International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 09, September-2021

To Design and Develop A Particle Swarm Optimization (PSO) based Test Suite Optimization Scheme

Kanika Budhwar[1], Dr. Pradeep Kumar Bhatia[2], Dr. O. P. Sangwan[3]
Deptt. of CSE
G.J. University of Science & Technology, Hisar, Haryana, India

***Abstract* :- The majority of search-based works focus on single-objective selection. In this light, we developed a mechanism for functional TC selection which considers two objectives simultaneously: maximize requirements' coverage while minimizing cost in terms of TC execution effort. This mechanism was implemented as a multi-objective optimization process based on Particle Swarm Optimization (PSO). We implemented two multi-objective versions of PSO (BMOPSO and BMOPSO-CDR). The experiments were performed on two real test suites, revealing very satisfactory results (attesting the feasibility of the proposed approach). We highlight that execution effort is an important aspect in the testing process, and it has not been used in a multi-objective way together with requirements coverage for functional TC selection.**

Keywords: Multi objective optimization; Test case selection; Software testing.

INTRODUCTION

In a competitive world, where costumers search for high quality products, the software testing activity has grown in importance, aiming to assure quality and reliability to the product under development. Nevertheless, this activity is sometimes neglected, since it is very expensive, reaching up to 40% of the final software development cost [1]. Thus, it is of central importance to improve the efficiency and effectiveness of the whole testing process.

We can identify in the related literature two main approaches for software testing: structural (white box) testing, which investigates the behavior of the software directly accessing its code; and functional (black box) testing, which investigates whether the software functionalities are responding/behaving as expected [2]. In both approaches, software testing is conducted via the execution of a Test Suite, that is, a set of Test Cases. A Test Case (TC), in turn, consists of a set of inputs, execution conditions, and a pass/fail condition [2].

Before starting the testing process, it is necessary to choose a test adequacy criterion in order to evaluate whether a suite is adequate with respect to the test goals [3]. A test adequacy criterion defines properties that must be observed by the test suite (e.g., code coverage, functional requirements' coverage, among others).

Several tools have been proposed to automate or optimize software testing tasks, from test generation to its execution. Regarding automatic TC generation, we can identify tools which generate test suites from some software artifact (such as code, functional requirements, use cases) [4], [5]. However, as these tools generate TCs in a systematic way (aiming to provide a good coverage of the testing adequacy criterion), the generated test suites are usually too large to be executed with the available resources (tools, time, people) [6].

When analyzing large test suits, we can identify redundancies in TCs. Hence, it is possible to cut down these suits, in order to fit the available resources, without severely compromising the coverage of the test adequacy criterion being observed. The task of reducing a test suite based on some selection criterion is known as Test Case selection [7]. The test selection criterion depends on the test adequacy criterion being used.

Clearly, TC selection should not be performed at random, in order to preserve the coverage of the testing criterion. In the absence of automatic tools, this task is usually manually performed in an ad-hoc fashion. However, manual TC selection is time-consuming and susceptible to errors. Different authors try to automatically solve the TC selection problem by deploying a variety of techniques. Some works focus on deterministic software engineering solutions (see [7]). Despite their good results, these works consider only a single criterion for TC selection. Yet, they are computationally expensive when dealing with large test suites.

On the other hand, some authors deploy artificial intelligence search-based techniques to find a subset of TCs which optimizes a given objective function (i.e., a given selection criterion). Yet, the majority of search-based works focus on single-objective selection (e.g., [6]'s work using Greedy Search for structural TC selection; and [8]'s work using PSO for functional TC selection). Regarding multi objective selection, we cite Yoo and Harman's [9] work, which uses Genetic Algorithms for structural TC selection. In this light, we developed a mechanism for functional TC selection which considers two objectives (selection criteria) simultaneously: maximize requirements' coverage (quality of the test suite) while minimizing cost in terms of execution effort (i.e., the time necessary to execute the TC).

This work investigated the use of Particle Swarm Optimization (PSO) [10] for multi-objective TC selection. An increasing attention has been given to PSO in recent years, motivated by its success in different problems when compared to Genetic

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 09, September-2021

Algorithms and other search techniques [19]. We implemented two different algorithms: a Binary Multi-Objective PSO (BMOPSO) [11], [12]; and a modified version, known as BMOPSO-CDR (which uses Crowding Distance and Roulette Wheel) [13]. These algorithms provide to the user a set of solutions (test suites) with different values of requirements' coverage versus execution effort. The user can then choose the solution that best fits the available resources for executing the tests. The implemented algorithms were executed on two different real-world test suites of functional TCs related to the mobile devices domain. They were evaluated using five well known multi objective metrics. The performed experiments obtained very satisfactory results.

SEARCH-BASED TEST CASE SELECTION

According to Lin [6] Test Case selection can be treated as an optimization problem in which search techniques explore the space of possible solutions (subsets of TCs) seeking the solution that best matches the test objectives. Nevertheless, the returned subset may not have the same coverage of the original test suite with respect to the test adequacy criterion being used.

Deterministic approaches do not seem suitable for TC selection, since this is known to be an NP-Complete problem [6]. Thus, heuristic search approach appears to be more feasible to treat this problem. Within this approach, we cite the HGS algorithm [14], the GRE algorithm [15], Greedy Search heuristics for set-covering [16], Genetic Algorithms [9] and PSO [8].

Besides the technique to be used in TC selection, another central issue is the test selection criterion. For structural testing, the most commonly used selection criterion consists of the amount of pieces of program code (e.g., blocks, statements, decisions) that should be exercised by TCs [14]. In turn, for functional testing, the most usual criterion is the amount of functional requirements covered by the selected TCs [7], [8].

In fact, in the related literature we identified few works which take into account the cost (effort) to execute the test cases [9], [17], [8]. For functional tests, in particular, it is difficult to consider the TC execution cost because, for that, it is necessary to estimate the cost of manually performing the TC [18].

Finally, we see that the majority of existing works in TC selection considers only one selection criterion at a time, using some single-objective method to optimize the chosen objective. However, for TC selection in general, it is desirable to consider more than one criterion at the same time (e.g., maximum requirements' coverage and minimum execution effort cost).

MULTI-OBJECTIVE PSO TO TEST CASE SELECTION

In this work, we propose a method that adopts Particle Swarm Optimization (PSO) to solve multi-objective TC selection problems. In contrast to single-objective problems, Multi-Objective Optimization (MOO) aims to optimize more than one objective at the same time.

We adopted in our work the PSO, which is a population based search approach, inspired by the behavior of birds flocks [10]. PSO has shown to be a simple and efficient algorithm compared to other search techniques, including for instance the widespread Genetic Algorithms [19]. The basic PSO algorithm starts its search process with a random population (also called swarm) of particles. Each particle represents a candidate solution for the problem being solved and it has four main attributes:

1) the position (t) in the search space (each position represents an individual solution for the optimization problem);
2) the current velocity (v), indicating a direction of movement in the search space;
3) the best position ($t^{\wedge}$) found by the particle (the cognitive component);
4) the best position ($g^{\wedge}$) found by the particle's neighborhood.

For a number of iterations, the particles fly through the search space, being influenced by their own experience $t^{\wedge}$ (particle memory) and by the experience of their neighbors $g^{\wedge}$ (social guide). Particles change position and velocity continuously, aiming to reach better positions and to improve the objective functions considered.

We developed in our work a Binary Multi-Objective PSO (BMOPSO) by merging two versions of PSO: (1) the binary version of PSO proposed in [11], since the solutions of the TC selection problem are represented as binary vectors; and (2) the MOPSO algorithm originally proposed in [12] to deal with multi-objective problems. Furthermore, we implemented the BMOPSO-CDR algorithm by adding the Crowding Distance and Roulette Wheel mechanism (as proposed by [13]) to the BMOPSO. The proposed algorithms were applied to select functional tests.

The objective functions to be optimized in our work are the functional requirements coverage and the execution effort of the selected TCs, in such a way that we maximize the first function and minimize the second one. The proposed methods return a set of non-dominated solutions (a Pareto front) considering the aforementioned objectives. Figure 1 illustrates a sample of Pareto front return by the proposed algorithms. Each dot in this figure represents the values of the objective functions for a different non-dominated solution returned in the MOO process. By receiving a set of diverse solutions, the user can choose the best one taking into account its current goals and available resources (e.g., the amount of time available at the moment to execute the test cases).

EXPERIMENTS AND RESULTS

This section presents the experiments performed in order to evaluate the algorithms implemented in our work. Two test suites, referred here as Suite 1 and Suite 2, from the context of mobile devices[2] were used on the performed experiments. Although both suites have the same size (80 TCs), it is possible to observe in table I that Suite 1 covers a higher number of requirements than Suite 2, and it is more complex since the total effort needed to execute its test cases is higher when compared to Suite 2.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 09, September-2021

Table I : Characteristics Of The Test Suites

	Suite 1	Suite 2
Total effort	1053.91 min	699.31 min
# of requirements	410	248
Redundancy	0.36%	14.09%
# of test case	80	80

The TCs in Suite 1 are less redundant[3], i.e. two distinct test cases rarely cover the same requirement. Hence, for Suite 1, it is expected to be more difficult to find non dominated solutions. In Suite 2, in turn, each test case individually covers a higher number of requirements, with a higher level of redundancy. This characteristic makes it easier to eliminate test cases in Suite 2.

METRICS

In our experiments, we evaluated the results (i.e., the Pareto fronts) obtained by the algorithms MOPSO and MOPSO-CDR for each test suite according to five different quality metrics usually adopted in the literature of multi objective optimization. The following metrics were adopted in this paper: Hyper volume, Spacing, Maximum Spread, Coverage and Coverage Difference. Each metric considers a different aspect of the Pareto front, as follows.

1) Hyper volume HV: this metric is defined by the hyper volume in the space of objectives covered by the obtained Pareto front [14]. For MOO with k objectives, HV is defined by:

$$HV = \left\{ \bigcup_i a_i \,|\, x_i \in \mathcal{P} \right\},$$

where x_i (i = 1; 2; :::; n) are the non-dominated solutions in the Pareto front (P), n is the number of solutions in P and ai is the hyper volume of the hypercube delimited by the position of solution x_i in the space of objectives and the origin. In practice, this metric computes the size of the dominated space, which is also called the area under the curve. A high value of hyper volume is desired in MOO problems.

2) Spacing (S): this metric estimates the diversity of solutions in the obtained Pareto front [15]. S is derived by computing the relative distance between adjacent solutions of the Pareto front as follows:

$$S = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (\bar{d} - d_i)^2},$$

where d_i is the distance between adjacent solutions to the solution x_i and d is the average distance between the adjacent solutions. S = 0 means that all solutions of the Pareto front are equally spaced. Hence, values of S near zero are preferred.

3) Maximum Spread (MS): this metric evaluates the maximum extension covered by the non-dominated solutions in the Pareto front [16]. MS is computed by eq.:

$$MS = \sqrt{\sum_{m=1}^{k} (max_{i=1}^{n} f_m^i - min_{i=1}^{n} f_m^i)^2},$$

where k is the number of objectives and f^i_m is the value of the m-th objective function for the i^{th} solution in the Pareto front. High values of this metric are preferred.

4) Coverage (C): this is a measure to pairwise comparison of different algorithms [217]. Let P^A and P^B be the Pareto fronts generated respectively by two techniques A and B. C is computed by eq.:

$$C(\mathcal{P}^A, \mathcal{P}^B) = \frac{|\{x' \in \mathcal{P}^B; \exists x \in \mathcal{P}^A : x \preceq x'\}|}{|\mathcal{P}^B|},$$

where P^A and P^B are two sets of non-dominated solutions.

The value $C(P^A;P^B) = 1$ means that all solutions in P^B are dominated by P^A. On the other hand, $C(P^A;P^B) = 0$ means that none of the solutions in P^B is dominated by P^A.

The Coverage metric indicates the amount of the solutions within the non-dominated set of the first algorithm which dominate the solutions within the non-dominated set of the second algorithm.

It is important to highlight that both $C(P^A;P^B)$ and $C(P^B;P^A)$ must be evaluated, since $C(P^A;P^B)$ is not necessarily equal to $1-C(P^B;P^A)$. If $0 < C(P^A;P^B) < 1$ and $0 < C(P^B;P^A) < 1$, then neither P^A totally dominates P^B nor P^B totally dominates P^A.

5) Coverage Difference (D): Let P^A and P^B be two sets of non-dominated solutions. The metric D, also proposed by Zitzler [17], is defined by:

$$D(P^A, P^B) = HV\,(P^A + P^B) - HV\,(P^B)$$

This metric gives the size of the space dominated by P^A but not dominated by P^B (regarding the objective space).

In addition, the D measure gives information about which set entirely dominates the other set, e.g., $D(P^A;P^B) = 0$ and $D(P^B;P^A) > 0$ means that P^A is dominated by P^B.

ALGORITHMS SETTINGS

In this section, we present the values of the parameters adopted for the MOPSO and the MOPSO-CDR algorithms.

We adopted the same values suggested by Coello et. al [12] and by Santana et. al [13]:

- number of particles: 20
- mutation rate: 0.5

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 09, September-2021

- inertia factor ω: linearly decreases from 0.9 to 0.4
- constants C_1 and C_2: 1.49
- EA's size: 200 solutions
- maximum fitness evaluations: 200,000

We also performed experiments using a purely random search method, as a basis of comparison. Despite its simplicity, the random search has the advantage of performing a uniform exploration of the search space, being very competitive in other software testing contexts (e.g., for test case generation [18]).

RESULTS

For statistical analysis purposes, we executed each search algorithm 60 times on each test suite. In each execution, a Pareto front was produced. The values of coverage and cost observed in the Pareto fronts were normalized since they are measured using different scales. After normalizing the Pareto fronts, all the evaluation metrics were computed. From Table II, we observe that the BMOPSO-CDR outperformed both BMOPSO and Random algorithms in terms of Hyper volume, Maximum Spread and Spacing for Suite 1. The gain obtained by the BMOPSO-CDR has shown to be statistically significant for all evaluated metrics (see Table III). For Suite 1, the BMOPSO-CDR covered a wider region of the search space and also generated a larger Pareto front with better spaced solutions. BMOPSO also statistically outperformed the Random algorithm in terms of Hyper volume and Maximum Spread, but it was equivalent to it considering the Spacing metric.

For Suite 2, the BMOPSO-CDR outperformed BMOPSO and Random algorithms in terms of Hyper volume and Maximum Spread (see tables IV and V). Thus, for Suite 2 the BMOPSO-CDR also covered a wider region of search space and obtained a larger Pareto front. However, considering the Spacing metric, all the algorithms were statistically equivalent. The BMOPSO was better than the Random algorithm only for the Hyper volume metric and they were statistically equal for Maximum Spread and Spacing. It is important to note that the Suite 2 is simpler than the Suite 1. Hence, it is easier to find good non-dominated solutions.

Table II : Mean value and standard deviation – Suite 1

	Hypervolume	Max. Spread	Spacing
BMOPSO	8.2 (0.11)	0.76 (0.04)	2.63E-4 (1.03E-4)
BMOPSO-CDR	8.44 (0.14)	0.85 (0.04)	2.11E-4 (7.5E-5)
Random	8.06 (0.12)	0.74 (0.03)	2.89E-4 (1.18E-4)

Table III : Difference matrix of Means – Suite 1

	BMOPSO	BMOPSO-CDR	Random
Hypervolume			
BMOPSO	0	-0.25*	0.14*
BMOPSO-CDR	0.25*	0	0.38*
Random	-0.14*	-0.38*	0
Maximum Spread			
BMOPSO	0	-0.09*	0.02*
BMOPSO-CDR	0.09*	0	0.11*
Random	-0.02*	-0.11*	0
Spacing			
BMOPSO	0	5E-5*	-3E-5
BMOPSO-CDR	-5E-5*	0	-8E-5*
Random	3E-5	8E-5*	0

Table IV : Mean value and Standard deviation – Suite 2

	Hypervolume	Max. Spread	Spacing
BMOPSO	9.56 (0.05)	0.62 (0.07)	6.01E-4 (3.74E-4)
BMOPSO-CDR	9.64 (0.04)	0.7 (0.07)	7.27E-4 (4.25E-4)
Random	9.48 (0.06)	0.61 (0.07)	7.01E-4 (4.03E-4)

Tables VI and VII show the results considering the Coverage and Difference Coverage. Regarding the Coverage metric, it is possible to see that both BMOPSO and BMOPSO-CDR outperformed the Random approach, dominating approximately 97% and 94% (respectively for Suite 1 and Suite 2) of the non-dominated solutions found by Random. In contrast, the Random non-dominated solutions dominated approximately only 2% and 4% (for both suites) of non-dominated solutions found by the BMOPSO and BMOPSO-CDR algorithms.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 09, September-2021

Table V : Difference Matrix of Means – Suite 2

	BMOPSO	BMOPSO-CDR	Random
Hypervolume			
BMOPSO	0	-0.09*	0.08*
BMOPSO-CDR	0.08*	0	0.17*
Random	-0.08*	-0.17*	0
Maximum Spread			
BMOPSO	0	-0.08*	0.008
BMOPSO-CDR	0.08*	0	0.09*
Random	-0.008	-0.09*	0
Spacing			
BMOPSO	0	-1.3E-4	-1E-4
BMOPSO-CDR	1.3E-4	0	3E-5
Random	1E-4	-3E-5	0

Table VI : Coverage and Coverage Difference – Mean value and standard deviation-Suite 1

Algorithm	BMOPSO	BMOPSO-CDR	Random
C(BMOPSO, -)	-	0.55 (0.1)	0.97 (0.02)
D(BMOPSO, -)	-	0.007 (0.01)	0.16 (0.13)
C(BMOPSO-CDR, -)	0.34 (0.09)	-	0.94 (0.04)
D(BMOPSO-CDR, -)	0.25 (0.15)	-	0.38 (0.16)
C(Random, -)	0.02 (0.03)	0.04 (0.04)	-
D(Random, -)	0.02 (0.05)	2.7E-4 (6.6E-4)	-

Table VII : Coverage and Coverage Difference – Mean value and standard deviation – Suite 2

Algorithm	BMOPSO	BMOPSO-CDR	Random
C(BMOPSO, -)	-	0.62 (0.13)	0.97 (0.03)
D(BMOPSO, -)	-	0.01 (0.01)	0.1 (0.06)
C(BMOPSO-CDR, -)	0.28 (0.13)	-	0.94 (0.06)
D(BMOPSO-CDR, -)	0.1 (0.06)	-	0.17 (0.07)
C(Random, -)	0.02 (0.05)	0.04 (0.05)	-
D(Random, -)	0.01 (0.03)	0.002 (0.009)	-

Different from the previous results, however, the BMOPSO was better than BMOPSOCDR on both suites. In fact, this was the only metric in which BMOPSO outperformed the BMOPSO-CDR. The results of the Difference Coverage metric also show that the BMOPSO-CDR approach outperformed BMOPSO and the Random method. It means that the BMOPSO-CDR covers larger regions of the search space that are not covered by the others.

CONCLUSION

In this work, we propose the use of multi-objective PSO for functional TC selection. The main contribution of the current work was to investigate PSO in a multi-objective way for selecting functional test cases considering both requirements coverage and execution effort. To best of our knowledge, multi-objective PSO was not yet investigated in the context of TC selection. We highlight that the developed method can be adapted to other test selection criteria and it is not limited to two objective functions. Furthermore, we expect that these good results can also be obtained on other application domains.

REFERENCES:

[1] R. Ramler and K. Wolfmaier, "Economic perspectives in test automation - balancing automated and manual testing with opportunity cost," in Workshop on Automation of Software Test, ICSE 2006, 2006.
[2] M. Young and M. Pezze, Software Testing and Analysis: Process, Principles and Techniques. John Wiley & Sons, 2005.
[3] A. Andrews, R. France, S. Ghosh, and G. Craig, "Test adequacy criteria for uml design models," Journal of Software Testing, Verification and Reliability, vol. 13, pp. 95–127, 2003.
[4] P. Borba, D. Torres, R. Marques, and L. Wetzel, "Target - test and requirements generation tool," in Motorola's 2007 Innovation Conference (IC'2007), 2007.
[5] Y. Kissoum and Z. Sahnoun, "A formal approach for functional and structural test case generation in multi-agent systems," in IEEE/ACS International Conference on Computer Systems and Applications, 2007, 2007, pp. 76–83.
[6] J.-W. Lin and C.-Y. Huang, "Analysis of test suite reduction with enhanced tie-breaking techniques," Inf. Softw. Technol., vol. 51, no. 4, pp. 679–690, 2009.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 09, September-2021

[7] P. Borba, A. Cavalcanti, A. Sampaio, and J. Woodcock, Eds., Testing Techniques in Software Engineering, Second Pernambuco Summer School on Software Engineering, PSSE 2007, Recife, Brazil, December 3-7, 2007, Revised Lectures, ser. Lecture Notes in Computer Science, vol. 6153. Springer, 2010.
[8] L. S. Souza, R. B. C. Prudencio, and F. d. A. Barros, "A constrained particle swarm optimization approach for test case selection," in In Proceedings of the 22nd International Conference on Software Engineering and Knowledge Engineering (SEKE 2010), Redwood City, CA, USA, 2010.
[9] S. Yoo and M. Harman, "Pareto efficient multi-objective test case selection," in Proceedings of the 2007 International Symposium on Software Testing and Analysis, 2007, pp. 140–150.
[10] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in Proceedings of the IEEE International Joint Conference on Neural Networks, 1995, pp. 1942–1948.
[11] ——, "A discrete binary version of the particle swarm algorithm," in Proceedings of the World Multiconference on Systemics, Cybernetics and Informatics, 1997, pp. 4104– 4109.
[12] C. Coello, G. Pulido, and M. Lechuga, "Handling multiple objectives with particle swarm optimization," IEEE Transactions on Evolutionary Computation, vol. 8, no. 3, pp. 256– 279, 2004.
[13] R. A. Santana, M. R. Pontes, and C. J. A. Bastos-Filho, "A multiple objective particle swarm optimization approach using crowding distance and roulette wheel," in Proceedings of the 2009 Ninth International Conference on Intelligent Systems Design and Applications, ser. ISDA '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 237–242.
[14] E. Zitzler and L. Thiele, "Multiobjective optimization using evolutionary algorithms - a comparative case study," in Conference on Parallel Problem Solvin from Nature Algorithms (PPSN V), 1998, pp. 292–301.
[15] J. R. Schott, "Fault tolerant design using single and multicriteria genetic algorithms," Master's thesis, Department of Aeronautics and Astrounatics, Massacgusettes Institute of Technology, Bosto, MA, 1995.
[16] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results," Evolutionary Computation Journal, vol. 8(2), pp. 125–148, 2000.
[17] E. Zitzler, "Evolutionary algorithms for multiobjective optimization: Methods and applications," Ph.D. dissertation, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, 1999.
[18] M. Takaki, D. Cavalcanti, R. Gheyi, J. Iyoda, M. d'Amorim, and R. B. Prud`encio, "Randomized constraint solvers: A comparative study," Innovations in Systems and Software Engineering: A NASA Journal, vol. 6, no. 3, pp. 243–253, 2010.