Journal of Theoretical and Applied Information Technology
31st October 2024. Vol.102. No. 20

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

THE METAHEURISTIC OF HYBRID EVOLUTIONARY WITH BLACK-HOLE ALGORITHM FOR COMBINATORIAL PRODUCT LINE TESTING

RABATUL ADUNI SULAIMAN[1*], NUREZAYANA ZAINAL[1], MAZIDAH MAT REJAB[1], NORHAMREEZA ABDUL HAMID[1], DAYANG NA JAWAWI[2], WAN NOOR HAMIZA WAN ALI[3]

[1]Department of Software Engineering, Faculty of Computer Science and Information System, Universiti Tun Hussein Onn Malaysia, 86400 Parit Raja, Johor, Malaysia
[2]Department of Software Engineering, Faculty of Computing, Universiti Teknologi Malaysia, 81110, Skudai, Johor, Malaysia
[3]Faculty of Artificial Intelligence, Universiti Teknologi Malaysia, Jalan Sultan Yahya Petra, 54100 Kuala Lumpur, Malaysia

E-mail: [1]rabatul@uthm.edu.my

ABSTRACT

Software testing is very challenging due to the demand of product quality. This process is related to an optimization procedure that neglects the cost and effectiveness measures of products. The existing method lacks search process, causing it unable to efficiently find optimal solution. The objective of this study is to propose and evaluate an experimental hybrid technique of Black-Hole (BH) with Binary Particle Swarm Algorithm (BPSO) called BH-BPSO using t-way combinatorial testing for test case generation in SPL. This study proposes the BH-BPSO which is based on modification and integration of BPSO and BH algorithm in the searching process. Evaluation of the proposed work is implemented based on four different sizes of SPL case studies. The result shows that the proposed BH-BPSO is comparatively efficient for test case generation. BH-BPSO managed to outperform existing methods based on total execution time, size of test suite, pairwise coverage and test case redundancy measure for large size case studies. It is concluded that this research shows the feasibility of the proposed approach in the SPL testing. This approach achieved improvement in terms of parallel metaheuristic measure for large size of case studies.

Keywords: *Software Testing, Metaheuristic Algorithm, Optimization Problem, Soft Computing, Software Product Line.*

1. INTRODUCTION

Software product line (SPL) which is also called software product line development, is a set of software engineering practices for making similar software systems from a single set of software assets and using the same production method for all of them [1]. In other words, SPL is a group of products that are put together from a set of features. A Feature Model (FM) decides which products are valid to be reused. Most of the time, it is not possible to test all the products. Thus, good quality products must be selected for test execution [2]. Many effective SPL testing techniques have been implemented, for example model-based testing [3]– [5] and search-based testing [6]–[9].

Effective testing strategies will help organizations to have good quality assets and at the same time can save testing costs. Retesting can be prevented should an effective testing technique was implemented [10]. It leads to a minimal cost of testing and helps an organization to minimize costs during software testing process [11]. The SPL assets keep increasing since assets keep developed causing efficient testing technique is required to be implemented. Testing an entire product line takes a long execution time and requires higher cost [12], thus, testing effectiveness can be improved by making test cases generation runs automatically. In order to automate test cases generation, more research needs to be done on SPL testing process because it is impossible to test all individual systems that make up a large SPL software system.

Test case generation in SPL is based on variation point management. Perrouin *et al*. 2012 [13] highlighted that SPL testing is hard, but it would be best if all SPL products were set up correctly. In fact, some features can be set up in tens of millions of different ways. Since there is pressure to make test suites for the whole product line smaller, it will be hard to test each product in an SPL and stay on budget. Combinatorial testing and other testing methods can cut down on the number of test suites needed, but they also come with their own problems with regard to scalability issue [14]. Combinatorial Testing (CT) offers covering array test suite testing by testing all required combinations of parameter values. CT is able to detect failures by interacting with parameters in the testing process. In SPL, CT has been implemented previously to automate test case generation. The goal of CT is to have a product without any bugs and can work with a wide range of inputs by covering array of the test suite.

Pairwise testing, also called "*all-pairs testing*" is a way to check software quality by comparing the actual results to what was expected. Here, software testers look at all possible pairs of parameters used to test a feature and compare them. Pairwise testing has become an important tool for software testers over the past few years. This method has been used for almost 20 years, but it has been in demand in the last five years. Pairwise testing verifies software by testing with permutations of two inputs. This method helps to understand how inputs interact, improving product quality and dependability. Pairwise testing is useful for testing SPL, which are collections of configurable products. Pairwise testing can improve product line testing and be applied to a case study [18]. Paired with other methods, this technique may reduce testing costs and improve quality [19]. The paper recommends pairwise testing to reduce test cases.

t-wise testing checks all input value permutations with a constraint of "*T*" inputs. This permutation process can help the tester to test many inputs. t-wise testing balances test case volume and coverage. SPL model-based *t*-wise testing creates a test suite with comprehensive *t*-wise coverage. A valid *t*-set has *t* features that meet some constraints. Covering arrays in SPL testing improves system failure detection. For testing, a two-layer covering array is used to represent equivalence classes and compute their names in the second layer. Covering arrays are used to test component interactions in a systematic manner. Let N, t, k, and v be integers with $k \geq t \geq 2$ and $v \geq 2$. A covering array CA (N; t, k, v) is an $N \times k$ array A in which each entry is from a different alphabet, and there is a row of B that equals x for every $N \times t$ subarray B of A and every $x \in \Sigma \wedge t$. Then t denotes the covering array's strength, k is the number of factors, and v is the number of levels. CT is also known as an optimization technique that is commonly used in search-based testing (SBT) [15]. In SBT implementation for CT, there are many SBT techniques which consider parallel metaheuristic outcome for test case generation. However, there are lack of parallel metaheuristic method with searching techniques implemented in SPL [16]. The current SBT for CT needs to be improved before this technique can be applied for parallel-based metaheuristic.

In SPL, SBT techniques have been documented related to searching technique for optimization problems. Examples of the implemented searching techniques are harmony search [17], evolutionary algorithm [9], [18], [19], genetic algorithm [20], and metaheuristic algorithm [12], [21]. Each technique consists of different searching processes and mechanisms. It is vital that the searching process of SBT is application dependent [15], [22] because it will result to some techniques to be not applicable for parallel metaheuristic. This study focuses on the improvement of existing metaheuristic algorithm in SPL for CT which is Black-Hole optimization and Binary Particle Swarm Algorithm (BH-BPSO). The goal of this study to generate test cases using proposed BH-BPSO technique in CIT for parallel metaheuristic. This study focuses on implementation of BH-BPSO in SPL testing. This study does not focus on other domains except for software testing. The contribution of this study can be summarized as follows:

• Develop an experimental hybrid technique called BH-BPSO for t-way combinatorial testing for test case generation.

• Evaluate the proposed BH-BPSO technique of parallel metaheuristic problem by using two SPL case studies.

The existing metaheuristic algorithm implemented in SPL focuses on single problem making it difficult to measure different problems at a single time, means more time is required for the testing process. In the scope of SPL, work by Guo *et al*. [23] presented SMTIBEA that joins two algorithms which are Indicator-Based Evolutionary Algorithm (IBEA) with Satisfiability Modulo Theories (SMT). However,

this proposed algorithm is only applicable for configurations of SPL constraints and not validated in testing domain context. In [24], a hybrid approach called '*Hyper-PSOBBO*' which is combination of Particles Swarm Algorithm (PSO), Biogeography-Based Optimization (BBO) and hyper-heuristic algorithms has been proposed. This technique differs from ours since it is used for regression test selection in SPL whereas our main goal is to generate test cases using CT. Our work is similar with Hitesh *et al.*[19] that proposed evolutionary algorithm for feature selection in SPL using genetic algorithm (GA). However, this technique is not applied the parallel metaheuristic concept since this proposed work considered domain dependent technique.

In the scope of SPL testing using hybrid technique, Yazan et al. [25] proposed Bat-inspired algorithm as a new *t*-way strategy using pairwise testing. This approach covers only effectiveness of the algorithm against the selected case study. There is lack of parallel metaheuristic implemented to investigate multiple measures of SPL core assets. Work by Abid et al. [18] proposed evolutionary algorithms for CT in SPL. Algorithms covered include Indicator Based Evolutionary Algorithm (IBEA), Strength Pareto Evolutionary algorithm II (SPEA-II), Multi-Objective Evolutionary Algorithms based on Decomposition (MOEA/D) and Non-Dominated Sorting Genetic Algorithm II (NSGA-II).

Although this hybrid algorithm covers parallel metaheuristics, there are still lacks evaluation on algorithm complexity covered. In summary, there is lack of implementation of hybrid techniques in SPL, especially on combinatorial testing for SPL. Implementation of hybrid techniques for test case generation using parallel metaheuristic can help to improve quality of test cases. In terms of CT, hybrid techniques can help as searching process for constraints configurations. This paper is structured as follows: Section 2 highlights the proposed method followed by empirical studies in Section 3. Section 4 presents the results whereas Section 5 focuses on discussion of the result. Finally, Section 6 focuses on conclusion and future works.

2. PROPOSED APPROACH

In SPL, there are three problems that encapsulate the main issues. First, huge number of conceivable industrial SPL products makes it impractical to individually check their conformity with each criterion. Second, it is impractical to execute an exhaustive test that accounts for every possible parameter and value combination. Lastly, the evaluation process and comparison also need to be considered.

2.1 Black-Hole Optimization and Binary Particle Swarm Algorithm

The process flow of the proposed approach is presented in Fig. 1. Based on the figure, the process starts with getting input from feature model (FM). FM input will be used to generate test cases. The approach starts by initializing populations with defined strength, covering array (V_A) and initial size of input from FM. The proposed BH-BPSO algorithm starts with size of particles, best particle (p_{best}), random position and fitness = *0*. Best particle and global best are defined based on two parameters which are velocity and position. These two parameters will evaluate the features input of FM. The objectives function of the proposed approach covers the parallel metaheuristic functions which are based on cost and effectiveness measures. In terms of cost, execution time and size of test suite are utilized as metrics whereas for effectiveness, pairwise coverage of CT will be used to present the quality of test suite.

Based on the initial values and parallel objectives defined, the first generation of X_i will be generated, at the same time, initiating the Black-Hole algorithm. Black-Hole is a search algorithm that can also be used to generate optimization objectives. This algorithm is a combination of binary version and CIT. The random generation result from PSO will be set as initial stars of the black-hole. In this phase, the star's position is evaluated. The selected star is based on the star with the highest fitness function (also known as black-hole point (BH_p)). This point is used to help stars move around. This searching process is defined as follows:

Based on equation 1, the fitness value with the better fitness will be selected as the best particle. Next, the latest point of BH_P will be used to evaluate the event horizon Eh. If the stars reach the defined limit, a new random start will be initialized, replacing the current value. The limit is set based on the defined objective functions in Section 2.2. Once the value of the maximum iteration is reached, the binary PSO process will be started. The value obtained will be used to rank the best particles. Once the maximum iterations have been reached, the process will end. The process flow of the proposed approach is

presented in pseudocode of BH-BPSO as in Algorithm 1.

1) EMPIRICAL STUDIES

The aim of this study is to assess the effectiveness of the proposed algorithm in CIT in order to achieve minimal test case generation cost (execution time and size of test suite) and effectiveness (pairwise coverage). In line with the aim, two research questions have been focused on which are:

- RQ1: How can BH-BPSO minimize the cost in terms of execution time and size of test suite?
- RQ2: How can BH-BPSO maximize effectiveness in terms of pairwise coverage and test case redundancy measures?

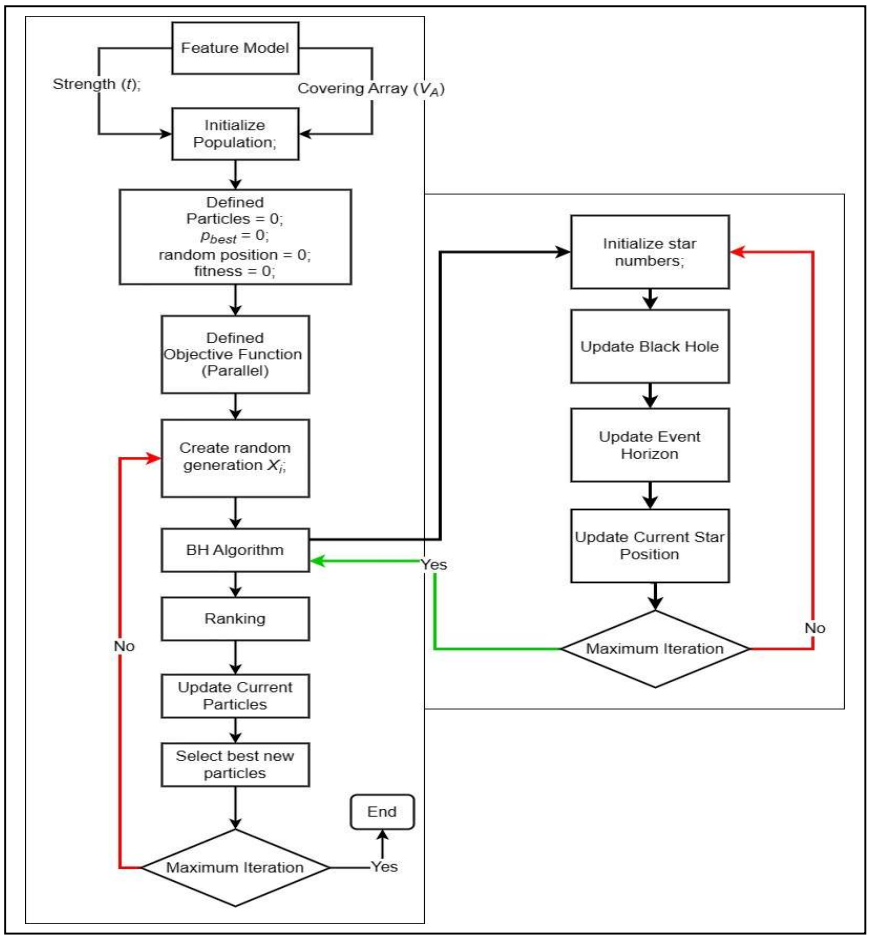

Fig 1: Process Flow of the Proposed Approach.

Let
i = star;
random
Xi is random generation;
d = dimension.

$$Xi\ (t+1) = Xi\ (t) \times (BH_p - Xi\ (t)) \quad \text{where } i = 1,2..n \tag{1}$$

Algorithm 1: Pseudocode for BH-BPSO

Input: *Set of covering array (V_A) from FM input*
Output: *Set of generated test case.*
1. ***Loop:***
2. *Initialize P = 0;*
3. *Initialize P_{best} = 0;*
4. *Initialize Random Position = 0;*
5. *Initialize Fitness = 0;*
6. ***For** each P = [P_1,P_2,…,P_n] **do***
7. *Set $\sum_{Time}$ = 0;*
8. *Set n_x = 0;*
9. *Set PC_{TS} = 0;*
10. *Set* **RScore** = *0;*
11. Let the X_i in random
12. population order ***do***
13. Initialize star population
14. numbers.
15. Update the best fitness
16. Values of $\sum_{Time}$, n_x, PC_{TS},
17. **RScore**
18. Change the location of stars
19. **If** a star >= fitness value then
20. Exchange their location();
21. Update Event Horizon();
22. Update Current Star
23. position();
24. **Else**
25. Maintain location();
26. Maintain iteration;
27. Update ranking value();
28. Update current particles();
29. Select new P_{best}();
30. *Get Xi*
31. **If** X_i uncovered in X **then**
32. Return to (6);
33. **Else**
34. Terminate();
35. **End**;

3.1 Objective Functions

Objective functions present the measurement that will be used by algorithms to fulfill the theoretical concepts that have been discussed. Parallel metaheuristic covers the multiple objectives that can be fulfilled at one time and help to save implementation cost. Details of the objective functions can be found in Table 1.

3.2 Case Studies

The proposed algorithm used four different types of case studies in order to evaluate its effectiveness. The four case studies are Mobile Phone, Vending Machine, Online Shop and IoT Device. Inputs from these case studies are used in the proposed algorithm. These case studies need to be drawn using SPLOT, an SPL online tool, to obtain the features input. Table 2 presents the summary of the case studies. There are two case studies with small sizes, one with medium size and one with large size. Details for each case study are described in the following subsections.

Table 1: Objective Functions

Function	Equation	Description
Execution time	$ET = \sum_{Time}$	where *Time* is the total generation time
Size of Test Suite	$TS = n_x$	where *n* is the list of test cases in test suite
Pairwise Coverage	$PC_{TS} = \frac{NumPair_{pi}}{NumberAllPair_{pi}}$	where $NumPair_{pi}$ is the unique pairwise covered in selected test suite whereas $NumberAllPair_{pi}$ is the total of all pairs covered for testing
Test Case Redundancy	$RScore = \sum^{Redundant\ test\ cases}_{Test\ cases}$	where *Redundant* test cases is total of redundant test cases

3.3 Parameters Tuning

Settings of the proposed algorithm are presented in Table 3. To evaluate the proposed BH-BPSO, we used NetBeans and programmed the BH-BPSO based on defined objective functions. The experiments are conducted using 12th Gen Intel(R) Core (TM) i7-1255U 1.70 GHz processor. In terms of benchmark setup, the code is not publicly available to be reused since the setting might be slightly different, which is out of our control. For comparison purposes, one existing study was used to compare with the proposed work. Similar settings are used for these two approaches.

Table 2: Summary of Case Studies.

Case Study	Number of Features	Industry	Size
Mobile Phone [26]	10	Technology	Small
Vending Machine [27]	6	Business	Small
Online Shop [28]	22	Business	Medium
IoT Device [29]	41	Technology	Large

Table 3: Parameters Settings for the Proposed Approach.

Parameter	Range	Description
Population Size	*{0,..,100}*	To set the maximum size of population can be generated by PSO.

Number of Particles	*{0,..,80}*	To set the number of maximum particles.
Minimum Velocity	*5*	To define the minimum velocity of PSO.
Maximum Velocity	*10*	To define the maximum velocity of PSO.
Number of Stars	*{0,..,80}*	To set the number of stars that can be made from enhanced black-hole.
Number of Black-Holes	*{0,..,10}*	To set the number of black-hole that can be obtained.
Maximum Iterations	*{0,..,100}*	To set the maximum looping that can be reached in single generation.

3.4 Statistical Analysis

The evaluation of the result obtained will be evaluated based on statistical analysis with 95% confident level (α = 0.05). In order to handle this analysis, we adopt Mann Whitney U-test to discover the statistical difference with respect to the existing approach. The Mann Whitney U-test was selected in this research since the fact that the attained results are not normally distributed (non-parametric). Hypothesis of this work are defined as follows:

H_0 = There is no significant difference of the proposed BH-BPSO against existing approaches.

To control Type 1 family wise error in result comparison, we have adopted Holm-Bonferroni to adjust α value. This validation is based on the probability of making one or more false in statistical testing. If the p value is less than the a_{Holm}, the hypothesis will be accepted and vice versa.

4. RESULTS

Four case studies are used to compare the results of the proposed BH-BPSO approach with existing approaches in [30]. Based on defined research questions in Section 3, the following section will discuss each result accordingly.

4.1 RQ1: How can BH-BPSO minimize the cost in terms of execution time and size of test suite?

This section compares results based on test cases generation on four different product line case studies. Results of this study are summarized by using dot-boxplot as in Fig. 2 and Fig. 3. In order to get the actual trend, 20 executions have been implemented to obtain concrete outcomes. According to the results, we observe that for test case execution time, the work by [30] outperforms the proposed BH-BPSO in two case studies which are Vending Machine and Mobile Phone. For the other two case studies, which are Online Shop and IoT Device, the result reflects that the proposed BH-BPSO outperforms the existing study. As for size of test suite, the minimal size of test suite of Vending Machine case study can be obtained by using [30] whereas for the Mobile Phone, Online Shop and IoT Device case studies, the proposed BH-BPSO outperforms the existing study. As seen in Fig.3, the result obtained is very competitive. In fact, the BH-BPSO can generate eight out of 12 maximum test cases for Mobile Phone, 18 out of 26 maximum test cases for Online Shop and 39 out of maximum 51 test cases for IoT Device.

4.2 RQ2: How can BH-BPSO maximize effectiveness in terms of pairwise coverage and test case redundancy measures?

In Fig. 4 and Table 4, we present the effectiveness of test cases generation in terms of pairwise coverage. Based on the result, we found that for Vending Machine case study, the pairwise coverage is from 80% to 100% with average value of 94.15% for [30] and 93.0% for BH-BPSO. For Mobile Phone, the pairwise coverage is from 78% to100% with similar average values of 90% for both BH-BPSO and [30]. For Online Shop, the coverage is from 60% to 90% where BH-BPSO outperforms [30] with average of 77.73%. For IoT Device, the coverage is 50% to 80% with average values of 71.58% (BH-BPSO) and 58.65% ([30]).

Fig. 5 presents comparison in terms of test case redundancy results. We found that for Vending Machine, minimal redundancy was recorded (one redundant test case) for both BH-BPSO and [30]. Whereas for Mobile Phone, the BH-BPSO outperforms [30] with only two redundant test cases. For Online Shop, only one redundant test case separates BH-BPSO and [30] and for IoT Device, [30] recorded seven redundancy test cases whereas BH-BPSO reported six redundancy test cases.

Journal of Theoretical and Applied Information Technology
31st October 2024. Vol.102. No. 20

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

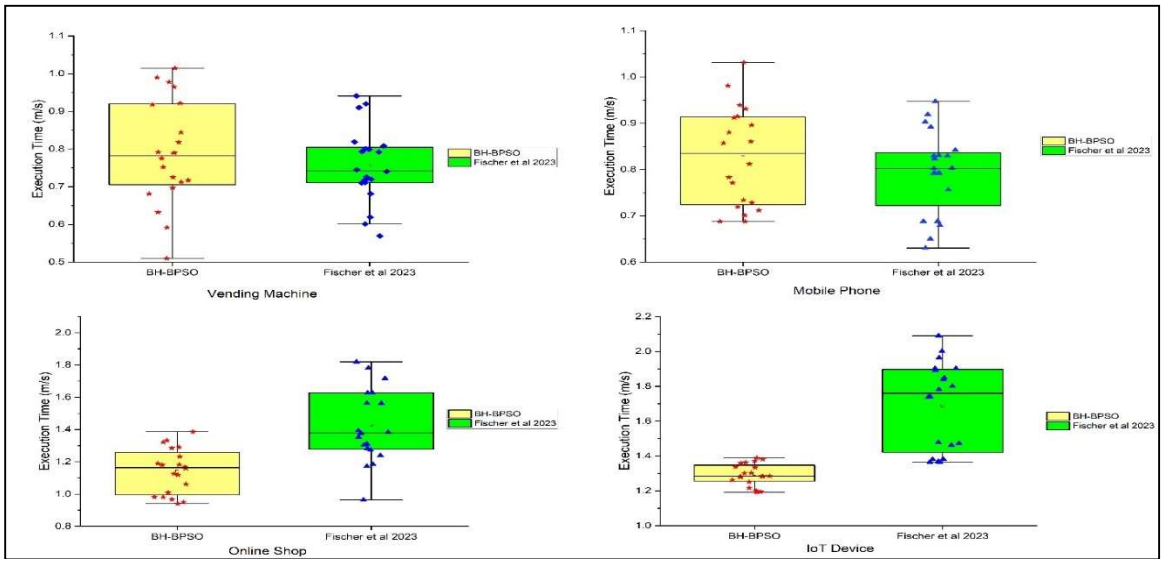

Fig. 2: Result Analysis of BH-BPSO against Fischer et al. (2023) for Execution Time.

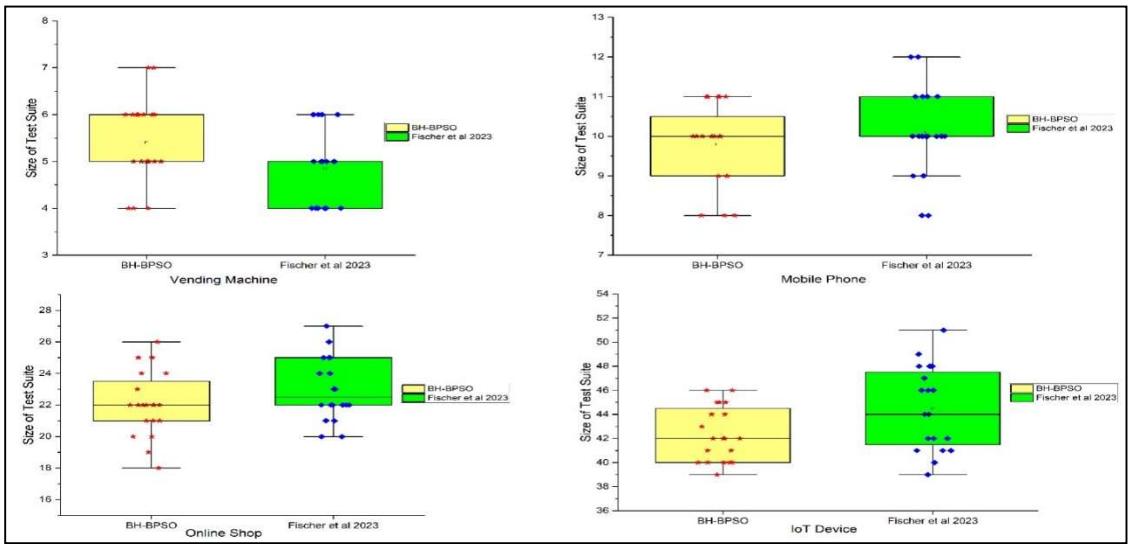

Fig. 3: Result Analysis of BH-BPSO against Fischer et al. (2023) for Size of Test Suite.

Journal of Theoretical and Applied Information Technology
31st October 2024. Vol.102. No. 20

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

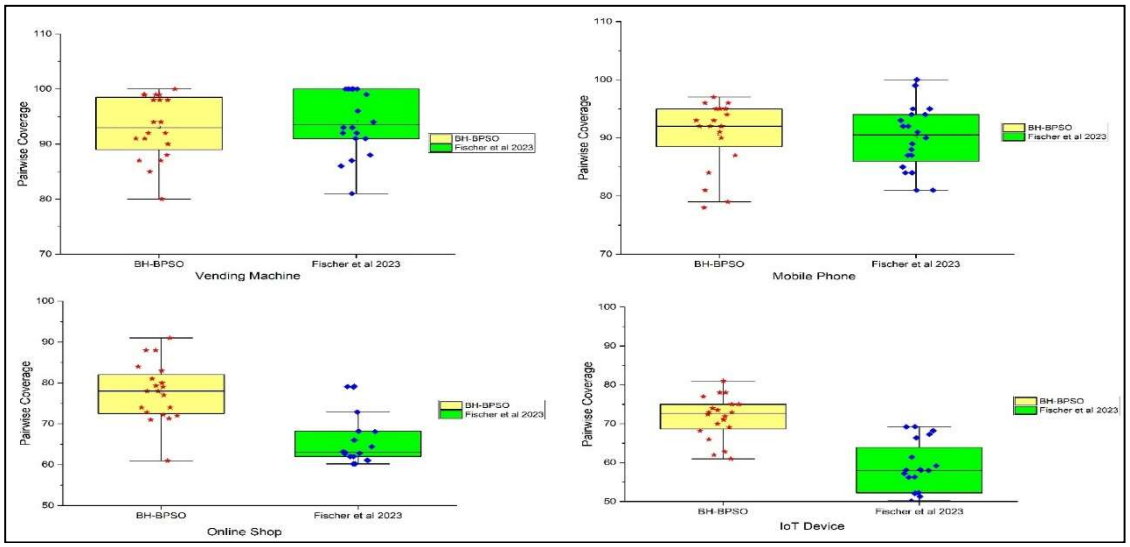

Fig. 4: Result Analysis of BH-BPSO against Fischer et al. (2023) for Pairwise Coverage.

Table 1: Pairwise Coverage Comparison Result.

	Vending Machine		Mobile Phone		Online Shop		IoT Device	
	BH-BPSO	Fischer *et al*. (2023)	BH-BPSO	Fischer *et al*. (2023)	BH-BPSO	Fischer *et al*. (2023)	BH-BPSO	Fischer *et al*. (2023)
	99.00	100.00	81.00	94.00	78.00	79.00	66.00	56.29
	98.00	96.00	97.00	91.00	88.00	62.80	70.00	58.00
	92.00	91.00	84.00	94.00	71.00	61.10	74.00	58.00
	100.00	94.00	87.00	81.00	72.00	68.10	75.00	59.20
	87.00	93.00	92.00	84.00	74.00	66.00	77.00	52.20
	80.00	92.00	92.00	85.00	77.00	62.70	78.00	58.10
	99.00	87.00	95.00	88.00	78.00	60.20	78.00	52.10
	99.00	100.00	78.00	92.00	88.00	62.00	72.40	61.40
	98.00	88.00	95.00	95.00	80.00	64.40	71.00	68.20
	98.00	81.00	79.00	87.00	91.00	62.00	75.00	69.24
	91.00	100.00	96.00	89.00	72.80	79.20	72.90	52.20
	91.00	100.00	93.00	100.00	84.00	68.20	68.20	58.20
	92.00	99.00	95.00	90.00	61.00	61.04	69.10	67.30
	90.00	93.00	96.00	92.00	74.00	63.00	72.90	69.20
	85.00	100.00	92.00	95.00	81.00	60.20	62.00	66.40
	99.00	86.00	91.00	93.00	83.00	63.20	81.00	57.20
	94.00	100.00	90.00	87.00	72.20	62.00	71.90	50.14
	87.00	91.00	92.00	99.00	79.00	72.90	62.80	52.30
	88.00	100.00	94.00	84.00	71.30	79.10	61.00	56.20
	94.00	92.00	93.00	81.00	79.30	68.20	73.50	51.30
Average:	**93.05**	**94.15**	**90.60**	**90.05**	**77.73**	**66.26**	**71.58**	**58.65**

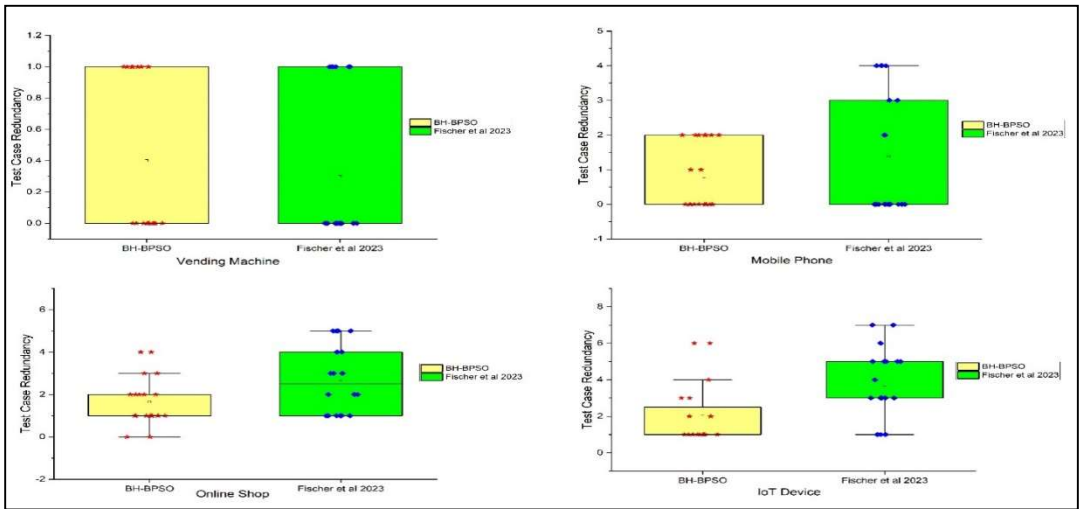

Fig. 5: Result Analysis of BH-BPSO against Fischer et al. (2023) for Test Case Redundancy.

5. DISCUSSION

In order to analyze our results in Section 4, we used the Mann Whitney U-Test (Wilcoxon two-sample test) which is a non-parametric test by using IBM SPSS. This statistical test was chosen since we have no proof for the data to be normally distributed. The aim of the test is to analyze if one approach is stochastically dominated by another approach or not. The null hypothesis of this test is stated in Section 3.4.

To prove our defined hypothesis and reject H_0, the statistical analysis result is presented in Table 5. The *U*-test compared the mean rank whereas *Z*-test is used to compare the differences of analysis in distributions and Wilcoxon rank sums. We have compared the BH-BPSO with an approach by Fischer *et al.* (2023). The mean rank of statistical refers to the average termination value for all execution. Here, the lower means rank presents the faster convergence of algorithm. As per result, for execution time, BH-BPSO for vending machine, online shop and IoT device case studies present the faster execution time with *p*-value implies significant ($p < 0.05$). Whereas for mobile phone case study, the *p*-value implies less significance of results ($p = 0.358$) that reflect the null hypothesis is accepted.

For size of test suite, statistical analysis reflects that for vending machine and IoT device the BH-BPSO implies significant and reject null hypothesis. However, two case studies which are mobile phone and online shop reflect null hypothesis accepted. In terms of pairwise coverage measure, result shows that for online shop and IoT device the *p*-value is less than 0.05 and null hypothesis is rejected. Whereas for two small case studies which are vending machine and mobile phone, the null hypothesis is accepted. In terms of test case redundancy measure, statistical analysis reflects that BH-BPSO using two large size of case studies which are online shop and IoT device implies significant of *p*-value < 0.05 and null hypothesis is rejected. Whereas for small size of case studies which are vending machine and mobile phone, result shows that both case studies less significant with *p*-value > 0.05 for BH-BPSO and null hypothesis is accepted.

Reflecting on the undertaken study, there are some analyses that can be discussed more as a lesson is learned. We group the discussion section into two parts which are on cost and effectiveness. Concerning cost measures, we foresee BH-BPSO as a consistent algorithm that can evaluate the large number of features in SPL. In existing study by [30], the proposed method was able to generate good test cases for small number of case studies. For BH, the candidate solution process will randomly select and change the position to retrieve the best solutions. By having hybrid technique with PSO, the BH algorithm will achieve good search process. BH-BPSO is able to replace existing algorithms in searching for accurate test cases.

Table 2: Result of Statistical Analysis

Statistical Test for Comparison Approaches											
Evaluation Types	**Case Studies**	**Approaches**	**N**	**Mean Rank**	**Sum of Ranks**	**Evaluation Types**	**Case Studies**	**Approaches**	**N**	**Mean Rank**	**Sum of Ranks**
Execution Time (m/s)	Vending Machine	BH-BPSO	20	10.50	210.00	**Size of Test Suite**	Vending Machine	BH-BPSO	20	23.93	478.50
		Fisher	20	30.50	610.00			Fisher	20	17.08	341.50
	Test Statistics						**Test Statistics**				
	Mann-Whitney U		0.00				Mann-Whitney U		131.500		
	Wilcoxon		210.00				Wilcoxon		341.500		
	Z		-5.41				Z		-1.967		
	Asymp Sig. (2-tailed)		<0.001				Asymp Sig. (2-tailed)		0.048		
	Mobile Phone	BH-BPSO	20	22.20	444.00		Mobile Phone	BH-BPSO	20	19.15	383.00
		Fisher	20	18.80	376.00			Fisher	20	21.85	437.00
	Test Statistics						**Test Statistics**				
	Mann-Whitney U		166.00				Mann-Whitney U		173.00		
	Wilcoxon		376.00				Wilcoxon		383.00		
	Z		-0.920				Z		-0.779		
	Asymp Sig. (2-tailed)		0.358				Asymp Sig. (2-tailed)		0.478		
	Online Shop	BH-BPSO	20	13.40	268.00		Online Shop	BH-BPSO	20	17.80	356.00
		Fisher	20	27.60	552.00			Fisher	20	23.20	464.00
	Test Statistics						**Test Statistics**				
	Mann-Whitney U		58.00				Mann-Whitney U		146.00		
	Wilcoxon		268.00				Wilcoxon		356.00		
	Z		-3.842				Z		-1.491		
	Asymp Sig. (2-tailed)		0.001				Asymp Sig. (2-tailed)		0.149		
	IoT Device	BH-BPSO	20	11.10	222.00		IoT Device	BH-BPSO	20	16.80	336.00
		Fisher	20	29.90	598.00			Fisher	20	24.20	484.00
	Test Statistics						**Test Statistics**				
	Mann-Whitney U		12.00				Mann-Whitney U		126.00		
	Wilcoxon		222.00				Wilcoxon		336.00		
	Z		-5.085				Z		-2.017		
	Asymp Sig. (2-tailed)		<0.001				Asymp Sig. (2-tailed)		0.046		

Pairwise Coverage	Vending Machine	BH-BPSO	20	18.55	371.00	**Test Case Redundancy**	Vending Machine	BH-BPSO	20	21.50	430.00
		Fisher	20	22.45	449.00			Fisher	20	19.50	390.00
	Test Statistics						**Test Statistics**				
	Mann-Whitney U		161.00				Mann-Whitney U		180.00		
	Wilcoxon		371.00				Wilcoxon		390.00		
	Z		-1.062				Z		-0.655		
	Asymp Sig. (2-tailed)		0.301				Asymp Sig. (2-tailed)		0.602		
	Mobile Phone	BH-BPSO	20	21.74	435.00		Mobile Phone	BH-BPSO	20	19.00	380.00
		Fisher	20	19.25	385.00			Fisher	20	22.00	440.00
	Test Statistics						**Test Statistics**				
	Mann-Whitney U		175.00				Mann-Whitney U		170.00		
	Wilcoxon		385.00				Wilcoxon		380.00		
	Z		-0.679				Z		-0.898		
	Asymp Sig. (2-tailed)		0.512				Asymp Sig. (2-tailed)		0.429		
	Online Shop	BH-BPSO	20	27.73	554.50		Online Shop	BH-BPSO	20	16.80	336.00
		Fisher	20	13.28	265.50			Fisher	20	24.20	484.00
	Test Statistics						**Test Statistics**				
	Mann-Whitney U		55.500				Mann-Whitney U		126.00		
	Wilcoxon		265.50				Wilcoxon		336.00		
			0								
	Z		-3.910				Z		-2.083		
	Asymp Sig. (2-tailed)		0.001				Asymp Sig. (2-tailed)		0.046		
	IoT Device	BH-BPSO	20	29.23	584.50		IoT Device	BH-BPSO	20	15.38	307.50
		Fisher	20	11.78	235.50			Fisher	20	25.63	512.50
	Test Statistics						**Test Statistics**				
	Mann-Whitney U		25.50				Mann-Whitney U		97.50		
	Wilcoxon		235.50				Wilcoxon		307.50		
	Z		-4.72				Z		-2.887		
	Asymp Sig. (2-tailed)		<0.001				Asymp Sig. (2-tailed)		0.005		

In terms of cost, BH-BPSO is able to consistently minimize cost for large test suites and reduce execution time compared to existing study [30]. However, inconsistencies were recorded for small feature sizes because both algorithms can search for an optimal solution for small case studies sizes. This means that BH-BPSO is capable of achieving good quality for small and large case studies. In terms of effectiveness, it is very important to know the outcome of the research towards quality of the generated test case. In this paper, the evaluation is based on pairwise coverage and test case redundancy. According to the results, for pairwise coverage, BH-BPSO can achieve a high percentage of pairwise coverage in two case studies which are Online Shop and IoT Device.

However, the proposed algorithm recorded inconsistencies in terms of small size case because both algorithms are capable of handling such test cases generations. On the other hand, the proposed method was able to improve large case studies test cases generations. As a summary, the proposed algorithm can achieve a good result in terms of cost (time execution and size of test suite) and effectiveness (test coverage and test case redundancy). Compared to the existing studies that only focus on different evaluation criteria, this makes the proposed work more reliable on handling the multi-objective optimization problem.

Generally, based on both research questions outcome, the hybrid algorithm of BH-BPSO with parallel metaheuristics algorithm can help in solving optimization problem for test case generation. By having event horizon and stars positions update, this makes the ranking process can be started to find the optimal solutions.

6. CONCLUSION AND FUTURE WORKS

Variability and commonality of SPL represent the most common issues for industry in managing their products. The challenge was treated via the proposed hybrid of PSO and Black-Hole algorithm using CIT (BH-BPSO). In this paper, parallel metaheuristic was applied for parallel metaheuristics. From the results, BH-BPSO is able to generate test cases in large size case studies more efficiently compared to existing methods. This was achieved via exploration power of BH and binary string update of PSO. Black-hole update of particle helps PSO to rank the solution faster and obtain the optimal solution quicker. Another successful aspect of this algorithm is minimum execution time achieved for large size features. For future works, the algorithm's optimization can be further improved and extended using multiple types of parallel measurement (for example 3-wise generation).

Example:

ACKNOWLEDGEMENT

This research was supported by Ministry of Higher Education (MOHE) through Fundamental Research Grant Scheme (FRGS/1/2022/ICT01/UTHM/03/2).

REFERENCES:

[1] Hierons, R. M., Li, M., Liu, X., Parejo, J. A., Segura, S., & Yao, X. (2020). Many-objective test suite generation for software product lines. ACM Transactions on Software Engineering and Methodology, 29(1). https://doi.org/10.1145/3361146

[2] Petersen, K., Vakkalanka, S., & Kuzniarz, L. (2015). Guidelines for conducting systematic mapping studies in software engineering: An update. Information and Software Technology, 64, 1–18. https://doi.org/10.1016/j.infsof.2015.03.007

[3] Engström, E., & Runeson, P. (2011a). Software product line testing - A systematic mapping study. In Information and Software Technology (Vol. 53, Issue 1, pp. 2–13). Elsevier B.V. https://doi.org/10.1016/j.infsof.2010.05.011

[4] Meng, W.Y., Shibghatullah, A.S.B., Subaramaniam, K.,2022, "Smart Tourism Guide Application Using Location-Based Services-Go.Travel", Proceedings of International Conference on Artificial Life and Robotics, 219-228.

[5] Cavalcanti, A. , Sampaio, A., & Woodcook, J. (2010). Testing Techniques in Software Engineering (P. Borba, A. Cavalcanti, A. Sampaio, & J. Woodcook, Eds.; Vol. 6153). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-14335-9

[6] Mendonca, M., Branco, M., & Cowan, D. (2009a). S.P.L.O.T. - Software product lines online tools. Proceedings of the Conference on Object-Oriented Programming Systems, Languages, and Applications, OOPSLA, 761–762. https://doi.org/10.1145/1639950.1640002

[7] Christopher Henard, by, Cohen, M. B., & Jean-Marc Jézéquel, D.-I. (2015). Enabling Testing of Large Scale Highly Configurable Systems with Search-based Software Engineering: The Case of Model-based Software Product Lines.

[8] Henard, C., Papadakis, M., Perrouin, G., Klein, J., Heymans, P., & Traon, Y. le. (2014). Bypassing the combinatorial explosion: Using similarity to generate and prioritize t-wise test

configurations for software product lines. IEEE Transactions on Software Engineering, 40(7), 650–670. https://doi.org/10.1109/TSE.2014.2327020

[9] Dominka, S., Mandl, M., Dubner, M., & Ertl, D. (2018). Using combinatorial testing for distributed automotive features: Applying combinatorial testing for automated feature-interaction-testing. 2018 IEEE 8th Annual Computing and Communication Workshop and Conference, CCWC 2018, 2018-January, 490–495. https://doi.org/10.1109/CCWC.2018.8301632

[10] Shibghatullah, A.S., Jalil, A., Wahab, M.H.A., Soon, J.N.P., Subaramaniam, K., Eldabi, T.,2022, "Vehicle Tracking Application Based on Real Time Traffic", International Journal of Electrical and Electronic Engineering and Telecommunications, 11 (1), 67-73.

[11] Li, D. (2011). Identifying best practice by analyzing the evolution of the FISCAN MTMSIS software product line. Proceeding of the 2nd International Workshop on Product Line Approaches in Software Engineering - PLEASE '11, 40. https://doi.org/10.1145/1985484.1985495

[12] Thüm, T., Apel, S., Kästner, C., Schaefer, I., & Saake, G. (2014). A Classification and Survey of Analysis Strategies for Software Product Lines. ACM Computing Surveys, 47(1), 1–45. https://doi.org/10.1145/2580950

[13] Lochau, M., Bürdek, J., Bauregger, S., Holzer, A., von Rhein, A., & Beyer, D. (2016). On Facilitating Reuse in Multi-goal Test-Suite Generation forSoftwareProduct Lines 12.

[14] Meinicke, J., Wong, C.-P., Kästner, C., Thüm, T., & Saake, G. (2016). On essential configuration complexity: measuring interactions in highly-configurable systems. Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering, 483–494. https://doi.org/10.1145/2970276.2970322

[15] Nguyen, H. V., Kästner, C., & Nguyen, T. N. (2014). Exploring variability-aware execution for testing plugin-based web applications. Proceedings of the 36th International Conference on Software Engineering, 907–918. https://doi.org/10.1145/2568225.2568300

[16] Haini, M.S.B., Mon, C.S., Shibghatullah, A.S.B., Jalil, A.B., Subaramaniam, K.A.P., Hussin, A.A.A.,2019. An investigation into requirement of mobile app for apartment residents. International Journal on Advanced Science, Engineering and Information Technology, 9 (6), 1841-1848.

[17] Ferreira, T. N., Lima, J. A. P., Strickler, A., Kuk, J. N., Vergilio, S. R., & Pozo, A. (2017). Hyper-Heuristic Based Product Selection for Software Product Line Testing. IEEE Computational Intelligence Magazine, 12(2), 34–45. https://doi.org/10.1109/MCI.2017.2670461

[18] Al-Hajjaji, M., Thüm, T., Lochau, M., Meinicke, J., & Saake, G. (2019). Effective product-line testing using similarity-based product prioritization. Software and Systems Modeling, 18(1), 499–521. https://doi.org/10.1007/s10270-016-0569-2

[19] Monteiro, C. B. A. L., Dias, L. A. V., & da Cunha, A. M. (2014a). A case study on pairwise testing application. ITNG 2014 - Proceedings of the 11th International Conference on Information Technology: New Generations, 639–640. https://doi.org/10.1109/ITNG.2014.42

[20] Wang, Y., Sun, Y., Wu, X., Shanghai cai jing da xue, Tong ji da xue (China), Suzhou da xue, Institute of Electrical and Electronics Engineers. Beijing Section, & Institute of Electrical and Electronics Engineers. (2018). Proceedings of the 2018 IEEE International Conference on Progress in Informatics and Computing : December 14-16, 2018, Suzhou, China.

[21] Akhtar, Y., Maity, S., & Chandrasekharan, R. C. (2015). Covering Arrays of Strength Four and Software Testing (pp. 391–398). https://doi.org/10.1007/978-81-322-2452-5_26

[22] Xiang, Y., Huang, H., Member, S., Li, M., Li, S., & Yang, X. (2020). Looking For Novelty in Search-based Software Product Line Testing. In IEEE TRANSACTIONS ON SOFTWARE ENGINEERING.

[23] Rabatul Aduni Sulaiman, Dayang Norhayati Abang Jawawi, & Shahliza Abdul Halim. (2022). Classification Trends Taxonomy of Model-based Testing for Software Product Line: A Systematic Literature Review. KSII Transactions on Internet and Information Systems, 16(5). https://doi.org/10.3837/tiis.2022.05.008

[24] Lachmann, R., Beddig, S., Lity, S., Schulze, S., & Schaefer, I. (2017). Risk-based integration testing of software product lines. ACM International Conference Proceeding Series, 52–59. https://doi.org/10.1145/3023956.3023958

[25] Jung, P., Kang, S., & Lee, J. (2020). Efficient regression testing of software product lines by reducing redundant test executions. Applied

Sciences (Switzerland), 10(23), 1–21. https://doi.org/10.3390/app10238686

[26] Slowik, A., & Kwasnicka, H. (2020a). Evolutionary algorithms and their applications to engineering problems. In Neural Computing and Applications (Vol. 32, Issue 16, pp. 12363–12379). Springer. https://doi.org/10.1007/s00521-020-04832-8

[27] Huang, Z., Zhou, Y., Xia, X., & Lai, X. (2020). An improved (1+1) evolutionary algorithm for k-median clustering problem with performance guarantee. Physica A: Statistical Mechanics and Its Applications, 539. https://doi.org/10.1016/j.physa.2019.122992

[28] Weißleder, Stephan, and Hartmut Lackner. 2013. "Top-Down and Bottom-Up Approach for Model-Based Testing of Product Lines." Electronic Proceedings in Theoretical Computer Science 111(Mbt): 82–94.

[29] Köksal, and B. Tekinerdogan. 2019. "Architecture Design Approach for IoT-Based Farm Management Information Systems." Precision Agriculture 20(5): 926–58. https://doi.org/10.1007/s11119-018-09624-8.

[30] Hojjati, A., Monadi, M., Faridhosseini, A., & Mohammadi, M. (2018). Application and comparison of NSGA-II and MOPSO in multi-objective optimization of water resources systems. Journal of Hydrology and Hydromechanics, 66(3), 323–329. https://doi.org/10.2478/johh-2018-0006

[31] Jamil, M. A., Nour, M. K., Alhindi, A., Awang Abhubakar, N. S., Arif, M., & Aljabri, T. F. (2019). Towards Software Product Lines Optimization Using Evolutionary Algorithms. Procedia Computer Science, 163, 527–537. https://doi.org/10.1016/j.procs.2019.12.135

[32] Christopher Henard, Mike Papadakis, & Gilles Perrouin. (2013). PLEDGE: A Product Line Editor and Test Generation Tool.

[33] Lakshmi Prasad, M., Keerthi, M., Sai Srikar, K., & Divya, V. (2017). Generating Optimized Pairwise Test Cases by using K-Means Algorithm. www.ijatir.org

[34] Ren´, R., Bryce, R. C., & Colbourn, C. J. (2005). Test Prioritization for Pairwise Interaction Coverage.

[35] Kim, C. H. P., Khurshid, S., & Batory, D. (2012). Shared execution for efficiently testing product lines. Proceedings - International Symposium on Software Reliability Engineering, ISSRE, 221–230. https://doi.org/10.1109/ISSRE.2012.23

[36] Din, F., & Zamli, K. Z. (2019). Pairwise Test Suite Generation Using Adaptive Teaching Learning-Based Optimization Algorithm with Remedial Operator (pp. 187–195). https://doi.org/10.1007/978-3-319-99007-1_18

[37] Marijan, D., & Sen, S. (2017). Detecting and reducing redundancy in software testing for highly configurable systems. Proceedings of IEEE International Symposium on High Assurance Systems Engineering, 96–99. https://doi.org/10.1109/HASE.2017.31

[38] Benavides, D., Segura, S., & Ruiz-Cortés, A. (2010). Automated analysis of feature models 20 years later: A literature review. Information Systems, 35(6), 615–636. https://doi.org/10.1016/j.is.2010.01.001

[39] Classen, A., Cordy, M., Schobbens, P. Y., Heymans, P., Legay, A., & Raskin, J. F. (2013). Featured transition systems: Foundations for verifying variability- intensive systems and their application to LTL model checking. IEEE Transactions on Software Engineering, 39(8), 1069–1089. https://doi.org/10.1109/TSE.2012.86

[40] Segura, S., Benavides, D., & Ruiz-Cortés, A. (2011). Functional testing of feature model analysis tools: A test suite. IET Software, 5(1), 70–82. https://doi.org/10.1049/iet-sen.2009.0096

[41] Corradini, F., Fedeli, A., Fornari, F., Polini, A., & Re, B. (2021). FloWare: An Approach for IoT Support and Application Development. Lecture Notes in Business Information Processing, 421, 350–365. https://doi.org/10.1007/978-3-030-79186-5_23