International Journal of Engineering and Applied Technology (IJEAT)
Vol. 1 | November 2018
ISSN : 2620-9632

ANALYSIS OF THE MODEL USING GENERAL EVOLUTIONARY ALGORITHM

N. Hu
Department of Computer Science and Engineering, Normal University, China
*Email: nengfa@sina.com

Abstract

The study aims to solve the universality of evolutionary computation for solving different optimization problems, avoid the repeating design, and share codes, so as to improve the application efficiency of the algorithms. To achieve this, the authors put forward a general evolutionary model based on object-oriented programming language (C# language) in this paper. This model divides the problem domain, the fitness function, the algorithm and the parameter control into different classes, which are encapsulated into different objects so as to enhance the independence of each module. By utilizing the model, it is convenient to develop, reuse, extend and modify software while guaranteeing the universality of the algorithm at the same time.

Keywords: evolutionary computation, model, genetic algorithm, object-oriented

1 Introduction

Genetic algorithm which plays a key role in the evolutionary computation is a global optimization algorithm for simulating natural selection and natural genetic mechanism in the process of biological evolution. According to Darwin's theory of evolution and Mendel's theory of heredity, John Holland of Michigan University in America proposed genetic algorithm in 1975, which has been developed rapidly after being studied for more than forty years (Holland, 1975). This has been proved that it is a very effective method and has been widely applied in various fields such as neural networks, function optimization, image processing, system identification and expert systems due to its high efficiency and practicality (Fogel, Owens, & Walsh, 1966; Koza, 1994; Koza & Koza, 1992; Schwefel, 1995).

In fact, the essence of traditional genetic algorithms is to constantly evolve a population using population search technique based on the principle of "survival of the fittest" and finally to obtain the optimal solution. The specific realization method is as follows:

1. The feasible regions are determined according to specific problems and their solutions can be expressed by a coding scheme;
2. Each solution needs to be measured according to a measurement basis, which is generally signified by a nonnegative function, that is, fitness function;
3. The evolutionary parameters, namely, the population size (N), the crossover probability (P_c), the mutation probability (P_m) and the end conditions of evolution, are determined;
4. An initial population containing a number of individuals is generated, and each individual is a feasible solution to problems;
5. The fitness values of each individual in the population are calculated;
6. If the optimal solution is obtained, the loop stops;
7. According to the fitness values of each individual, the genetic operations including selection, hybridization and mutation are performed;
8. Going back to the fifth step.

Although genetic algorithms are very effective, they present a prominent shortcoming while being used to solve specific problems. That is, a problem can often be solved by employing different evolutionary algorithms, and an algorithm can also be used to solve different problems. In this case, if an application program needs to be designed for each problem and algorithm, it not only reduces the efficiency of work, but also brings difficulties in comparing various algorithms. Therefore, it is necessary to design a kind of evolutionary algorithm which has a high code reuse rate, preferable universality and expansibility so that effectively solve the above problems.

2 Design Model for Evolutionary Algorithm

2.1 Characteristics of the object-oriented programming C# language

C# language, as a comprehensive object-oriented programming language (Ge, Liu, Xiong, Tuzhilin, & Chen, 2011; Liu, Ge, Li, Chen, & Xiong, 2011; Tang, Wu, Sun, & Su, 2012; Zheng, Zheng, Xie, & Yang, 2012), inherits the advantages of C++ and Java languages and is characterized by simplicity, flexibility and powerful functions. In addition, it shows the most distinct characteristics such. as encapsulation, inheritance and polymorphism. C# language program which is composed of classes does not contain functions and variables independent from classes, and all attributes and methods are encapsulated in classes. Moreover, the examples of classes, that is, objects, are basic logic components of the programming.

The system shares the methods and attributes of classes based on the inheritance mechanism, while the communication among objects is realized by delivering information. The objects mainly consist of two parts, namely, attributes and methods, and there into, the former is used to describe the interior structures of objects, while the latter shows the behaviors of objects. The principal-agent mechanism provided by C# language can effectively realize the polymorphism of classes so as to reuse the code and improve the development efficiency of software.

2.2 Division of basic functions of evolutionary algorithm

Through analyzing the evolutionary algorithm, it is found that evolutionary algorithm can be divided into different modules, such as the problem domain, the fitness function, the evolutionary algorithm and the parameter control. Among them, the evolutionary algorithm is composed of some genetic operators including selection, hybridization and mutation. When designing the general evolutionary algorithm, the system can be divided into different function modules to guarantee the independence of all parts. The function figure (Figure 1) basically

reflects the composition of the overall system, and Figure 2 demonstrates the user interface for realizing Figure 1.

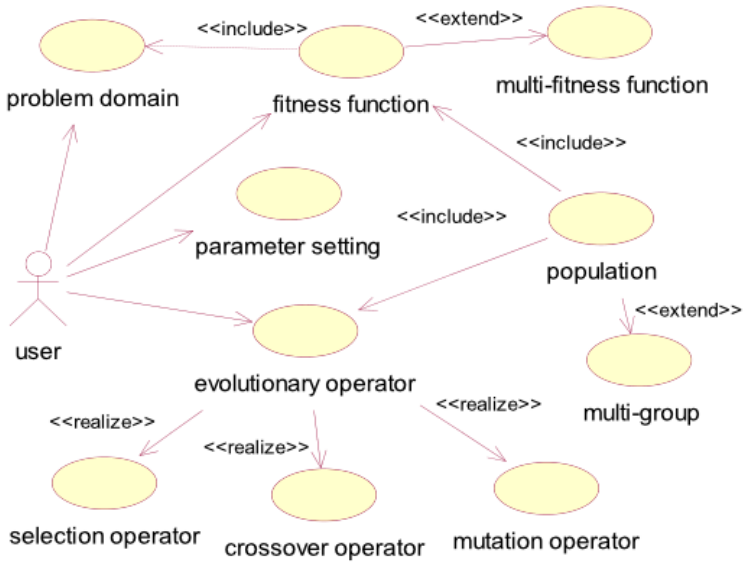

Figure 1. The functions of the system

In the function figure, the problem domain module mainly selects and processes mathematical models concerning different problems. While the fitness function module mainly calculates the fitness values of selected problems according to different coding schemes and rules to provide services for all individuals in the population, that is, calculating the fitness values of each individual. The evolutionary operator module mainly conducts genetic operations including selection, hybridization and mutation to evolve the population module so as to change the states of individuals in the population constantly.

All these operations performed by the evolutionary operator module are based on the services provided by the parameter setting module. The services refer to parameters such as the crossover probability, the mutation probability, the evolutionary generations and the end condition. While, the population module mainly computes the fitness values of all individuals in the population according to the services supplied by the fitness function module and marks the best and worst individuals as well as their current states. Meanwhile, external users can properly design, select, modify, set and maintain the system by changing the

contents of the different modules including the problem domain, the fitness function and the parameter setting to keep the system more stable and efficient.

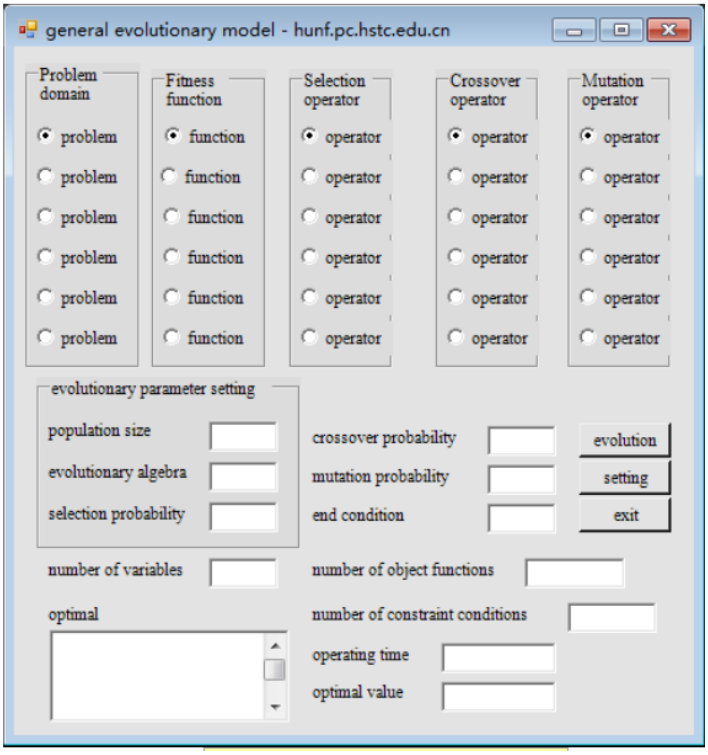

Figure 2. The home page of the user interface

2.3 The Static Model of the General Evolutionary Algorithm

The static model of the general evolutionary algorithm is shown as the class diagram in Figure 3. The general evolutionary algorithm consists of many classes, while the figure just presents the major ones. Among them, the problem class is used to describe the problem domain, in which the cause is the most important attribute which is an irregular two-dimensional (2D) array storing analytic expressions. An analytic expression is a mathematical model which is built after the abstraction of original problems and generally exists in the form of a multi-objective optimization problem with constraints. Analytic expressions can be divided into two types, namely, the objective ones and the constraint ones, both of which are stored into an irregular 2D array. The former is located in the elements on the first line, while the latter is stored in the elements on the second

line. Under circumstances that there is no constraint or merely one objective, the problem is supposed to be simplified as an optimization problem with a single objective or without constraints. Because the length of this irregular array like C# language is variable, it preferably solves problems in the case that the quantities of objective functions and constraint functions are unknown so that it is convenient to realize the software.

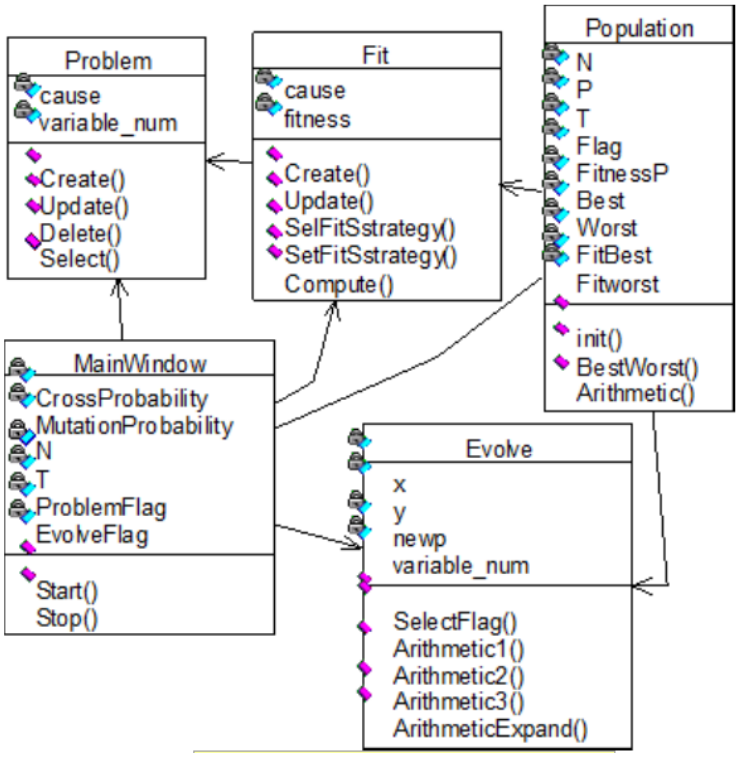

Figure 3. The classes of the evolutionary algorithm

Fit class is mainly used to calculate the fitness values of functions and can select different calculation schemas for class objects by employing self its strategy method. While when Compute method is adopted to calculate the fitness values of functions, it sends information to problem objects so as to call the corresponding problems to solve. Population class encapsulates main attributes and methods of an evolutionary population. There into, the attribute N marks the total number of individuals in the current population object; the attribute P is an irregular 2D array storing all individuals and contains N lines in total, while its column number depends on the number of variables in the solved problems; the

attribute Fitness P is a one-dimensional array belonging to double type and is mainly used to store the fitness values of all individuals in the population; while the attributes Best, Worst, Fit Best and Fit worst are used to record the subscripts and fitness values of the best and worst individuals, respectively; the attribute T represents the evolutionary generations. The sum of all attributes basically reflects the current state of the population. When the state satisfies certain conditions, the attribute Flag is set as 1.

Arithmetic method is mainly used to select individuals from the population and applies these selected individuals as parameters to perform genetic operations by using the method of Evolve class. When the algorithm ends, Best displays the evolutionary result. The method of Evolve class is mainly used for crossover and mutation operations to generate new individuals. Then, these new individuals and their fitness values are expected to return to the Population object, which then determines the individuals of the new population. Main Window class, as the entrance of whole system, is used to sends all information including the parameter setting, the selection of problems, the calculation rules of fitness values, the selection of genetic operators, and the start or stop of the algorithm.

2.4 Realization of the General Algorithm

C# language program is composed of the classes. On the basis of the designed class diagram, algorithms are designed and codes are programmed for each method in the classes. Due to the limitation of length of the paper, the authors only list a few main methods. In Fit class, many Compute methods are reloaded to calculate the fitness values of individuals. The mathematical model corresponding to problems in the algorithm is

$$\min\left\{ f_1(x), f_2(x), ..., f_k(x) \right\} (x \in D)$$

Where, K, n and q signify the numbers of objective functions, independent variables and inequality constraints, respectively. Meanwhile $D = \left\{ x \in S \subseteq R^n ; g_i(x) \le 0; i = 1, 2, ..., \mathrm{q} \right\}$. Especially, when *q=0*, $D = \left\{ x \in S \subseteq R^n \right\}$ which is a problem without constraints. The following algorithm is to solve multi-

objective optimization problems according to the weights. When a problem just has one objective, the value of *problem.cause [0].* length is set as 1. Here, the algorithm is converted to that for solving the optimization problem with a signal objective.

```
Public double compute () {
Problem problem=new Problem ( );
double fit;
for(int i=0; i< problem.cause [0].length; i++)
fit+=w*problem.cause[0][i];
for(int i=0; i < problem.cause[1].length; i ++)
   if (problem.cause[1][i]>0)
fit+=Math.Abs(problem.cause[1][i]);
return fit; }
```

Arithmetic1 method and other methods of Evolve class are sued for genetic operations such as hybridization and mutation. Each method can be reloaded by using a same function name, and can be extended or modified as well. The algorithm is:

```
Public void Arithmetic1 () {
Problem problem=new Problem( );
for(int i=0; i<variable_num; i++)
newp[i]=u*x[i]+(1-u)y[i]; }
```

3 EXPERIMENTAL ANALYSIS

In this paper, the authors select some examples to test, and the results show that the algorithm has favorable universality.

Example 1: $Min\ f(x)=\prod_{i=1}^{n}\sum_{j=1}^{5} j\cos\left[(j+1)x_i+j\right]$, s.t.-10 ≤ x_i ≤ 10, i=1, 2, ..., n.

Result: There are total 18 optimal solutions with the best value being –186.73.

Example 2: Solving equation $\left|\sin(30x)\right|\left(1-\frac{|x|}{2}\right)-0.973=0$. Result: There are 2 solutions, namely,0.0517 and -0.051799.

Example 3: Solving equation $\begin{cases} (x+99.71)^2 + y^2 - 10000 = 0, \\ \sin(5x) + \cos(5y) - 19932 = 0, \end{cases}$, x∈[-2,2], y∈[-2,2].

Result: One optimal solution is obtained, that is, $\begin{cases} x = 0.290899 \\ y = 0.001901 \end{cases}$. Meanwhile, there are two suboptimal solutions, namely, $\begin{cases} x = 0.283 \\ y = 1.256 \end{cases}$ and $\begin{cases} x = 0.283 \\ y = -1.256 \end{cases}$.

4 Conclusion

Based on the object-oriented characteristic of C# language, the modules including the problem domain, the fitness value of individuals, the population, the evolutionary operators and the parameters are further divided into different classes. The purpose is to strengthen the independence of all modules so that make it convenient to develop, reuse, extend and modify software. The experiment illustrates that the algorithm is effective in improving the efficiency of software and shows relatively high universality.

References

Fogel, L. J., Owens, A. J., & Walsh, M. J. (1966). Artificial intelligence through simulated evolution.

Ge, Y., Liu, Q., Xiong, H., Tuzhilin, A., & Chen, J. (2011). *Cost-aware travel tour recommendation.* Paper presented at the Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining.

Holland, J. H. (1975). Adaptation in natural and artificial systems, University of Michigan press. *Ann arbor, MI, 1*(97), 5.

Koza, J. R. (1994). *Genetic programming II: automatic discovery of reusable programs*: MIT press.

Koza, J. R., & Koza, J. R. (1992). *Genetic programming: on the programming of computers by means of natural selection* (Vol. 1): MIT press.

Liu, Q., Ge, Y., Li, Z., Chen, E., & Xiong, H. (2011). *Personalized travel package recommendation.* Paper presented at the 2011 IEEE 11th International Conference on Data Mining.

Schwefel, H. (1995). E volution and O ptimum S eeking. J ohn Wiley and Sons. *New York*.

Tang, J., Wu, S., Sun, J., & Su, H. (2012). *Cross-domain collaboration recommendation.* Paper presented at the Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining.

Zheng, V. W., Zheng, Y., Xie, X., & Yang, Q. (2012). Towards mobile intelligence: Learning from GPS history data for collaborative recommendation. *Artificial Intelligence, 184*, 17-37.