An adaptive flower pollination algorithm for minimizing software testing redundancy

Muhammad Nomani Kabir[1*], Jahan Ali[1], AbdulRahman A. Alsewari[1], Kamal Z. Zamli[1]
[1]Faculty of Computer Systems & Software Engineering, University Malaysia Pahang, 26300 Gambang, Pahang, Malaysia
*Corresponding author: nomanikabir@ump.edu.my

Abstract—**Optimization is the selection of a best set of parameters from available alternative sets. Global optimization is the task of finding the absolutely best set of parameters. In this paper, we present an adaptive flower pollination algorithm for solving an optimization problem, i.e., minimization of software testing redundancy. In software testing, test engineers often generate a set of test cases to validate against the user requirements to avoid deficiency of the software. A large number of lines of codes cause potential redundancies in software testing. In order to tackle the issue of redundancy, global optimization algorithms are used to systematically minimize the test suite for software testing. We tested the adaptive flower pollination algorithm on a number of experiments in software tests. The results were compared with existing results of some existing algorithms to demonstrate the strength of our algorithm. Comparison shows that our algorithm performs slightly better than the existing algorithms and thus, the proposed algorithm can potentially be used by researchers and test engineers to obtain optimal test suite requiring the minimum time for software testing.**

Keywords—Global Optimization; Stochastic method; Software Test Suite; Test-cases Redundancy Reduction.

I. INTRODUCTION

Optimization is the process of finding the minimum value of a cost function. Global optimization is the task of searching the absolute minimum value. The tasks of minimization and maximization are trivially related to one another. In practice, optimization can mean to accomplish a task in the most efficient way or the highest quality or to produce maximum yields with given limited resources. For example, a manager may need to decide an appropriate investment portfolio according to the specific investment objective; or an automobile industry wants to build the most fuel-efficient car [1].

In this work, our scope is software testing redundancy minimization. In the present era, software and its diffusion into many applications are growing and thus complexity is increasing, which in turn, causes overlapping in software test cases that make unwarranted redundancies. In software test-cases redundancy, a test to satisfy a specific requirement is covered by multiple tests [2-4]. Test-suite size will increase with higher redundancy which substantially elevates the overall testing time and cost. Thus, minimization of test suite redundancy is important in order to obtain a minimum number of test cases that suffice for satisfying the test requirements.

Global optimization methods are solved using the several algorithms based on stochastic approach such as Particle Swarm Optimization (PSO), Simulated Annealing (SA) and Genetic Algorithm (GA). As Software-test redundancy problem contains many local minima, no single strategy e.g., PSO, SA or GA can do well in all scenarios. Each method has advantages and disadvantages. PSO does not often perform well to find a global optimal solution when the optimizing variables in objective function are of a large dimension [5, 6]. SA is preferable for problems in which the global optimum with the best solution is less important than an local optimum with an acceptable solution in a certain time [7]. For higher dimensions, complexity of GA exponentially grows [8]. To develop an algorithm that has less complexity, but provides higher convergence rate is always challenging. Although there exist stochastic based global optimization algorithms e.g., Particle Swarm Optimization, Ant Colony Optimization and Genetic Algorithm which use the random sequence permutation to achieve the global optimum. Each algorithm has its own advantages and disadvantages over the others in terms of convergence and complexity.

In this work, we scrutinized the convergence and complexity issues of an optimization technique - flower pollination algorithm and reformulated a new algorithm to obtain a better trade-off between convergence and complexity. Our proposed technique is a modified flower pollination algorithm (MFPA) that is developed by devising a new strategy through updating the parameters (step lengths) at each iteration. The step lengths are set to increase if the convergence is satisfactory. On the other hand, they are reduced if the convergence is weak. This new strategy is aimed at accelerating the convergence. Performance and efficiency of the proposed algorithm were tested on minimization problems of software testing redundancy by comparing the results with the results of existing algorithms to demonstrate the strength of the proposed algorithm. Thus, implemented code of the algorithm will be useful to generate test cases with minimum redundancy of system requirements. Researchers and test engineers will have a tool to obtain satisfactory test cases compared to the existing tools and the generated optimal test case will require the minimum time for software testing.

The rest of the paper is organized as follows. Section II presents the related works on different stochastic methods for software test-cases minimization. The next section describes the methodology of building the proposed algorithm MFPA.

The test results are provided in section IV and finally, conclusion is made in section V.

II. RELATED WORKS

In software testing, test engineers often generate a set of test cases to validate against the user requirements to avoid deficiency of the software. A large number of lines of codes cause potential redundancies in software testing. In order to tackle the issue of redundancy, global optimization algorithms are used to systematically minimize the test suite for software testing. Since software test case redundancy minimization is an optimization problem with many minima, Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Simulated Annealing (SA) can be used to find the best solution [2-4].

In Genetic algorithm (GA) is an iterative search technique that mimics the process of natural selection used to find the solution of global optimization problems. Genetic algorithms belong to the larger class of evolutionary algorithms (EA) that solve optimization problems using techniques inspired by natural evolution, such as inheritance, mutation, selection, and crossover. Repeated fitness function evaluation for complex problems is often a bottleneck with GA. With higher number of elements which are exposed to mutation, there is often an exponential increase in search space size [8]. GA is used by Fraser and Arcuri [9] for their work on scalable mutation-based generation of whole software test suites. Moreover, Matthew [10] used metaheuristic optimization approach for mutation-based test data generation.

Particle Swarm Optimization is a population-based stochastic algorithm which follows the social behavior of animals e.g., bird flocking and fish schooling. PSO has the properties of simple computation with rapid convergence rate. The algorithm requires to adjust a few parameters which is a main advantage. The algorithm tries to attain the best value from the interactions among particles. However, if the search space is high, the convergence rate slows down near the global optimum. The algorithm performs poor when the optimization problem consists of a large and complex data set. As mentioned before, the algorithm fails to attain a global optimum if the objective function contains a large number of optimizing variables [6]. Ahmed et al. [11] introduced a modified version of PSO algorithm called Cuckoo search algorithm for configuration-aware software test. Jin et al. [12] combined artificial neural network and quantum particle swarm optimization. Gosciniak [5] proposed a new approach to particle swarm optimization algorithm, Jensi [13] investigated a modified version of PSO using a levy flight for global optimization. Ouyang [14] presented a hybrid harmony search technique with PSO with global dimension selection.

Simulated annealing (SA) is a stochastic algorithm that imitates the forging process of metal when the metal is rapidly heated followed by gradual cooling process. SA can often find a good solution even in the presence of noisy data; However, the global solution may not be found by this technique. SA may be preferable to alternatives e.g., brute-force search or gradient descent, for finding the solutions of the problems in a certain time where the precise global optimum is less important than an acceptable local optimum [7].

Combinatorial techniques are alternatives to solve the software test redundancy problems. Some of the combinatorial techniques [15, 16] are tried to minimize software testing redundancy.

As the problem of software test-cases redundancy minimization may contain many local minima, no single strategy can work well for all the problems. In software test-cases, a test suite can be constructed by a sequence of permutation of test cases where the test cases can be formulated in any order. Note that one of the (concatenated) permutations can minimally satisfy all the requirements using an exhaustive permutation. Nonetheless, for a large problem, exhaustive permutation may be expensive and difficult to manage. Hence, this work proposes a new global optimization algorithm using stochastic approaches. Specifically, we develop an adaptive flower pollination algorithm for solving the problem of software test-cases redundancy minimization.

III. METHODOLOGY

The proposed algorithm modified flower pollination algorithm (MFPA) is a metaheuristic algorithm which can be used for solving the problem of software test-cases redundancy minimization. Flower pollination algorithm was originally formulated by Yang [17] based on the flower-pollination process. This algorithm was developed considering four rules:

Rule 1: Global pollination is represented by biotic and cross-pollination, where a pollinator maintains a Levy flight.

Rule 2: Local pollination is designated by abiotic and self-pollination.

Rule 3: A reproduction probability that is proportional to the similarity between two flowers is used to measure the flower constancy.

Rule 4: Switching between local and global pollinations is performed with a probability $p \in [0,1]$ value. Local pollination is slightly dominant over global pollination because of the physical closeness and other factors e.g., wind.

Above four rules can be formulated as equations. First the local pollination: rule 2 is represented by:

$$x_{t+1}^{i} = x_t^{i} + c\,(x_t^{j} - x_t^{k}) \quad (1)$$

where x_t^i denotes the current pollen (i.e., the current solution at iteration t; x_t^j and x_t^k are the solution vectors which are the selected pollens from two different flowers of the same plant-species with $c \in [0,1]$ i.e., the value of c obtained from a uniform distribution in [0, 1]. In our case of software test-cases redundancy minimization, x_t^j and x_t^k are the test cases that selected randomly in the list.

Rule 1 can be translated as

$$x_{t+1}^{i} = x_t^{i} + aL(b)(g^{*} - x_t^{i}) \quad (2)$$

where g^* denotes the best solution at iteration t, $a>0$ represents the step size and L(b) implies Levy flight. For our problem, g* is best test case in the list. Levy flight imitates the characteristics of long move of the pollinators. Thus, L is deduced from a Levy flight as follows:

$$L = \frac{b\Gamma(b)\sin(\pi b/2)}{\pi}\frac{1}{s^{1+b}}, \quad (s >> s0 > 0) \qquad (3)$$

where b is the Levy exponent and Γ(b) is the gamma function. The formula (3) of L is valid for a large step s. The proposed algorithm MFPA is shown in Algorithm 1. Note that in the algorithm, we use the value of c ∈ [0.25, 0.75], since if the value of c is close to 0 or 1, then the next iterate with (1) will not progress. Furthermore, we modify a in equation (2) depending on steps.

If s > 1.5R, then

a = max(2*a, 1) and R = max(2*R, R_{max});

If s < 0.5R, then

a = min(a/2, 1.0e-4), and R = max(R/2, R_{min});

where R is the trust-region radius. R_{max} = 1000, R_{min} = 1.0e-4 with an initial R = 1;

Algorithm 1: Modified flower pollination algorithm (MFPA)

Input: Parameters, P and Set of values for each parameter, V = [v0 ..vj];
Output: Final test suite;

1. Initialize a set of n-candidate tests;
2. Find the best solution g* of n-candidate tests.
3. Generate all possible interactions based on P and V and set the results in a list T
4. while (T is not empty) do
5. while (t < MaxGeneration) do
6. for i = 1 : n (all n flowers in the population)
7. if rand() < p,
8. Compute L using (3)
9. Compute x_{t+1}^{i} using (2)
10. else
11. Draw c from a uniform distribution in [0,1]
12. Compute x_{t+1}^{i} using (1)
13. end if
14. Evaluate new solutions
15. If new solutions are better, update them in the population
16. end for
17. Find the current best solution g*
18. end while
19. Add the best test case, g* in final test cases .
20. Remove covered interactions elements from T.
21. End while

IV. TEST RESULTS

The adaptive flower pollination algorithm was implemented in Java with NetBeans environment under Windows 7 operating system. The implemented program was run on *intel core i*7-2640 CPU with 4GB RAM. We tested the program through two experiments in software tests. Two experimental results are presented to demonstrate the efficacy of our algorithm. Tables 1 and 3 provide the experimental setups where the problems PB1 to PB5 are defined according to the *t*-way of interaction with *t* = 2 to 6 which are used during the experiments.

Table 3 lists the test suite size for different problems with different degree interactions given in Table 1. In the table, columns 3-7 provide the test suite size from existing algorithms, i.e., HSS, IPOG, WHITCH, Jenny and TConfig [18-22]. The result of the proposed algorithm is reported in column 8. Best of the results obtained from all the algorithms is given in column 9. It can be noticed that our algorithm performs slightly better than the other algorithms. This can be checked by the total values given in the last row of the table. For the problems PB1, PB3-PB5, our algorithm outperforms the others. The second best results are achieved by HSS while WHIPBH even fails to execute on problems PB4-PB5, performing the worst.

Table 1: Experimental setup for Experiment 1

Problem No.	*t*	INPUT SPECIFICATIONS
PB1	2	7 3-valued parameters
PB2	3	7 3-valued parameters
PB3	4	7 3-valued parameters
PB4	5	7 3-valued parameters
PB5	6	7 3-valued parameters

Table 2: Test suite Size for different *t*-way interaction for experiment 1: seven input parameters with three possible values for each parameter

Problem No.	*t*	HSS	IPOG	WHIPBH	Jenny	PConfig	MFPA	BEST
PB1	2	**14**	17	15	16	15	**14**	**14**
PB2	3	50	57	**45**	51	55	50	**45**
PB3	4	**157**	185	216	169	166	**150**	**150**
PB4	5	437	561	NA	458	477	**425**	**425**
PB5	6	**916**	1281	NA	1087	921	**905**	**905**

Total	1574	2101		1781	1634	1544	1539

Table 3: Experimental setup for Experiment 2

Problem No.	*t*	INPUT SPECIFICATIONS
PB1	2	10 2-valued parameters
PB2	3	10 2-valued parameters
PB3	4	10 2-valued parameters
PB4	5	10 2-valued parameters
PB5	6	10 2-valued parameters

Table 4: Test suite Size for different t-way interaction for experiment 2: ten input parameters with two possible values for each parameter

Problem No.	*t*	HSS	IPOG	WHIPBH	Jenny	PBonfig	MFPA	BEST
PB1	2	8	10	**6**	10	9	**7**	**6**
PB2	3	**16**	19	18	18	20	17	**16**
PB3	4	**37**	49	58	39	45	38	**37**
PB4	5	**81**	128	NA	87	95	**78**	**78**
PB5	6	158	352	NA	169	183	**155**	**155**
Total		300	558		323	352	295	292

Table 4 provides the results from experiment 2, of which setups are given in Table 3. Similar to Table 2, Table 4 shows the test suite size for some existing algorithms along with our algorithm MFPA. The total value of suite size for MFPA is slightly better than its nearest competitor HSS. Again, WHIPBH makes the worst performance, failing to execute problems PB4-PB5.

Evaluating the experimental results, one can summarize that MFPA performs better than other algorithms and thus MFPA is a good candidate for using in reduction of test cases redundancy.

V. CONCLUSION

In this paper, we proposed an optimization technique MFPA by modifying the original flower pollination algorithm. MFPA provides an optimal test suite for software testing with the minimum redundancy. This proposed algorithm has faster convergence rate than the original algorithm. The test results of our algorithm were compared with existing results of some existing algorithms to demonstrate the strength of our algorithm. Comparison shows that our algorithm performs slightly better than the existing algorithms and thus, the proposed algorithm can potentially be used by researchers and test engineers to obtain optimal test suite requiring the minimum time for software testing. However, more efficient optimization algorithm is needed to solve the large problem with massive computational cost and to meet the requirements of accuracy and convergence. Therefore, new ideas, strategies and algorithms are necessary to improve the current state-of-the-art research in the area of global optimization.

ACKNOWLEDGMENT

This work was supported by Fundamental Research Grant Scheme (FRGS) No. RDU160102 from Ministry of Higher Education, Malaysia.

REFERENCES

1 Kabir, M.N., Issa-Salwe, A., and Ahmed, M.: 'Simplified model of combustion process in a diesel engine for real-time operation', International Journal of Vehicle Systems Modelling and Testing, 2010, 5, (4), pp. 347-357

2 Alsewari, A.R.A., and Zamli, K.Z.: 'Design and implementation of a harmony-search-based variable-strength t-way testing strategy with constraints support', Information and Software Technology, 2012, 54, (6), pp. 553-568

3 Hervieu, A., Marijan, D., Gotlieb, A., and Baudry, B.: 'Practical minimization of pairwise-covering test configurations using constraint programming', Information and Software Technology, 2016, 71, pp. 129-146

4 Zamli, K.Z., Alkazemi, B.Y., and Kendall, G.: 'A Tabu Search hyper-heuristic strategy for t-way test suite generation', Applied Soft Computing, 2016, 44, pp. 57-74

5 Gosciniak, I.: 'A new approach to particle swarm optimization algorithm', Expert Systems with Applications, 2015, 42, (2), pp. 844-854

6 Mahmoud, T., and Ahmed, B.S.: 'An efficient strategy for covering array construction with fuzzy logic-based adaptive swarm optimization for software testing use', Expert Systems with Applications, 2015, 42, (22), pp. 8753-8765

7 Yoo, S., and Harman, M.: 'Regression testing minimization, selection and prioritization: a survey', Software Testing, Verification and Reliability, 2012, 22, (2), pp. 67-120

8 Galeebathullah, B., and Indumathi, C.: 'A novel approach for controlling a size of a test suite with simple technique', Int. J. Comput. Sci. Eng, 2010, 2, pp. 614-618

9 Fraser, G., and Arcuri, A.: 'Achieving scalable mutation-based generation of whole test suites', Empirical Software Engineering, 2015, 20, (3), pp. 783-812

10 Patrick, M.: 'Metaheuristic Optimisation and Mutation-Driven Test Data Generation': 'Computational Intelligence and Quantitative Software Engineering' (Springer, 2016), pp. 89-115

11 Ahmed, B.S., Abdulsamad, T.S., and Potrus, M.Y.: 'Achievement of minimized combinatorial test suite for configuration-aware software functional testing using the Cuckoo Search algorithm', Information and Software Technology, 2015, 66, pp. 13-29

12 Jin, C., and Jin, S.-W.: 'Prediction approach of software fault-proneness based on hybrid artificial neural network and quantum particle swarm optimization', Applied Soft Computing, 2015, 35, pp. 717-725
13 Jensi, R., and Jiji, G.W.: 'An enhanced particle swarm optimization with levy flight for global optimization', Applied Soft Computing, 2016, 43, pp. 248-261
14 Ouyang, H.-b., Gao, L.-q., Kong, X.-y., Li, S., and Zou, D.-x.: 'Hybrid harmony search particle swarm optimization with global dimension selection', Information Sciences, 2016, 346, pp. 318-337
15 Yu, L., Lei, Y., Nourozborazjany, M., Kacker, R.N., and Kuhn, D.R.: 'An efficient algorithm for constraint handling in combinatorial test generation', in Editor (Ed.)^(Eds.): 'Book An efficient algorithm for constraint handling in combinatorial test generation' (IEEE, 2013, edn.), pp. 242-251
16 Zhang, Z., Yan, J., Zhao, Y., and Zhang, J.: 'Generating combinatorial test suite using combinatorial optimization', Journal of Systems and Software, 2014, 98, pp. 191-207
17 Yang, X.-S.: 'Flower Pollination Algorithm for Global Optimization': 'Unconventional computation and natural computation' (Springer, 2012), pp. 240-249
18 Ahmed, B.S., and Zamli, K.Z.: 'PSTG: A T-Way Strategy Adopting Particle Swarm Optimization', in Editor (Ed.)^(Eds.): 'Book PSTG: A T-Way Strategy Adopting Particle Swarm Optimization' (IEEE Computer Society, 2010, edn.), pp. 1-5
19 Ahmed, B.S., and Zamli, K.Z.: 'T-Way Test Data Generation Strategy Based on Particle Swarm Optimization', in Editor (Ed.)^(Eds.): 'Book T-Way Test Data Generation Strategy Based on Particle Swarm Optimization' (IEEE Computer Society, 2010, edn.), pp. 93-97
20 Ahmed, B.S., Zamli, K.Z., and Lim, C.P.: 'Constructing a T-Way Interaction Test Suite Using the Particle Swarm Optimization Approach', International Journal of Innovative Computing, Information and Control, 2012, 8, (1), pp. 431-452
21 Shiba, T., Tsuchiya, T., and Kikuno, T.: 'Using Artificial Life Techniques to Generate Test Cases for Combinatorial Testing', in Editor (Ed.)^(Eds.): 'Book Using Artificial Life Techniques to Generate Test Cases for Combinatorial Testing' (IEEE Computer Society, 2004, edn.), pp. 72-77
22 Alsewari, A.A., and Zamli, K.Z.: 'Interaction Test Data Generation Using Harmony Search Algorithm', in Editor (Ed.)^(Eds.): 'Book Interaction Test Data Generation Using Harmony Search Algorithm' (IEEE Computer Society, 2011, edn.), pp.