

Adopting Social Group Optimization Algorithm Using Mutation Testing for Test Suite Generation: SGO-MT

Shweta Rani(✉) and Bharti Suri

USICT, GGS Indraprastha University, Delhi, India
shweta2610@gmail.com, bhartisuri@gmail.com

Abstract. Test case generation is popular among the researchers and doing this manually is an exhaustive and time taking process. Automation can cut its cost and create an effective test suite that is evaluated for its adequacy over a set of faults. These faults can be created by applying mutagenic rules that have been used appropriately for searching the improved test inputs in search-based approaches. Researchers have advised these approaches combining mutation testing are more effective at test generation. This paper proposes a novel test generation algorithm SGO-MT by adopting social group optimization algorithm (SGO) with the goal to reveal maximum faults in the software. SGO follows the concept of learning the traits of humans in a group. It works in two phases: acquiring phase (learning from society) and improving phase (learning from the teacher) that try to enhance the fitness of each individual. In learning from society, each individual test case is influenced by another while in the latter case, test data are evolved with respect to the fittest test case. SGO-MT stops functioning when it achieves its desired objective ie. detection of maximum possible artificial faults.

Keywords: Mutation testing · Social group optimization · Search based mutation testing · Test case generation

1 Introduction

Software quality is an important factor and must be verified before releasing the software to the customer [1,2]. It heavily relies on how and at what extent, testing is performed. To ensure error- free delivery, software is executed with sufficient test cases that should be effective in identifying the faults. For exhaustive testing, artificial faults can be injected using mutation testing that is a prominent area of research [3,5]. These faults are also known as mutants and used to assess the quality of the test set (adequacy score). Quality of a test case is evaluated on the basis of how many faults it can detect. The more faults it recognizes, the more it is said to be effective. However, manually creating the test suite is quite cumbersome and costly. Automation can generate an effective test set and

© Springer Nature Switzerland AG 2019
S. Misra et al. (Eds.): ICCSA 2019, LNCS 11622, pp. 520–528, 2019.
https://doi.org/10.1007/978-3-030-24305-0_39

also cuts the cost of generation. Prior studies [6–10] suggest that search based algorithms perform better for automatic generation of test cases.

Initially studied by Miller and Spooner [11], search based testing algorithms start searching the solution space using a random set of initial population. This population is then evolved over a number of generations till it gets converged. In testing, at convergence, test data must successfully satisfy the coverage criteria ie. branch coverage, statement coverage, mutation coverage, path coverage and so on. Fraser in his studies [12,13] reveal that mutation coverage criteria is superior and generates an exhaustive test suite. In the literature [7,8,10], different population-based algorithms have been adopted. Genetic algorithm, particle swarm optimization, ant colony optimization, bee colony optimization are some of the popular techniques in the area. Researchers [14–21] are still working to simulate the behavior of different species and suggesting new search based algorithms for solving complex problems. Some of the recently developed techniques are lion optimization [15], grey wolf optimization [19], ant lion optimization [20], moth flame optimization [14], spider monkey optimization [17], social group optimization [21].

Search based algorithms have been effectively combined with mutation testing to solve the test suite related problems i.e. test suite generation, optimization, prioritization and selection and can be traced in these surveys [4,22–26]. Most of these algorithms require several input parameters and for optimal performance, these parameters must be chosen carefully. Social group optimization (SGO) is one of the recently proposed optimization algorithms and takes a few parameters and is quite simple to understand. It is originally developed by Satapathy and Naik [21] and simulates human learning behavior. In this paper, we propose a novel test data generation algorithm ie, SGO-MT by combining SGO and mutation testing. However, we could not find any test generation algorithm based on SGO. Despite the ease in understanding, SGO is found to perform well over GA and PSO when applied for fault analysis in dissolved gas analysis system [26] and when used for solving data clustering problems [27].

2 Search Based Algorithms

We study different search algorithms (refer to Table 1). Some of these techniques are popular and some are recently proposed as discussed below.

2.1 Genetic Algorithm (GA)

The principle of GA is laid upon the Darwinian principle of survival of fittest. It was initially conceived by Holland [28]. GA is one of the popular algorithms among the researchers [22,23,29]. The basic process of GA starts with the random initialization of the population which is forwarded for fitness evaluation and is evolved by applying a choice of reproduction operators. Reproduction operators consist of three types of operators ie. selection, crossover and mutation. It selects the best fit individuals for crossover and mutation. Crossover is performed between the two

individuals that exchange their properties while mutation makes some changes in the individual to diverse the complete solution. It iteratively quests for a better solution and provides the globally optimal solution at convergence.

2.2 Particle Swarm Optimization (PSO)

It is a swarm-based search technique that mimics the behavior of animals and is inspired by bird flocking and herding [30]. A number of solutions refer to the swarm of particles that fly in searching for the best solution. During the search, they keep updating their location and velocity and find the best solution in their path.

2.3 Ant Colony Optimization (ACO)

Initially proposed by Dorigo [31], the foundation of ACO is based on the food finding behavior of ants. In spite of being blind, ants find the smallest route to the food source by the detection rate of pheromone. It consists of two processes ie. pheromone deposition and evaporation. Each ant deposits the pheromone while trailing a path and this evaporates with the time. In the end, each ant updates the trail along with pheromone that in turns provides the shortest route to the food source.

Table 1. Search Based Algorithms

S. No	Algorithm	Inspired by	Year
1.	GA	Survival of fittest, mimic gene behavior of mating	1992
2.	PSO	Bird flocking	1995
3.	ACO	Ants food locating nature	1996
4.	SMO	Fission and fusion behaviour of spider monkeys	2013
5.	MFO	Navigation method of moth	2015
6.	SGO	Influencing behavior of human in a group	2015
7.	LOA	Prey locating behavior of lion	2016
8.	WOA	Bubble net feeding method of whale	2016

2.4 Moth Flame Optimization (MFO)

In 2015, Mirjalili [14] proposed an optimization algorithm mimicking the navigation method of moths. Moths try to evolve in the night and travel in a straight line at a fixed angle with respect to the moon. Artificial light misguides the Moths and makes it trapped in the spiral motion of light. Moths are basically the agent and fitness is referred to as flame. Each moth in the population repeatedly updates its location and flames according to the best moth and finds the best solution at the end.

2.5 Lion Optimization Algorithm (LOA)

Inspired by the prey locating behavior of lions, Yazdani, and Jolai [15] developed a new lion optimization algorithm (LOA). Previously Rajakumar [32] only proposed the lion's operators for mating and generating new solutions. Lions are wild cat animal and considered to be highly co-operative as well as antagonistic [33]. They are divided into two groups ie. residents (live in a group called pride), nomads (move alone or in pairs). In LOA, the initial population is randomly generated and some of them selected as nomads and rest as residents. They move and hunt to find the prey and attack it fastly. Weak lions die with time and only strong lions (solutions with food) remain alive at the end.

2.6 Spider Monkey Optimization (SMO)

Bansal et al. [17] proposed SMO that follows the idea of fission and fusion to find the food source. In this, a group of spider monkeys with a female leader starts searching the food and if she could not find the sufficient food then she divides the group into small groups that again hunt for food throughout the day. SMO impose limits on how many times a group can be divided. A member of the group can interact with other local member or a member in a different group for finding the food. They interact with each other using sounds, postures and positions.

2.7 Social Group Optimization (SGO)

The foundation of SGO is laid upon the influential behavior of humans [21]. They are great in mimicking the behavior. Each person has some knowledge and improves it with the influence of others knowledge and their behavior. SGO is a population-based algorithm where a group of person is considered as a population. SGO works in two phases ie. improving phase and acquiring phase. In the first phase, the best fit solution tries to improve each individual in the group. On the other hand, knowledge is enhanced by communicating with other members of that group in acquiring phase. In each phase, if the new solution is better then it is kept otherwise rejected. The whole process of acquiring the knowledge is repeated until convergence.

2.8 Whale Optimization Algorithm (WOA)

Inspired by the hunting nature of whale, Mirjalili and Lewis [16] proposed whale optimization algorithm. Whales are the biggest mammals and can be maximum 30 m long and 180t weight. They are social in nature and can live alone or in groups. They search their food by bubble net feeding method [34] in which a group of whales locate the food and catch it tightly by creating a net of bubbles. Whales have similar cells like human and these cells make them intelligent and emotional [35]. These cells help whales to locate the prey and consider it as the best solution. After finding the best solution by exploration and exploitation [16], all others try to evolve towards the best one.

3 Proposed Test Data Generation Approach: SGO-MT

In this section, we propose a novel search based algorithm by applying SGO and mutation testing for test data generation and named it SGO-MT. The flow chart in Fig. 1 illustrates the functioning of SGO-MT. The process starts with random initialization of the population of persons. Here, the population of persons is considered as a set of test cases. Each person has its knowledge to solve any problem. In our case, each test case is known for its fitness along with its value. Fitness is the adequacy score of a test case that is devised by executing it against the mutants. The more it kills the mutants, the higher its fitness is. Then, it works on evolving the test suite by improving it with respect to the best test case and other test cases. Evolution takes place in two phases ie. improving phase and acquiring phase. After repeated evolution, it provides the optimal solution at convergence.

Step 1: Initialize Population
Initially, decide the size of the initial population (N), number of generations (g), number of input variables (I) and limits of each variable (U_L, L_L). For the size of the population (N), randomly generate an initial set of test cases. Each test case consists of values equal to D. The population (P) is represented in the following form.

$$Population(P) = \begin{pmatrix} TC_{1,1} & TC_{1,2} & TC_{1,3} & ... & TC_{1,\mathrm{I}} \\ . & . & . & & . \\ . & . & . & & . \\ . & . & . & & . \\ TC_{\mathrm{N},1} & TC_{\mathrm{N},2} & TC_{\mathrm{N},3} & ... & TC_{\mathrm{N,I}} \end{pmatrix}$$

Step 2: Fitness Evaluation
To test the usefulness of the test case, each test case in P is intended for fitness evaluation. Here, fitness is the adequacy score of the test case which is measured by executing it against the mutants [3]. It is also known as mutation score (refer Eq. 1).

$$Fitness\ Score\ (FS) = (Killed\ Mutants/Total\ number\ of\ Mutants)\ * 100; \tag{1}$$

Step 3: Improving Phase:

In this phase, each test case is improved with the influence of the best-fit test case. It applies the concept of learning from a teacher who is considered to be the best in the group. Test data is enhanced using Eq. 2. We accept the new test case if found better. Let *gbest* be the best solution with the highest fitness.

$$P^{\mathrm{i}}_{\mathrm{new}}(j) = c\ * P^{\mathrm{i}}_{\mathrm{old}}(j)\ + r\ * (gbest\ -\ P^{\mathrm{i}}_{\mathrm{old}}(j)); \tag{2}$$

Here, P(i, j) is the test case value, i ε (1, N) and j ε (1, I). c is self-introspection parameter lies in $(0, 1)$. r is a random number lies in $(0, 1)$.

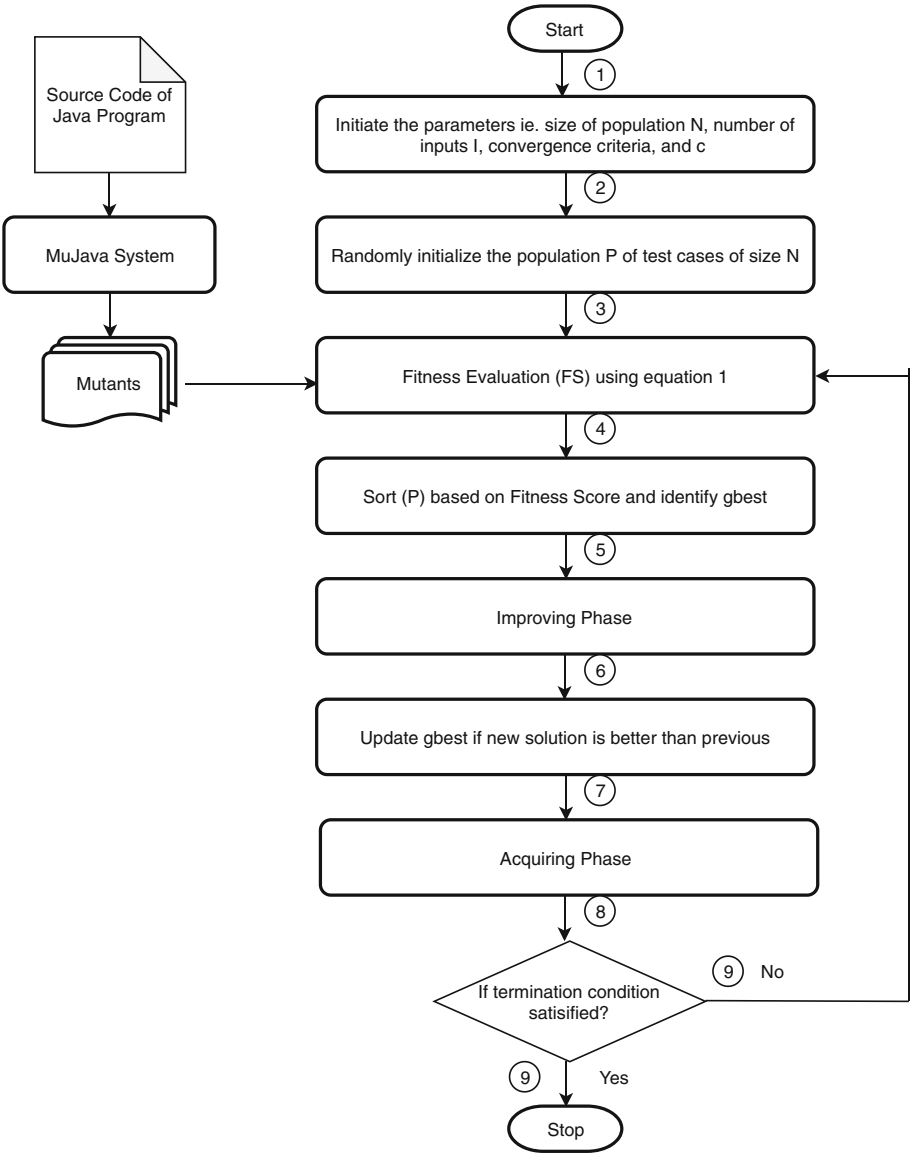

Fig. 1. Flow Chart of the proposed SGO-MT approach for Test Data Generation

Step 4: Acquiring Phase

In a group, each member tries to improve by learning from others. Here, each test case intends to improve its fitness with respect to other test cases in the population. Test data is enhanced using Eq. 3. The new test case is kept in the system if it contributes to the effectiveness of the test suite.

$$P^{\mathrm{i}}_{\mathrm{new}}(j) = \begin{cases} equation\ 4, & \text{if } FS(P_{\mathrm{i}}) < FS(P_{\mathrm{m}}) \\ equation\ 5, & \text{otherwise} \end{cases} \quad (3)$$

$$P^{\mathrm{i}}_{\mathrm{new}}(j) = P^{\mathrm{i}}_{\mathrm{old}}(j) + r1 * (P^{\mathrm{i}}_{\mathrm{old}}(j) - P^{\mathrm{m}}(j)) + r2 * (gbest - P^{\mathrm{i}}_{\mathrm{old}}(j)); \quad (4)$$

$$P^{\mathrm{i}}_{\mathrm{new}}(j) = P^{\mathrm{i}}_{\mathrm{old}}(j) + r1 * (P^{\mathrm{r}}_{\mathrm{old}}(j) - P^{\mathrm{i}}(j)) + r2 * (gbest - P^{\mathrm{i}}_{\mathrm{old}}(j)); \quad (5)$$

Here, in Eq. 4, P^{m} is a randomly selected test case taken for learning. r1 and r2 are random numbers within (0,1).

Step 5: Termination
In general, search-based algorithms stops functioning at convergence. SGO-MT converges when all the non-equivalent mutants are detected by the test suite i.e. achieving 100% mutation score. However, some mutants are hard to be killed and only detectable by value specific test cases. To avoid stuck in an infinite loop for finding such mutants, another criterion is also defined i.e. iteration limit. The approach stops functioning when both of the criteria satisfied.

3.1 Description of Mutants

Instead of real faults, artificial faults (mutants) are created using MuJava [36]. MuJava is an automated framework for mutant creation. Faults are injected by applying mutagenic rules proposed by Ma et al. [37]. Each operator is investigated for its equivalent behavior. Equivalent mutants are identical to the original program and could never be killed by any test suite [38]. We choose a small set of mutation operators that produce a fewer number of equivalent mutants.

4 Conclusion

In this paper, we propose a novel search algorithm for test generation ie. SGO-MT. It combines social group optimization algorithm (SGO) and mutation testing with the objective to find fault revealing improved test cases. The fitness of each test case is measured over a non-redundant set of mutants. These mutants are powerful and create a few redundant and equivalent mutants. SGO is inspired by the learning behavior of humans. They learn from society and from the teacher. Learning is associated with the evolution of test cases in SGO-MT and each test case keeps learning till stagnation. In the future, we plan to investigate and compare SGO-MT with other search-based testing algorithms for test data generation.

Acknowledgment. The authors would like to acknowledge Ministry of Electronics and Information Technology, Govt. of India for supporting this research under Visvesvaraya Ph.D. Scheme for Electronics and IT.

References

1. Myers, G.J.: The Art of Software Testing. Wiley, New York (1989)
2. Agarwal, K.K., Singh, Y.: Software Engineering. New Age International Publishers (2007)
3. Jia, Y., Harman, M.: An analysis and survey of the development of mutation testing. IEEE Trans. Softw. Eng. **37**(5), 649–678 (2011)
4. Zhu, Q., Panichella, A., Zaidman, A.: A systematic literature review on how mutation testing supports quality assurance processes. Softw. Test. Verif. Reliab. **28**(6), 1675 (2018)
5. Andrews, J.H., Briand, L.C., Labiche, Y.: Is mutation an appropriate tool for testing experiments? In: Proceedings of the 27th International Conference on Software Engineering, ICSE 2005, pp. 402–411. ACM (2005)
6. Dave, M., Agrawal, R.: Search based techniques and mutation analysis in automatic test case generation: a survey. In: 2015 IEEE International Advance Computing Conference (IACC), pp. 795–799 (2015)
7. McMinn, P.: Search-based software test data generation: a survey: research articles. Softw. Test. Verif. Reliab. **14**(2), 105–156 (2004)
8. McMinn, P.: Search-based software testing: past, present and future. In: Proceedings of the 2011 IEEE Fourth International Conference on Software Testing, Verification and Validation Workshops, ICSTW 2011, pp. 153–163. IEEE Computer Society (2011)
9. Sahin, O., Akay, B.: Comparisons of metaheuristic algorithms and fitness functions on software test data generation. Appl. Soft Comput. **49**, 1202–1214 (2016)
10. Ali, S., Briand, L.C., Hemmati, H., Panesar-Walawege, R.K.: A systematic review of the application and empirical investigation of search-based test case generation. IEEE Trans. Softw. Eng. **36**(6), 742–762 (2010)
11. Miller, W., Spooner, D.L.: Automatic generation of floating-point test data. IEEE Trans. Softw. Eng. **SE–2**(3), 223–226 (1976)
12. Fraser, G., Zeller, A.: Mutation-driven generation of unit tests and oracles. IEEE Trans. Softw. Eng. **38**(2), 278–292 (2012)
13. Fraser, G., Arcuri, A.: Achieving scalable mutation-based generation of whole test suites. Empir. Softw. Eng. **20**(3), 783–812 (2015)
14. Mirjalili, S.: Moth-flame optimization algorithm: a novel nature-inspired heuristic paradigm. Knowl.-Based Syst. **89**, 228–249 (2015)
15. Yazdani, M., Jolai, F.: Lion Optimization Algorithm (LOA): a nature-inspired metaheuristic algorithm. J. Comput. Des. Eng. **3**, 24–36 (2016)
16. Mirjalili, S., Lewis, A.: The Whale Optimization Algorithm. Adv. Eng. Softw. **95**, 51–67 (2016)
17. Bansal, J.C., Sharma, H., Jadon, S.S., Clerc, M.: Spider Monkey Optimization algorithm for numerical optimization. Memetic Comput. **6**(1), 31–47 (2013)
18. Shah-Hosseini, H.: The intelligent water drops algorithm: a nature-inspired swarm-based optimization algorithm. Int. J. Bio-Inspired Comput. **1**(1), 71–79 (2009)
19. Mirjalili, S., Mirjalili, S.M., Lewis, A.: Grey Wolf Optimizer. Adv. Eng. Softw. **69**, 46–61 (2014)
20. Mirjalili, S.: The Ant Lion Optimizer. Adv. Eng. Softw. **83**, 80–98 (2015)
21. Satapathy, S., Naik, A.: Social group optimization (SGO): a new population evolutionary optimization technique. Complex Intell. Syst. **2**, 173–203 (2016)
22. Silva, R.A., de Souza, S.D.R.S., de Souza, P.S.L.: A systematic review on search based mutation testing. Inf. Softw. Technol. **81**, 19–35 (2017)

23. Jatana, N., Suri, B., Rani, S.: Systematic literature review on search based mutation testing. e-Inform. Softw. Eng. J. **11**(1), 59–76 (2017)
24. Rodrigues, D.S., Delamaro, M.E., Correa, C.G., Nunes, F.L.S.: Using genetic algorithms in test data generation: a critical systematic mapping. ACM Comput. Surv. **51**(2), (41)1–(41)23 (2018)
25. Souza, F.C., Papadakis, M., Durelli, V.H.S., Delamaro, M.E.: Test data generation techniques for mutation testing: a systematic mapping. In: Proceedings of 11th Workshop on Experimental Software Engineering Latin Americal Wrokshop (ESELAW) (2014)
26. Fang, J., Zhang, H., Liu, J., Zhao, J., Zhang, Y., Wang, K.: A transformer fault diagnosis model using an optimal hybrid dissolved gas analysis features subset with improved social group optimization-support vector machine classifier. Energies MDPI Open Access J. **11**(8), 1–18 (2018)
27. Naik, A., Satapathy, S.C., Ashour, A.S., Dey, N.: Social group optimization for global optimization of multimodal functions and data clustering problems. Neural Comput. Appl. **30**(1), 271–287 (2016)
28. Holland, J.H.: Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology. Control and Artificial Intelligence. MIT Press, Cambridge (1992)
29. Yang, X.S.: Nature Inspired Metaheuristic Algorithms, vol. 504. Luniver Press (2010)
30. Kennedy, J., Eberhard, R.: Particle swarm optimization. In: IEEE International Conference on Neural Networks, vol. 4, pp. 1942–1948 (1995)
31. Dorigo, M., Maniezzo, V., Colorni, A.: The ant system: optimization by a colony of cooperating agents. IEEE Trans. Syst. Man Cybern. **26**(1), 29–41 (1996)
32. Rajakumar, B.: The Lion's Algorithm: a new nature-inspired search algorithm. Procedia Technol. **6**, 126–135 (2012)
33. McComb, K., Pusey, A., Packer, C., Grinnell, J.: Female lions can identify potentially infanticidal males from their roars. Biol. Sci. R. Soc. **252**(1333), 59–64 (1993)
34. Watkins, W.A., Schevill, W.E.: Aerial observation of feeding behavior in four baleen whales: eubalaena glacialis, balaenoptera borealis, megaptera novaeangliae, and balaenoptera physalus. J. Mammal. **60**(1), 155–163 (1979)
35. Hof, P.R., Van Der Gucht, E.: Structure of the cerebral cortex of the humpback whale, Megaptera novaeangliae (Cetacea, Mysticeti, Balaenopteridae). Adv. Integr. Anat. Evol. Biol. Anat. Rec. **290**, 1–31 (2007)
36. Ma, Y.-S., Offutt, J., Kwon, Y.R.: MuJava: an automated class mutation system. Softw. Test. Verif. Reliab. **15**(2), 97–133 (2005)
37. Ma, Y.-S., Offutt, J.: Description of method-level mutation operators for Java. Technical report, Electronics and Telecommunications Research Institute, Korea (2005)
38. Grun, B.J.M., Schuler, D., Zeller, A.: The impact of equivalent mutants. In: 2009 International Conference on Software Testing, Verification, and Validation Workshops, pp. 192–199. IEEE (2009)