Kong Polytechnic University. The project started in December 1996. Members of the project team are Mr. Charles Tse, Mr. Julian Kwan with the author as the project leader. The author wishes to express his appreciation to all the team members. In particular, sincere thanks go to Dr. S. H. Hou, Head of the Applied Mathematics Department of the Hong Kong Polytechnic University, for his valuable comments and support of this project.

References

[1] D.E. Goldberg, *Genetic and Evolutionary Algorithms Come of Age*, Comm. ACM, Vol. 37, 1994, pp. 113-119.

[2] S. Forest, *Genetic Algorithms: Principles of Natural Selection Applied to Computation*, Science, Vol. 261, 1993, pp. 872-878.

[3] D.E. Goldberg, *Genetic Algorithm in Search, Optimization, and Machine Learning*, Addision-Wesley, Reading, Mass., 1989.

[4] John H, Holland, *Adaptation in Natural and Artificial Systems*, Ann Arbor: The University of Michigan Press, 1975.