Test case Optimization using Genetic Algorithm

Reena* Pradeep Kumar Bhatia
GJU&ST, Hisar

Abstract: Testing plays significant role in the development of software. Testing is done with the help of test cases. The aim of test cases is to find the errors in the program. Genetic Algorithm is heuristic methods used for the optimization of test cases. GA is an iterative process. In each generation, it is the responsibility of GA to select fit individuals from the pool of individuals and discard the unfit individuals. The process of mutation and crossover is responsible for the selection of individuals for the next generation. In each generation, some individuals/chromosomes are combined using a crossover operator. After mutation, a new population is selected from the offspring and the original population. In this paper, we have applied genetic algorithm to optimize test cases on program i.e. HCF of two numbers. The best test case comes out to be (7, 2) with fitness value 140. The purpose of this paper is to provide optimized results by improving the efficiency of GA.

Keywords: Genetic algorithm, test case, optimization, COTS, Testing.

1. Introduction

Genetic Algorithm (GA) is a famous evolutionary search optimization methodology that imitates the natural processes of evolution to evolve answers to problems that have big search spaces. The genetic algorithm follows the biological evolution mechanism to find large and complex search spaces [1, 2]. John Hollad developed a genetic algorithm at the Michigan University in 1970's. Salvatore Mangano defined GA as "Genetic Algorithms are good at taking large, potentially huge search spaces and navigating them, that looks for optimal combinations of things, solutions you might not otherwise find in a lifetime" [3]. There are many Evolutionary Algorithms which are shown in figure 1.

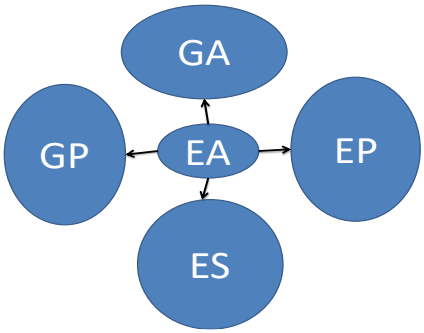

Figure 1. The EA Family

EA: Evolutionary Algorithms; GA: Genetic Algorithms; GP: Genetic Programming;

ES: Evolutionary Strategies; EP: Evolutionary Programming.

GA is a nature-inspired algorithm that converts the task of test case generation into an optimal solution [4].

There are many techniques of test case minimization have been suggested by the researchers. In this work, the focus will be on the Genetic Algorithm search technique to minimize the test cases. We will propose a more effective technique of fitness scaling in the Genetic Algorithm to make the test case minimization model [5, 6].

Depending on the biological evolution mechanism; the Genetic algorithm is a random and directed search algorithm that is aimed to find large and complex search spaces.

2. Different Search Optimization Techniques

There are many search optimization techniques (as shown in figure 2). The major techniques are an enumerative and random search [7, 8]. Every point of the search space is evaluated by enumerative techniques to get the most favorable result. An example of enumerative search techniques is dynamic programming. On the other hand, additional information concerning the search space used by guided random search techniques to guide the search process to reach an optimal solution in less time. The genetic algorithm is type of a guided random search technique [9].

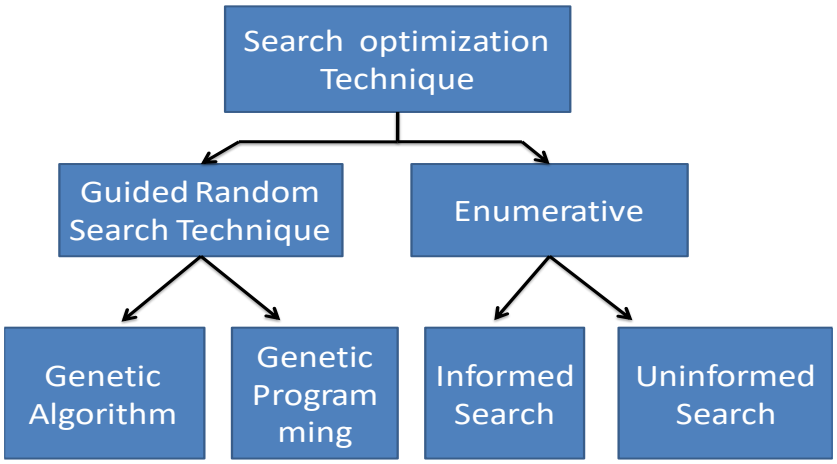

Figure 2. Classes of Search Techniques

3. Process and Algorithm of Genetic Algorithm

GA is a repetitive activity in which each repetition is known as "Generation". Each generation is responsible for selecting better and fit individuals from a large population of individuals (known as "chromosomes") and throwing away individuals who are less fit [10].

Actually, by maintaining a population of different individuals, GA carries out a multi-directional search and also knowledge interchange among these individuals. A simulated evolution is performed by individuals. By simulated evolution, we mean to say that relatively good individuals reproduce at each subsequent generation while relatively bad individuals die. An objective function is used to differentiate among individuals.

The selection of individuals is done by the process of mutation and crossover. In each generation, some chromosomes are combined using a crossover operator. Some of the individuals of this set are mutated and then the selection process is used to opt the new population from the children and the original population. Combination and selection are guided by the fitness function i.e. the chromosomes having more fitness value will have more chance to be selected and combined [11]. The pseudo-code of GA can be written as:

Pseudo-code of Genetic Algorithm

Initialize the population;

Assess the population (Using the given fitness function);

While Stopping Condition Not Gratified

{

Choose chromosomes for reproduction;

Carry out crossover and mutation;

Assess population;

}

3.1 Steps involved in Genetic Algorithm

As per the pseudo-code, the main components and their functionalities required to implement the genetic algorithm [12] are:-

Step 1. A Genetic Representation (Chromosomes) for Potential Solutions to the Problem

The chromosome is the word derived from natural genetics. These are the genetic representation of the individuals in a population which are also known as strings. Chromosomes are made of units called genes (or characters or alleles). Binary string or any other data structure acts as a gene in the chromosome. The primary population of chromosomes can be randomly generated with their length depending on the required precision.

Step 2. Initial Population for Potential Solutions

The initialization process is very simple and a population of chromosomes is created where each chromosome can be encoded either in binary form or in any other form depending on user requirement. Usually, first generation of chromosome is randomly generated.

Step 3. Evaluation Method

Evaluation method evaluates the fitness value of chromosomes. This fitness value of chromosomes describes their perfectness in meeting the particular objective. Fitness function is the function that needs to be optimized and we apply the genetic algorithm for its optimization. Fitness function can either be minimized or maximized according to our requirements.

Step 4. Genetic Operators

After the selection of best individuals using evaluation function; genetic operators are applied to them to form the next generation i.e. to form the children of the previous generation. These operators alter the composition of children to make them better in each successive generation. The operators used are:

a) Crossover

b) Mutation

Crossover: - The crossover operator creates two new chromosomes (child) by exchanging genes from two chromosomes (parents). The crossover probability (pc) is set by the user, According to which crossover occurs. A random number (r) is generated in the range [0, 1] and the crossover is applied to parents only if pc>r. It increases the exploitation of the search space. For example:

Chromosome 1 = (r1, r2, r3, r4, r5),

Chromosome 2 = (s1, s2, s3, s4, s5)

Crossing the chromosomes after the second gene would produce the offspring.

Progeny 1 = (r1, r2, s3, s4, s5),

Progeny 2 = (s1, s2, r3, r4, r5)

Mutation:-It alters one or more genes of a chromosome and creates one new chromosome [13]. The mutation is an adjustable parameter and it takes place according to a mutation probability (pm). A random number (r) is generated in the domain [0, 1] for each bit within the chromosome and mutation is applied on the bit only if (pm>r). It increases the exploration in the search space. For example:

Chromosome = 1001100111

The offspring after mutation becomes:

Offspring = 1011100101

Step 5. Values of various Parameters

Various parameter values need to be decided while applying the genetic algorithm to a problem. The parameters which are to be set includes: –

- Size of chromosomes.
- Population size.
- Crossover probability (usually set at 0.6).
- Mutation probability (usually set at 0.01).
- Number of generations.

The main advantage of the Genetic algorithm over conventional searching techniques is its robustness. GA does not break easily even if there is some noise or when the inputs get changed. So, GA may provide significant benefits in searching large search spaces over typical search optimization techniques [14, 15].

4. Result Discussion

The application of GA for various generations has been shown in tables. The values of these tables are obtained after the execution of the program i.e. HCF of two numbers using MATLAB programming platform version 2016a.

Table 1. Fitness Finding for Generation 1

S. No.	Test Cases	F	Pi	Ci	Random No.	N	M
1	11, 12	108	0.1698	0.1698	0.4387	3	3
2	6, 10	140	0.2201	0.3899	0.3816	2	2
3	14, 3	172	0.2704	0.6604	0.7655	4	4
4	3, 12	76	0.1195	0.7799	0.7952	5	5
5	15, 12	140	0.2201	1.0000	0.1869	2	2

Table 2. Crossover and Mutation for Generation 1

S. No.	N	M	Crossover	Mutation
1	3	3	(15, 2)	(14, 3)
2	2	2	(10, 6)	(6, 10)
3	4	4	(12, 3)	(3, 12)
4	5	5	(15, 12)	(15, 12)
5	2	2	(6, 10)	(6, 10)

Table 3. Fitness Finding for Generation 3

S. No.	Test Cases	F	Pi	Ci	Random No.	N	M
1	10, 6	172	0.2251	0.2251	0.8212	5	5
2	15, 12	140	0.1832	0.4084	0.0154	1	1
3	15, 12	140	0.1832	0.5916	0.0430	1	1
4	15, 12	140	0.1832	0.7749	0.1690	1	1
5	10, 6	172	0.2251	1.0000	0.6491	4	4

Table 4. Crossover and Mutation for Generation 3

S. No.	N	M	Crossover	Mutation
1	5	5	(10, 6)	(10, 6)
2	1	1	(14, 2)	(14, 2)
3	1	1	(14, 2)	(14, 2)
4	1	1	(14, 2)	(14, 2)
5	4	4	(15, 12)	(12, 15)

Table 5. Fitness Finding for Generation 50 with Simple GA

S. No.	Test Cases	F	Pi	Ci	Random No.	N	M
1	7, 2	140	0.2448	0.2448	0.4896	3	3
2	2, 7	108	0.1888	0.4336	0.2698	2	3
3	2, 3	108	0.1888	0.6224	0.9897	5	5
4	7, 2	140	0.2448	0.2448	0.4896	3	3
5	2, 3	108	0.1888	1.0000	0.8617	5	5

The best test case out of five randomly generated test cases is (7, 2) with fitness value = 140 but it can easily be seen that this is not the best test case because the criteria for the selection of best test case it must come several time in a single run with maximum fitness value. So, it is clear that simple GA sometimes leads to premature convergence of individuals which is the main drawback of simple GA.

5. Conclusion

In software testing, many times redundant test cases are used for a small piece of code. Testing is a tedious task and it requires more effort and time. Mostly numbers of defects are not uniformly distributed in COTS and defect that does not occur frequently requires more effort to remove. So, test case minimization techniques with proper test plans are required. To increase the efficiency of testing, a more effective way has been introduced, which saves a lot of time and resources i.e. GA. GA is the ever-emerging technique of great importance for researchers in testing. By application of GA on test cases, we found best test case is (7, 2) with fitness value 140. In future, we will work on limitation of GA by applying fitness scaling on GA.

References

[1] Ali, Shaukat, Yan Li, Tao Yue, and Man Zhang, "An empirical evaluation of mutation and crossover operators for multi-objective uncertainty-wise test minimization", In *2017 IEEE/ACM 10th International Workshop on Search-Based Software Testing (SBST)*, pp. 21-27. IEEE, 2017.

[2] Aljahdali, Sultan H., and Mohammed E., El-Telbany, "Genetic algorithms for optimizing ensemble of models in software reliability prediction", *International Journal on Artificial Intelligence and Machine Learning (AIML) ICGST* 8, no. 1 (2008): 5-13.

[3] Alsmadi, I., "Using genetic algorithms for test case generation and selection optimization", *23rd Canadian Conference Electrical and Computer Engineering (CCECE)*, pp. 1-4, IEEE, (2010).

[4] Alzabidi, Maha, Ajay Kumar, and A. D. Shaligram, "Automatic Software structural testing by using Evolutionary Algorithms for test data generations", *International Journal of Computer Science and Network Security* 9, no. 4 (2009): 390-395.

[5] Amjad, Muhammad Kamal, Shahid Ikramullah Butt, Rubeena Kousar, Riaz Ahmad, Mujtaba Hassan Agha, Zhang Faping, Naveed Anjum, and Umer Asgher, "Recent research trends in genetic algorithm based flexible job shop scheduling problems", *Mathematical Problems in Engineering* 2018 (2018).

[6] Amoui, Mehdi, Siavash Mirarab, Sepand Ansari, and Caro Lucas, "A genetic algorithm approach to design evolution using design pattern transformation", *International Journal of Information Technology and Intelligent Computing* 1, no. 2 (2006): 235-244.

[7] Bala, Anju, and Aman Kumar Sharma, "A comparative study of modified crossover operators", In *2015 Third International Conference on Image Information Processing (ICIIP)*, pp. 281-284. IEEE, 2015.

[8] Bhatia, S., A. Bawa, and V. K., Attri, "A review on genetic algorithm to deal with optimization of parameters of constructive cost model", In *international Journal of Advanced Research in Computer and Communication Engineering* 4, no. 4 (2015).

[9] Bhardwaj, Harshit and Pankaj Dashore, "A novel genetic programming approach to control bloat using crossover and mutation with intelligence technique", In *2015 International Conference on Computer, Communication and Control (IC4)*, pp. 1-6. IEEE, 2015.

[10] Böhmer, Kristof, and Stefanie Rinderle-Ma, "A genetic algorithm for automatic business process test case selection", In *OTM Confederated International Conferences On the Move to Meaningful Internet Systems*, pp. 166-184. Springer, Cham, 2015.

[11] Bright K., Vikash Y., "Automatic test case generation for performance enhancement of software through genetic algorithm and random testing", *Journal Engineering Science Research Science*, pp. 186-191, 2018.

[12] David, Lawrence - Davis, "Handbook of Genetic Algorithms", International Journal of Machine Learning and Computing, Vol. 7, No. 1, February 2017.

[13] Deb, K., and B. R. Agrawal, "Simulated Binary Crossover for Continuous Search Space (Technical Reports IITK/ME/SMD-94027)", *Convenor: Indian Institute of Technology, Department of Mechanical Engineering* (1994).

[14] Deb, Kalyanmoy, Ashish Anand, and Dhiraj Joshi, "A computationally efficient evolutionary algorithm for real-parameter optimization", *Evolutionary computation* 10, no. 4 (2002): 371-395.

[15] Ghiduk, Ahmed S., and Moheb R. Girgis, "Using genetic algorithms and dominance concepts for generating reduced test data". *Informatica* 34, no. 3 (2010).