SEMARAK ILMU PUBLISHING

Journal of Advanced Research in Applied Sciences and Engineering Technology

Journal homepage:
https://semarakilmu.com.my/journals/index.php/applied_sciences_eng_tech/index
ISSN: 2462-1943

A Comparative Study on Testing Optimization Techniques with Combinatorial Interaction Testing for Optimizing Software Product Line Testing

Nur Farrahin Maidin[1,*], Sa'adah Hassan[1], Salmi Baharom[1], Abu Bakar Md. Sultan[1]

[1] Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, 43400 Serdang, Selangor, Malaysia

ARTICLE INFO

Article history:
Received 12 July 2023
Received in revised form 19 September 2023
Accepted 24 June 2024
Available online 25 July 2024

Keywords:
Software Testing; Software Product Line; Testing Optimization; Combinatorial Interaction Testing

ABSTRACT

A software product line (SPL) is a combination of software products that have similarities in features and functions. These combinations usually result in many feature combinations that challenge the testing process. The explosion of the combination of features can lead to exhaustive testing. This exhaustive testing will affect the time and cost for the product to be delivered to the market. This paper aims to identify the best algorithm and interaction strength to avoid exhausting testing and reduce the time and cost of the testing process. An experiment has been conducted on the most commonly used optimization algorithms in previous studies. The optimization algorithms we explored are the Genetic Algorithm, Cuckoo Search algorithm, Ant Colony algorithm, and Particle Swarm Optimization algorithm. Each algorithm has been tested with different combinatorial interaction strengths from two to six. This paper aims to get the best meta-heuristic algorithm and the optimum number of interaction strengths for optimizing the number of configurations in the SPL testing. Results show the best optimization algorithm is the Genetic Algorithm and the optimum interaction strength is t=5. This interaction strength achieves the optimum number of features combination that is sufficient for the testing process and thus can avoid the exhaustive testing in SPL testing. By using the best optimization algorithm with the optimum number of interaction strengths, the complexity of the SPL testing process could be reduced without prejudicing the quality of the software system itself.

1. Introduction

Software Product Line (SPL) is an approach used in software product development to form a new product by reusing the existing features from the other software product that share the commonality of functions and core assets. SPL gives many benefits to the software developer, such as reduces the time and cost of the development, maintenance cost gives the best quality of software. However, employing the SPL approach contributes issues in testing process. SPL is a massive software system made up of intricate interactions between all of the system's features, which cause the number of test cases to skyrocket [1]. The number of features exponentially influences the number of valid

* *Corresponding author.*
E-mail address: gs57343@student.upm.edu.my

https://doi.org/10.37934/araset.49.1.7794

configurations; as features increase, so do the configurations. It is almost impossible to verify each setup due to the rise in the number of configurations. Therefore, an optimization approach has been introduced to optimize the number of configurations that need to be evaluated to decrease the testing effort. SPL testing optimization has been considered active research for the past five years [2]. SPL testing optimization reduces the number of configurations in the test suites by using a meta-heuristic algorithm, evolutionary algorithm, and mathematical linear.

A meta-heuristic algorithm is utilized in most studies in addressing the optimization issue in SPL testing [3-6]. A meta-heuristic algorithm is based on the nature of the animal or plant. This meta-heuristic algorithm used the combinatorial objective, which means the fitness function is in the form of any combination of functions such as an orthogonal array, mixed covering array, or uniform strength covering array. The meta-heuristic algorithm has been proposed as a solution to the non-deterministic polynomial-time hard (NP-hard) problem. Meta-heuristic is a generic algorithm framework or a black box optimizer that can be applied to almost all optimization problems [7]. It has two primary functions, exploration and exploitation. Exploration tends to search for the best solutions in surrounding areas, while exploitation will invade the new search area for the solutions.

Furthermore, the number of feature combinations for SPL testing can be decreased via combinatorial interaction testing (CIT) [8]. CIT is also known as *t*-ways testing, where the *t* indicates the interaction strength, and the value of the strength is usually between two and six. The higher the value of strength the better results will be given, but it can be more complex compared to the lower strength. CIT will select the possible pairs of configurations based on the feature model (FM) included in the test data, however testing all the combinations chosen in the test suite can be an NP-hard problem. Thus, combining the CIT and optimization algorithm can generate more accurate and efficient number of test configurations that need to be tested. Combination of meta-heuristic algorithm and combinatorial interaction testing has shown promising results in optimizing the number of configurations in the test suite to be tested in SPL [9]. Most meta-heuristic algorithms need predefine software products as the seeds or the initial population to run it. By Combining with CIT, it will define the initial population using the covering array (CA) and get the valid initial population instead of the random initial population. This combination can help increases the accuracy of the results.

This paper aims to:

i. find meta-heuristic algorithm that can give the best optimization for SPL testing
ii. find the optimum number of the interaction strength.

In this work, the meta-heuristic algorithms were combined with CIT method to optimize the number of configurations in SPL testing. The CIT presents all the valid configurations that were selected from the FM and then used as an initial population in the meta-heuristic algorithm to optimize the testing process. The selected meta-heuristic algorithms used in this study are based on the comparison analysis carried out earlier. The selected algorithms were compared to determine the best algorithm, and the most optimum interaction strength of CIT that can be implemented in the SPL testing process. An experiment has been conducted to four different meta-heuristic algorithms and using interaction strength (i.e., between 2 to 6). Each of the algorithms were run on ten different case studies for accurate results. Using the best algorithm with the optimum number of interaction strength are important aspects for optimization in SPL testing process. In which, we can optimize the number of configurations, consequently, it reduces the time and cost of testing. Besides, software testers do not have to try every single algorithm and interaction strength to get the best-optimized results during the testing process.

The remaining of this paper is structured as follows: Section 2 provides a literature review of this study. Section 3 highlights the related work and section 4 describes the experiment conducted. The finding and discussion are presented in section 5, and finally, section 6 concludes this work and highlights future work.

2. Literature Review

This section provides literature review on optimization algorithms for SPL testing and combinatorial interaction testing as the foundation for this study.

2.1 Optimization Algorithms

Meta-heuristic algorithms have been developed and implemented for optimizing the number of test cases, such as Tabu search [10], Genetic Algorithm [11], Ant Colony Optimization [12], Particle Swarm Optimization [13], Simulated Annealing [14], Harmony Search [15], Flower Pollination [16,17], Bee Algorithm [18], BAT Algorithm [19], Cuckoo Search [20], and Firefly Algorithm [21]. ~~A~~ study has been conducted to identify which meta-heuristic algorithms that mostly used in SPL testing in the recent 5 years. The results show that the Genetic Algorithm (GA) has a higher percentage at 40%, followed by Particle Swarm Optimization (PSO) at 34%. While only 10% used Ant Colony Optimization (ACO), and 7 % used Cuckoo Search (CS). Whereas it is less than 4% for the BAT, Black Hole, and Harmony Search. Thus, this study focuses on the first four optimization algorithms to compare their performance. Each of the algorithms is described in the following sub-sections. There is an existing study that uses Weighted Optimization in achieving the minimization by using the weighted GP model. This model consists of the objective function and the goal constraints [47].

2.1.1 Genetic algorithm

GA is computational modelling that stimulates biological development through Darwin's theory of genetic selection. In recent years, GA has emerged as a valuable tool for the heuristic solution for optimization problems. This algorithm is based on the principle of natural evolution. GA became a meta-heuristic search method for complex optimization problems. The ability to handle a large sample size has been an excellent advantage for optimizing SPL. Besides, GA is an ideal solution for optimization due to its ability to search in a vast and highly non-linear space. GA has a simple computational, which is decisive for improving searching operations. This algorithm begins with the initialization of the population, which represents the chromosome by generating it randomly. The binary string represents the chromosome. Then, a series of genetic operations are applied to the solution in the generated population.

The three main genetic operations in GA are selection, crossover, and mutation [49]. Figure 1 shows the pseudocode of the GA. The selection plays a part in selecting the chromosome from the population to be parents for crossover. Darwin's theory said the best chromosome could survive and create new offspring during the crossover operation [22]. The selection process could happen by using Roulette Wheel Selection [23]. This Roulette Wheel selection method will select the parents based on their fitness; if the chromosome has high fitness, the higher chances of being selected. The second operation is a crossover, the process of selecting the chromosome as parents will swap their bits. Usually, the rate of crossover used is from 0.5 to 1. It is performed by choosing random genes along the chromosome and switching them with other one-chromosome genes. After the swap, the new genes are produced known as offspring. Then, mutation operation is implemented. Mutation

operation is the process of flipping the bits of the offspring from 1 to 0 and other ways around. This process usually used 0.001 for the mutation rates. The use of GA to generate automatic data has attracted many researchers in recent years. This method has been proven as efficient and effective in generating test data. GA is also flexible and robust due to the optimization of its structured problems and does not need a specific formula to generate test data.

Algorithm 1 Genetic Algorithm Pseudocode

1. Initialize random population;
2. Evaluate population;
3. While termination criteria are not satisfied
4. Generation = Generation + 1;
5. Select a good chromosome;
6. Perform crossover with probability crossover;
7. Perform mutation with the probability of mutation;
8. Evaluate the population;

Fig. 1. GA pseudocode

2.1.2 Particle Swarm Optimization (PSO) algorithm

Particle Swarm Optimization (PSO) is an optimization algorithm design by Kennedy and Eberhart [24] is one of the earliest meta-heuristic algorithms explored by the researcher to use in the optimization problem. PSO imitates birds' behaviours by using a population in which birds or particles search for the best food sources from the information of inertia, knowledge itself, and knowledge from the swarm. The algorithm uses random positions and searching for the optimum (best) position by the given fitness function. The velocity and iteration are updated based on the previous and global best positions. PSO is an evolutionary algorithm, like GA, as it initializes with random candidates of populations, and the searching process happens by updating the generations. However, PSO does not have an evolutionary operator (e.g., crossover and mutations) and only required a few parameter settings [25]. Figure 2 shows the associated pseudo code of PSO.

Algorithm 2 Particle Swarm Optimization

1	**For each** Particle *Pi*
2	Initialize *Pi*
3	**End for**
4	**Do**
5	For each Particle *Pi*
6	Compute Fitness;
7	**If** fitness > it is personal best
8	Update current values as the new personal best;
9	**End If**
10	**End For**
11	Select the particle *P* with the best fitness value of all as the global best;
12	**For each** Particle *Pi*
13	Compute V_{id} using Equation 3;
14	Compute P_{id} using Equation 2;
15	**End For**
16	**While** {the termination criteria are not attained;}

Fig. 2. PSO pseudocode

Eq. (1) and Eq. (2) update the iteration's velocity (V) and position (S). Where *t* is the counter, C is the acceleration coefficients and R is the random number between (0, 1). The objective function will evaluate the particle, and after several iterations, the best values will return as optimal solutions.

$$V_{id}(t+1) = V_{id}(t) + C_1R_1(P_{id} - S_{id}(t)) + C_2R_2(P_{gd} - S_{id}(t)) \quad (1)$$

$$S_{id}(t + 1) = S_{id}(t) + V_{id}(t + 1) \quad (2)$$

Eq. (3) is used to update the inertia weight, where W_{mx} and W_{mi} are the minimum numbers and maximum values *W* can take, *I* is the current iterations, and I_{mx} is the total number of iterations.

$$W = W_{mx} - ((W_{mx} - W_{mi})?I_{mx})*1. \quad (3)$$

PSO algorithm has a few advantages, such as fast convergence, few parameters, and high efficiency. Further:

i. the algorithm itself does not depend on specific information, and it has strong versatility
ii. it is simple and easy to implement
iii. the ability to store and retain the best information of an individual. However, it also has disadvantages, such as poor local search ability and accuracy [26].

2.1.3 Cuckoo Search (CS) algorithm

Yang and Deb [27] introduced Cuckoo Search (CS) in 2009, an optimization algorithm based on the natural behaviours of the cuckoo bird, which has a host bird and a brood-parasitic nature. A cuckoo laid an egg in another host bird's nest. This bird selects the nest where the host bird most recently laid an egg to lay her eggs. The cuckoo egg often hatches slower than the egg of the host bird. To raise the share of the food the host bird provides, the cuckoo chick will push the other egg outside the nest after hatching. Cuckoos choose a random nest to place their egg in. Future generations will inherit the nest with the highest-quality eggs. The host egg has a chance of recognizing the cuckoo eggs (Pa [0, 1]). Each egg in the nest represents a new solution, and the number of eggs reflects the total number of solutions. The worst of the existing solutions will be replaced with the new one if it produces a superior outcome. The nest may contain multiple eggs representing a group of solutions.

CS offers more trustworthy and cost-effective solutions compared to other meta-heuristic algorithms. This algorithm provides a delicate compromise between convergence and unpredictability with fewer control parameters. CS algorithm uses three basic rules;

i. the nest is initially chosen at random
ii. only the nest that produces the highest caliber eggs will survive into succeeding generations
iii. there is a chance that the host bird will spot the egg, leave the nest, and build a new one. The CS algorithm pseudocode is as shown in Figure 3.

Algorithm 3 Cuckoo Search Algorithm	
1	Set value of an objective function, initial population, step size, range of input and maximum generation
2	Initialize the population using the initial population
3	Repeat step from 4 to 11 until the number of iterations exceeds the maximum generation or stopping criterion reached
4	Select a cuckoo randomly and generate a new solution using Le'vy flight
5	Calculate fitness value (F_i) of the solution using the objective function
6	Randomly select a nest from the available nest (say j)
7	**If** fitness value (F_i) is better than fitness value (F_j), **then**
8	Replace j with the new solution
9	**End if**
10	Abandon a fraction (P_a) of the worst solution
11	Keep track of the best solutios, rank them, and find the current best
12	List out the result

Fig. 3. CS pseudocode

Wildlife such as birds and animals naturally use the foraging trail to find food. It is a random walk because the next step depends on the current situation and the likelihood of the transition to the next place. The CS algorithm's performance has been improved by switching from ordinary random walk to Le'vy flight. The Le'vy Flight Eq. (4), has been used to create a mathematical model of this computation. Where α present the step size, and its values must always be positive. Generally, the value α is 1. The symbols of (denote the entry -wise multiplication. The random step of the Le'vy flight can be found using the Eq. (5) [28].

$$X_i^{(t+1)} = x_i(t) + \alpha \oplus \text{Le'vy}(\lambda) \quad (4)$$

$$\text{Le'vy} \sim u + t^{-\lambda}, (1 < \lambda \leq 3) \quad (5)$$

2.1.4 Ant Colony Optimization algorithm

Ant colony optimization is a meta-heuristic approach proposed in 1990 to address combinatorial optimization issues [29]. This concept is inspired by how ants move about searching for food sources. The ant spreads the pheromone throughout its route to find food sources. This pheromone trail benefits the ant in two ways: first, it indicates the path for a different ant that moves randomly, and second, it can serve as a map for the ant to return to the sources at any time. Other ants that can detect the pheromone will follow it and add pheromone to it. It increases the number of pheromones along the trail. Due to the ongoing pheromone evaporation, the other ant will select the path leaving the most extensive pheromone trail. The evaporation will make it easier for the ant to choose the regionally ideal solution. If the path is long, more pheromones will evaporate, causing the leftover pheromone to decrease. As a result, the other ants will take the shortest route based on how many pheromones are still there. The first ant to get back to its sources is the one that randomly moves and takes the quickest route. The amount of pheromone along the trail has risen due to this forward and backward movement, making it the best route for the ant to travel. Ant colony optimization refers to this action of the ants travelling backwards and forwards from the colony to the food sources. Figure 4 illustrates the procedure. Their study shows that ACO is more economical than GA. [48]

Algorithm 4 Ant Colony Optimization	
1	Begin
2	Initialize
3	While stop criteria are not satisfied
4	Do
5	Initialize ant population
6	Repeat
7	For each node
8	Do
9	Choose next node by applying updates on the transition
10	Update pheromone
11	End
12	For every node
13	Update best solution

Fig. 4. ACO pseudocode

2.2 Combinatorial Interaction Strength

Combinatorial interaction testing (CIT), commonly referred to as *t*-way or *t*-wise testing, has been utilized to reduce the interaction explosion caused by feature explosion and exhaustive testing. CIT is introduced to detect the problem from the parameter interaction due to the most failure cause by the interaction of the parameters. CIT is highly suitable for managing the complexity and feature explosion in the SPL and optimizing the number of test cases for the testing process. It focuses on creating test cases that account for every potential interaction between each system feature. CIT is a systematic strategy for sampling extensive test data domains. It is based on the finding that interactions between relatively few factors cause most problems. The definition of pairwise (or 2-wise) testing results from this. By choosing the set of all combinations, this method ensures that the test data set has every pair of possible variable values. The *t*-wise testing, which samples the input domain to cover all *t*-wise combinations, has generalized pairwise testing. This involves choosing the smallest group of items for SPL testing; where each *t*-wise features interaction occurs at least once.

The most optimum number for test case optimization can be achieved by combining the *t*-way testing technique with the meta-heuristic algorithm. Thus, the effectiveness of software testing can be increased [30]. The meta-heuristic algorithm work as a technique that will sample an optimization set of test cases from large combinatorial values that get from the interaction strength (*t*). The meta-heuristic algorithm starts the process by using the existing test cases and then implements the natural movement based on the inspired algorithm to improve the number of test cases. This movement process will run until the selected test cases have covered all the parameter interactions.

3. Related Work

In this section, related works on comparative studies on algorithms in SPL testing optimization are discussed.

A study conducted by Lopez-Herrejon *et al*., [10] analysed and compared the Non-dominated Sorting Genetic Algorithm (NSGA-II), Multi-Objective Cellular Genetic Algorithm (MOCell), Strength Pareto Evolutionary Algorithm (SPEA2), and Pareto Achieved Evolution Strategy (PAES) algorithm to identify the best algorithm. This study only concentrates on the strength of pairwise interactions. The interaction strength of the CIT has been compared to get the most optimum number of

interaction strengths that can be used for the SPL testing to optimize the testing process and avoid exhausting testing.

In addition, Carvalho *et al*., [32] also conducted a comparative study to select hyper-heuristics for multi-objective optimization problems. This study compared nine meta-heuristic algorithms and compared the performance to determine the most suitable in the real world of the multi-objective optimization problem. It shows that the meta-heuristic performs better than the single hyper-heuristic optimization algorithm. This study did not use CIT method with the meta-heuristic algorithm in optimizing the testing process.

While Panteleev *et al*., [31] conducted a comparative study of meta-heuristic algorithms that focuses on the Whale Optimization Algorithm (WOA), Grey Wolf optimizer (GWO), and Perch School Search (PSS). It analyses the performance of the three algorithms to get the most suitable optimization methods for global algorithm accuracy and convergence pattern. This study differs from our study as it did not find the optimum number of interaction strengths that can give better optimization results in SPL testing.

Up to date, it can be concluded that there is lack of comparative study on meta-heuristic algorithms with the combination of combinatorial interaction strength. Moreover, the comparison and analysis are based on the reduced number of configurations, and there is no discussion on the optimum number of configurations and coverage of the test suite.

4. The Experiment

This section discusses on the experiment that we conducted for optimizing the number of test configurations in SPL testing. The aim of this study is to determine the best optimization algorithm and the optimum number of interaction strengths for SPL testing optimization. This section describes the experimental materials and procedures carried out for this study. This study was driven by the following research questions (RQs):

i. **RQ1:** What is the meta-heuristic algorithm that can optimize the optimal number of test cases?
ii. **RQ2:** What is the best interaction strength for combinatorial interaction strength in optimizing the number of test cases?

4.1 Case Study

Internet of Things (IoT) product is complex, and it will take more time to perform testing using the usual methods. IoT is still evolving for both academia and industry [50] Thus, optimization techniques must be applied to minimize and optimize the number of test cases to be tested. Therefore, IoT of the Home system seems suitable as a case study for this experiment. The feature model (FM) of the Home Interactive System (HIS) was taken from [33] is used for this experiment. The feature model has 23 features, as shown in Figure 9. Each feature model is divided into three categories: mandatory, optional, and alternative. This is crucial to get a valid configuration from the FM. The children that inherit the mandatory features are also required in the configurations. Optional features operate differently from alternative features in that even when the parent feature of the optional feature is selected, the children can still be selected or deselected.

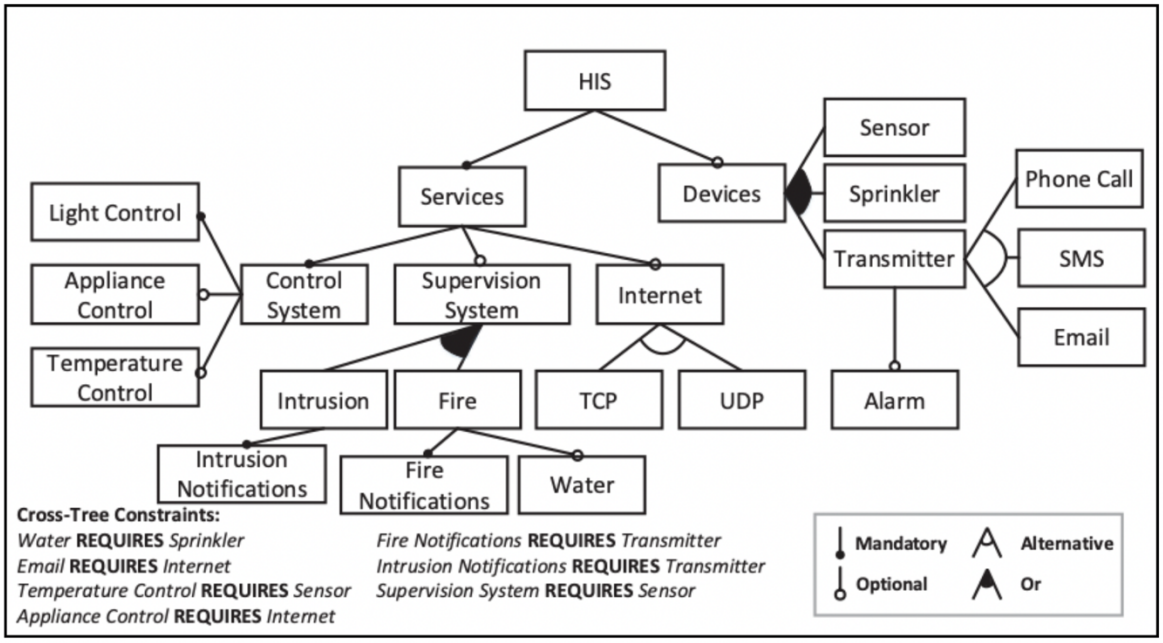

Fig. 5. Feature Model of Home Interactive System

The Home Interactive System (HIS) consists of two groups of features, as shown in Table 1, divided by 1-valued parameter and two-valued parameters. The 1-valued parameter will hold only one value (i.e., True) for the configurations, which are HIS, services, devices, light control, control system, instruction notification, and fire notification. The 2- valued parameters (i.e., True or False) are appliance control, temperature control, supervision system, internet, sensor, sprinkler, transmitter, alarm, phone call, SMS, Email, Intrusion, Fire, TCP, UDP, and water.

Table 1
The Home Interactive System configurations

System configuration	Values	
1-valued parameter (HIS, services, devices, light control, control system, instruction notification, fire notifications)	True	
2-valued parameter (appliance control, Temperature control, Supervision system, Internet, Sensor, Sprinkler, Transmitter, Alarm, Phone call, SMS, Email, Intrusion, Fire, TCP, UDP, water)	True	False

The possible number of the configurations for Home Interactive System can be generated for the testing is 1^7 x 2^{16} = 65536 configurations. Besides, the number of configurations grows with the increase of features in the FM. Looking at this number of configurations, it seems impossible to test every configuration without increasing the time to deliver the system to market and the development cost. Commonly, the cost of testing takes 50% to 60% of the overall development cost. Thus, test case optimization is needed to reduce the time and cost of the testing process.

Covering array (CA) is the mathematical calculation used for *t*-way strategies. CA consists of four parameters: N, t, p and v. This parameter represents the CA's parameters, values, and interaction strength [14]. For example, CA (2, 1, 2^{16}) represents the test suite containing 2 x 16 arrays. For this experiment, the binomial coefficient formula been used to calculate the new number of configurations based on the CA listed from the FM. Eq. (6) shows the equation of the binomial coefficient, where *n* represents the population, and *k* is the subset of *n*. This formula shows number of samples of *k* elements gained from a larger set of *n*. Table 2 shows the list of CA used for the FM and the number of configurations.

$$C(n,k) = \binom{n}{k} = \frac{n!}{(k!(n-k)!)} \quad (6)$$

Table 2
New number of configurations after implementing the t-way strategy

Covering Array (CA)	No. of configurations
CA (2, 1^7 x 2^{16})	136
CA (3, 1^7 x 2^{16})	680
CA (4, 1^7 x 2^{16})	2380
CA (5, 1^7 x 2^{16})	6188
CA (6, 1^7 x 2^{16})	12376

4.2 Experimental Setup

The four optimization algorithms were executed for optimizing the number of configurations generated earlier by using the new number of configurations using *t*-way approaches. The set of attributes and parameters listed in Tables 3, 4, 5, and 6 were used for the experiment. For reliable results, each algorithm has been calibrated to be standardized. All algorithms in this study have proven statistically significant after 20 iterations. Besides, all the experimental parameters used are based on the previous studies, in which parameter value for the GA from [8,34], PSO from [8], Cuckoo search from [8,35], and ACO parameter values from [8,36].

Table 3
Parameter of GA optimization algorithm

Parameter	Value
Size of chromosome	10
Number of populations	136, 680, 2380, 6188, 12376
Selection	0.8
Crossover	0.75
Mutation	0.03
Number of generations	1000

Table 4
Parameter of the CUCKOO optimization algorithm

Parameter	Values
Number of populations	136, 680, 2380, 6188, 12376
Maximum number of iterations	1000
Probability of alien egg	0.25
Beta	1.5
Alpha	1

Table 5
Parameter of PSO optimization algorithm

Parameter	Data size
No of population	136, 680, 2380, 6188, 12376
Inertia weight	0.9
Minimum inertia weight	0.9
Minimum inertia weight	0.4
Lower bound	1
Upper bound	20
Number of iterations	1000

Table 6
Parameter of ACO optimization algorithm

Parameter	Values
Number of populations	136, 680, 2380, 6188, 12376
Number of generations	1000
Initialization of pheromone	1.6
Pheromone persistence	0.5
Pheromone amount	0.01
Initial pheromone	0.4

Pareto optimal set has been used to find the optimum number of configurations that need to be tested in recent studies [37-39]. There are two objectives for this experiment; coverage, and test suite size and both are equally important. The set of solutions is considered non-dominated if it non-dominates another. The Pareto optimal set is a non-dominated solution that non-dominates each other in the search space. The Pareto optimal set will find the optimal solution for both objectives and give the best optimization results.

Table 7 demonstrates how each method maximized the number of configurations from the case study, with interaction strength ranging from two to six. The quantity of configurations that must be tested during the testing process is decreased. As this experiment uses the Pareto optimum set to obtain the best solutions, this number has achieved the optimum number to be tested. The ACO algorithm has the most significant number of optimum configurations for interaction strength, t=2. In contrast, the CS algorithm has the fewest configurations for an interaction strength t=3, and the same is true for an interaction strength t=6. Out of the five interaction strengths, PSO is the most optimal.

Table 7
Number of configurations after implementing optimization algorithm

Interaction strength	No. of configurations	GA	PSO	ACO	Cuckoo
2	136	80	94	77	91
3	680	480	484	465	430
4	2380	1750	1622	1090	1198
5	6188	4298	3908	4138	5099
6	12376	8516	8207	8864	8154

Table 8 compares the optimization for each algorithm and level of interaction by listing the number of reductions. Another eight feature models were added which are, WS, EC, James [40], Smart Mobile [41], Car audio system [42], Vendor Machine [43], Gold Kid Bus [44], and Snake FOP Games [45]. Since the experiment used ~~a~~ large FM, it is essential to remember that a smaller FM can achieve 90–100% coverage because the number of reductions is also minimal. In this experiment we did not compare the performance in terms of time as the evaluation can be unfair due to the various fitness evaluations for each strategy.

Table 8
Number of configurations for all FM

FM	t-wise	Algorithm			
		GA	PSO	ACO	CUCKOO
HIS	2	80	94	77	91
	3	480	484	465	430
	4	1750	1622	1090	1198
	5	4298	3908	4138	5099
	6	8516	8207	8864	8154
WS	2	219	171	198	201
	3	1258	1329	1321	1493
	4	7093	6985	6319	6715
	5	25131	23413	20397	24993
	6	41736	33976	30512	40179
EC	2	165	107	139	171
	3	813	532	798	625
	4	4129	3725	4469	3975
	5	12461	10710	11371	8293
	6	19532	21743	35915	20431
James	2	65	52	41	57
	3	262	205	195	239
	4	631	429	315	503
	5	1091	615	964	1153
	6	942	815	1173	713
Smart Mobile	2	43	39	37	51
	3	139	113	127	141
	4	299	235	319	326
	5	405	392	384	422
	6	260	239	241	219
Car Audio system	2	99	79	82	75
	3	433	385	379	321
	4	1217	1041	997	1184
	5	2842	2099	1817	2537
	6	2519	3078	2456	4193
Vendor Machine	2	139	141	129	97
	3	784	761	607	523
	4	2719	2583	2713	2663
	5	8348	8098	7593	7919
	6	9077	9765	15342	13219
Gold Kid Bus	2	36	32	41	37
	3	98	87	113	95
	4	174	199	197	180
	5	213	226	239	237
	6	143	159	268	170
Snake FOP Game	2	105	93	82	95
	3	530	421	524	495
	4	1721	1683	1592	1565
	5	4017	3918	3652	3773
	6	4103	3915	3817	3972

The information in Table 9 shows the coverage matric to compare the coverage of the configuration for the interaction strength and the algorithms. The necessary test suite metrics are the high number of coverage as well as the lower number of configurations. It is crucial to achieving a higher value for this metric because it is expected to cover all expected configurations.

Table 9
Number of coverages for the optimization

FM	t-wise	Algorithm			
		GA	PSO	ACO	CUCKOO
HIS	2	0.59	0.69	0.57	0.67
	3	0.71	0.71	0.68	0.63
	4	0.74	0.68	0.46	0.50
	5	0.69	0.63	0.67	0.82
	6	0.69	0.66	0.71	0.66
WS	2	0.94	0.74	0.85	0.87
	3	0.81	0.86	0.85	0.96
	4	0.96	0.95	0.86	0.91
	5	0.95	0.88	0.77	0.94
	6	0.55	0.45	0.40	0.53
EC	2	0.86	0.56	0.73	0.90
	3	0.71	0.46	0.70	0.54
	4	0.85	0.76	0.92	0.82
	5	0.80	0.69	0.73	0.53
	6	0.50	0.56	0.92	0.52
James	2	0.83	0.65	0.52	0.73
	3	0.92	0.72	0.68	0.84
	4	0.88	0.60	0.44	0.70
	5	0.85	0.48	0.75	0.89
	6	0.55	0.47	0.68	0.41
Smart Mobile	2	0.78	0.71	0.67	0.92
	3	0.84	0.68	0.77	0.85
	4	0.91	0.71	0.97	0.98
	5	0.88	0.85	0.83	0.91
	6	0.56	0.52	0.52	0.47
Car Audio system	2	0.94	0.75	0.78	0.71
	3	0.95	0.85	0.83	0.71
	4	0.89	0.76	0.73	0.87
	5	0.94	0.69	0.60	0.84
	6	0.50	0.61	0.49	0.84
Vendor Machine	2	0.91	0.92	0.84	0.63
	3	0.96	0.93	0.74	0.64
	4	0.89	0.84	0.88	0.92
	5	0.97	0.94	0.88	0.92
	6	0.49	0.53	0.83	0.71
Gold Kid Bus	2	0.80	0.71	0.91	0.82
	3	0.81	0.73	0.94	0.80
	4	0.83	0.95	0.94	0.86
	5	0.85	0.90	0.95	0.67
	6	0.51	0.57	0.96	0.61
Snake FOP Game	2	0.59	0.69	0.57	0.67
	3	0.71	0.70	0.68	0.63
	4	0.74	0.68	0.46	0.50
	5	0.69	0.63	0.67	0.82
	6	0.69	0.66	0.71	0.66

In this experiment, we used the total coverage of the configuration as if the main goal for the study was only in the testing, with the focus being the subset of all possible configurations as the main target. This metric was calculated by using Eq. (7) as below:

$$cc/c \quad (7)$$

cc: No. of configuration covered by *t*
c: No. of configuration

4.3 Threats to Validity

There are some efforts taken to reduce such threats. Firstly, the meta-heuristic algorithms involved in this study are based on the analysis of research publications gathered from the preceding five years. This portion of the primarily employed algorithm comes from our findings, which may not agree with those of other researchers.

Secondly, the parameters used in this study are referring to earlier research work by different researchers. We relied on the parameter values employed in most prior studies that addressed issues from related scientific literature. However, changes in the values might affect the experiment's outcome, which does not apply to this study. Retuning the parameters may therefore be helpful to get the best results.

Thirdly, is about the choice of the case study for the feature model. The outcomes of various case studies may vary. Since IoT is a practical topic, we chose a case study with actual industry problems for this study. It is one of the SPL products with various properties that interact with one another and is appropriate for this experiment. We could not achieve the high number of characteristics included in this study due to the combination of significant aspects in this case study. A smaller case study will be used for the experiment to compare the results for future work.

The final threat to validity comes from the study's statistical analysis. The best values from the experiment were used in the study's statistical analysis, not the mean values. As a result, it is possible that the meta-heuristic algorithm's random generation led to the best result by accident, which could affect the conclusion of this study.

5. Discussion

The Friedman Test [8] has been applied in this study to check whether the differences between all the~~se~~ four algorithms are statistically significant or just a matter of chance. By using a confidence level of 95% (*p*-value under 0.05) as shown in Table 10. According to Friedman's null hypothesis, all the strategies are equivalent, so a rejection of these strategies shows differences in the performance of all the strategies. The null hypothesis is rejected if the Friedman statistic exceeds the critical value. It can be concluded that the results obtained are significant, and there are differences between each of the algorithms and CIT. It is statistically proven that these experimental results are valid.

Table 10
Friedman test for Table 8

Friedman Test	Conclusion
Degree of freedom = 3, α = 0.05 Critical value = 0.56415 Friedman statistic (χ^2) = 2.04	2.04 > critical value, reject H_o

Based on the findings, it helps to answer **RQ1**. It can be concluded that the best meta-heuristic algorithm is the GA as from the nine FMs; seven of them show the best optimization results by using the GA. Another two FMs show best results for Ant Colony Optimization and Cuckoo Search. The two FMs have number of features 11 and 10 compared to the other seven, which are 23, 24, 22, 14, 21,

18 and 21. Thus, we can conclude that GA has better performance in handling the bigger FM compared to other algorithms. The average of the performance for each interaction strength also showed that GA has better performance with interaction strength of 2 to 6.

In term of coverage, GA shows the higher coverage in most of the interaction strength for every single FM. This coverage has reached the ideal number, by adding the configuration in the test suite; it would not increase the coverage. It shows that the coverage of the configuration for the case study will be lower when the reduction is bigger because the number of reductions affected the coverage. As mentioned earlier, the metric should be higher coverage with low number of configurations. It is not appropriate to evaluate based on the total reduction only as the coverage play an important role to ensure the quality of the system that we test. For instance, if the reduction rate is high but the coverage is low, it may leave out critical configurations that need to be evaluated, in which could affect the product's quality.

Concerning **RQ2**, the combinatorial interaction strength t = 5 results in higher coverage rates than other strengths. It has been established that stronger interactions lead to better optimization outcomes [46]. Based on the experiment results, six of the FM show high coverage in the interaction strength t = 5, while one for interaction strength t = 4, and another two for interaction strength t = 3. For the interaction strength t = 6, the coverage is the lowest among all interaction strengths as the reduction of configuration is quite a big number of it. Most of the results show that the coverage is only at 0.5, which means half of the configurations. This is because of the explosion of the configuration number in the interaction strength t = 6, the optimization reduces almost half of the configuration. The average coverages between 5 and 6 interaction strengths reveal huge variations. Even though the number of configurations in the interaction strength t= 6 is higher than the interaction strength t= 5, it is crucial to focus on the coverage as insufficient test coverage will result in poor of system performance when delivered to the market. The higher the number of interaction strengths will give the better performance, but it is enough to do testing in the optimal strength to reduce the number of the test suite and the time to deliver the product to market as we have achieved the higher optimization coverage. Thus, the t-way = 5 gives an optimal number of the test suite and sufficient coverage for the configuration that needs to be tested.

6. Conclusion and Future Work

One of the main problems of SPL testing is the explosion combination of features number to be tested. This paper has identified four optimization algorithms that have mostly been used for optimizing the number of test suites for SPL testing. An experiment has been conducted to compare the performance of the four algorithms with different interaction strengths, t. The result has been evaluated based on the number of test case reductions and coverage. Based on the findings, it can be concluded that we cannot only consider the reduction rate; as the higher the reduction, the lower the test case coverage. The coverage should also be in consideration to get the best number of test cases for the testing process. By using the optimal Pareto set to get the optimum number of coverage and from the results it can be concluded that the Genetic Algorithm has the best performance compared to the other three algorithms, with interaction strength t = 5 for the optimum coverage.

This study suggests the best meta-heuristic algorithm that can be used in SPL testing optimization and the best interaction strength that satisfies the reduction number of the configuration and maintains high coverage for the configuration. By applying the interaction strength, t = 5, the optimum number of configurations with high coverage can be achieved. Therefore, it can help software testers use the most appropriate algorithm and combinatorial interaction strength based

on the optimum feature coverage for the SPL testing process. Besides, it reduces the time and cost of testing.

For future work, this study will be taking data redundancy into consideration. Reducing test redundancy also can significantly reduce the time and cost of testing while still ensuring software quality and reliability [51]. This redundancy reduction can optimize the test configuration more as it will ensure test configuration meet with the requirement without having unnecessary repetition.

Acknowledgement
The Ministry of Higher Education supported this work under the Fundamental Research Grant Scheme (FRGS/1/2020/ICT01/UPM/02/1).

References

[1] Asadi, Mohsen, Samaneh Soltani, Dragan Gašević, and Marek Hatala. "The effects of visualization and interaction techniques on feature model configuration." *Empirical Software Engineering* 21 (2016): 1706-1743. https://doi.org/10.1007/s10664-014-9353-5

[2] Kumari, A. Charan. "Feature selection optimization in SPL using genetic algorithm." *Procedia computer science* 132 (2018): 1477-1486. https://doi.org/10.1016/j.procs.2018.05.082

[3] Ferreira, Thiago N., Jackson A. Prado Lima, Andrei Strickler, Josiel N. Kuk, Silvia R. Vergilio, and Aurora Pozo. "Hyper-heuristic based product selection for software product line testing." *IEEE Computational Intelligence Magazine* 12, no. 2 (2017): 34-45. https://doi.org/10.1109/MCI.2017.2670461

[4] Mateen, Ahmed, Marriam Nazir, and Salman Afsar Awan. "Optimization of test case generation using genetic algorithm (GA)." *arXiv preprint arXiv:1612.08813* (2016). https://doi.org/10.5120/ijca2016911703

[5] Alsariera, Yazan A., Mazlina A. Majid, and Kamal Z. Zamli. "SPLBA: An interaction strategy for testing software product lines using the Bat-inspired algorithm." In *2015 4th international conference on software engineering and computer systems (ICSECS)*, pp. 148-153. IEEE, 2015. https://doi.org/10.1109/ICSECS.2015.7333100

[6] Shingadiya, Chetan J. "Genetic algorithm for test suite optimization: an experimental investigation of different selection methods." *Turkish Journal of Computer and Mathematics Education (TURCOMAT)* 12, no. 3 (2021): 3778-3787. https://doi.org/10.17762/turcomat.v12i3.1661

[7] Sangaiah, Arun Kumar, Zhiyong Zhang, and Michael Sheng, eds. *Computational intelligence for multimedia big data on the cloud with engineering applications*. Academic Press, 2018.

[8] Lopez-Herrejon, Roberto E., Stefan Fischer, Rudolf Ramler, and Alexander Egyed. "A first systematic mapping study on combinatorial interaction testing for software product lines." In *2015 IEEE Eighth International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, pp. 1-10. IEEE, 2015. https://doi.org/10.1109/ICSTW.2015.7107435

[9] Zamli, Kamal Z., Basem Y. Alkazemi, and Graham Kendall. "A tabu search hyper-heuristic strategy for t-way test suite generation." *Applied Soft Computing* 44 (2016): 57-74. https://doi.org/10.1016/j.asoc.2016.03.021

[10] Díaz, Eugenia, Javier Tuya, and Raquel Blanco. "Automated software testing using a metaheuristic technique based on tabu search." In *18th IEEE International Conference on Automated Software Engineering, 2003. Proceedings.*, pp. 310-313. IEEE, 2003.

[11] Lopez-Herrejon, Roberto Erick, Javier Javier Ferrer, Francisco Chicano, Evelyn Nicole Haslinger, Alexander Egyed, and Enrique Alba. "A parallel evolutionary algorithm for prioritized pairwise testing of software product lines." In *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, pp. 1255-1262. 2014. https://doi.org/10.1145/2576768.2598305

[12] Biswas, Sumon, M. Shamim Kaiser, and S. A. Mamun. "Applying ant colony optimization in software testing to generate prioritized optimal path and test data." In *2015 International Conference on Electrical Engineering and Information Communication Technology (ICEEICT)*, pp. 1-6. IEEE, 2015. https://doi.org/10.1109/ICEEICT.2015.7307500

[13] Jianqi, Shi, Huang Yanhong, Li Ang, and Cai Fangda. "An optimal solution for software testing case generation based on particle swarm optimization." *Open Physics* 16, no. 1 (2018): 355-363. https://doi.org/10.1515/phys-2018-0048

[14] Wang, Kun, Yichen Wang, and Liyan Zhang. "Software testing method based on improved simulated annealing algorithm." In *2014 10th International Conference on Reliability, Maintainability and Safety (ICRMS)*, pp. 418-421. IEEE, 2014. https://doi.org/10.1109/ICRMS.2014.7107215

[15] Alazzawi, Ammar K., Helmi Md Rais, Shuib Basri, and Yazan A. Alsariera. "PhABC: A hybrid artificial bee colony strategy for pairwise test suite generation with constraints support." In *2019 IEEE Student Conference on Research and Development (SCOReD)*, pp. 106-111. IEEE, 2019. https://doi.org/10.1109/SCORED.2019.8896324

[16] Nasser, Abdullah B., Kamal Z. Zamli, AbdulRahman A. Alsewari, and Bestoun S. Ahmed. "Hybrid flower pollination algorithm strategies for t-way test suite generation." *PloS one* 13, no. 5 (2018): e0195187. https://doi.org/10.1371/journal.pone.0195187

[17] Ramli, Nuraminah, Rozmie Razif Othman, Zahereel Ishwar Abdul Khalib, and Muzammil Jusoh. "A review on recent t-way combinatorial testing strategy." In *MATEC Web of Conferences*, vol. 140, p. 01016. EDP Sciences, 2017. https://doi.org/10.1051/matecconf/201714001016

[18] Sharma, Sangeeta, and Pawan Bhambu. "Artificial bee colony algorithm: A survey." *International Journal of Computer Applications* 149, no. 4 (2016): 11-19. https://doi.org/10.5120/ijca2016911384

[19] Hasan, Luma Salal. "Artificial Bee Colony Algorithm and Bat Algorithm for Solving Travel Salesman Problem." *Webology* 19, no. 1 (2022): 4185-4193. https://doi.org/10.14704/WEB/V19I1/WEB19276

[20] Rakhshani, Hojjat, and Amin Rahati. "Snap-drift cuckoo search: A novel cuckoo search optimization algorithm." *Applied Soft Computing* 52 (2017): 771-794. https://doi.org/10.1016/j.asoc.2016.09.048

[21] Pandey, Abhishek, and Soumya Banerjee. "Test suite optimization using chaotic firefly algorithm in software testing." In *Research Anthology on Recent Trends, Tools, and Implications of Computer Programming*, pp. 722-739. IGI Global, 2021. https://doi.org/10.4018/978-1-7998-3016-0.ch032

[22] Swathi, Baswaraju, and Harshvardhan Tiwari. "Integrated pairwise testing based genetic algorithm for test optimization." *International Journal of Advanced Computer Science and Applications* 12, no. 4 (2021). https://doi.org/10.14569/IJACSA.2021.0120419

[23] Guo, Jianmei, Jules White, Guangxin Wang, Jian Li, and Yinglin Wang. "A genetic algorithm for optimized feature selection with resource constraints in software product lines." *Journal of Systems and Software* 84, no. 12 (2011): 2208-2221. https://doi.org/10.1016/j.jss.2011.06.026

[24] Chen Xiang, Gu Qing, Wang Ziyuan, and Chen Daoxu. "Framework of Particle Swarm Optimization Based Pairwise Testing." *Journal of Software* 22, no. 12: 2879-2893. https://doi.org/10.3724/SP.J.1001.2011.03973

[25] Afzal, Uzma, Tariq Mahmood, Ayaz H. Khan, Sadeeq Jan, Raihan Ur Rasool, Ali Mustafa Qamar, and Rehan Ullah Khan. "Feature selection optimization in software product lines." *IEEE Access* 8 (2020): 160231-160250. https://doi.org/10.1109/ACCESS.2020.3020795

[26] Liu, Congcong. "Research on Software Test Data Generation based on Particle Swarm Optimization Algorithm." In *2021 5th International Conference on Trends in Electronics and Informatics (ICOEI)*, pp. 1375-1378. IEEE, 2021. https://doi.org/10.1109/ICOEI51242.2021.9452870

[27] Yang, Xin-She, and Xingshi He. "Bat algorithm: literature review and applications." *International Journal of Bio-inspired computation* 5, no. 3 (2013): 141-149. https://doi.org/10.1504/IJBIC.2013.055093

[28] Sharma, Sanjiv, S. A. M. Rizvi, and Vineet Sharma. "A framework for optimization of software test cases generation using cuckoo search algorithm." In *2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, pp. 282-286. IEEE, 2019. https://doi.org/10.1109/CONFLUENCE.2019.8776898

[29] Mohan, B. Chandra, and R. Baskaran. "A survey: Ant Colony Optimization based recent research and implementation on several engineering domain." *Expert Systems with Applications* 39, no. 4 (2012): 4618-4627. https://doi.org/10.1016/j.eswa.2011.09.076

[30] Ramgouda, P., and V. Chandraprakash. "Constraints handling in combinatorial interaction testing using multi-objective crow search and fruitfly optimization." *Soft Computing* 23 (2019): 2713-2726. https://doi.org/10.1007/s00500-019-03795-w

[31] Panteleev, A. V., I. A. Belyakov, and A. A. Kolessa. "Comparative analysis of optimization strategies by software complex "Metaheuristic nature-inspired methods of global optimization"." In *Journal of Physics: Conference Series*, vol. 2308, no. 1, p. 012002. IOP Publishing, 2022. https://doi.org/10.1088/1742-6596/2308/1/012002

[32] de Carvalho, Vinicius Renan, Ender Özcan, and Jaime Simão Sichman. "Comparative analysis of selection hyper-heuristics for real-world multi-objective optimization problems." *Applied Sciences* 11, no. 19 (2021): 9153. https://doi.org/10.3390/app11199153

[33] Grindal, Mats, Jeff Offutt, and Sten F. Andler. "Combination testing strategies: a survey." *Software Testing, Verification and Reliability* 15, no. 3 (2005): 167-199. https://doi.org/10.1002/stvr.319

[34] Shiba, Toshiaki, Tatsuhiro Tsuchiya, and Tohru Kikuno. "Using artificial life techniques to generate test cases for combinatorial testing." In *Proceedings of the 28th Annual International Computer Software and Applications Conference, 2004. COMPSAC 2004.*, pp. 72-77. IEEE, 2004.

[35] Miyake, Yuki, Wataru Kumagai, Kenichi Tamura, and Keiichiro Yasuda. "Search point ranking-based adaptive cuckoo search." *IEEJ Transactions on Electrical and Electronic Engineering* 13, no. 7 (2018): 1075-1076. https://doi.org/10.1002/tee.22667

[36] Mueller, Carsten. "Multi-objective optimization of software architectures using ant colony optimization." *Lecture notes on software engineering* 2, no. 4 (2014): 371. https://doi.org/10.7763/LNSE.2014.V2.152
[37] Jamil, Muhammad Abid, Mohamed K. Nour, Ahmad Alhindi, Normi Sham Awang Abhubakar, Muhammad Arif, and Tareq Fahad Aljabri. "Towards software product lines optimization using evolutionary algorithms." *Procedia Computer Science* 163 (2019): 527-537. https://doi.org/10.1016/j.procs.2019.12.135
[38] Ferreira, Thiago Nascimento, Silvia Regina Vergilio, and Jerffeson Teixeira de Souza. "Incorporating user preferences in search-based software engineering: A systematic mapping study." *Information and Software Technology* 90 (2017): 55-69. https://doi.org/10.1016/j.infsof.2017.05.003
[39] Sheng, Wanxing, Yongmei Liu, Xiaoli Meng, and Tianshu Zhang. "An Improved Strength Pareto Evolutionary Algorithm 2 with application to the optimization of distributed generations." *Computers & Mathematics with Applications* 64, no. 5 (2012): 944-955. https://doi.org/10.1016/j.camwa.2012.01.063
[40] Qian, Yekan, Cheng Zhang, and Futian Wang. "Selecting products for high-strength t-wise testing of software product line by multi-objective method." In *2018 IEEE International Conference on Progress in Informatics and Computing (PIC)*, pp. 370-378. IEEE, 2018. https://doi.org/10.1109/PIC.2018.8706270
[41] Al-Hajjaji, Mustafa, Jacob Krüger, Sandro Schulze, Thomas Leich, and Gunter Saake. "Efficient product-line testing using cluster-based product prioritization." In *2017 IEEE/ACM 12th International Workshop on Automation of Software Testing (AST)*, pp. 16-22. IEEE, 2017. https://doi.org/10.1109/AST.2017.7
[42] Matnei Filho, Rui A., and Silvia R. Vergilio. "A multi-objective test data generation approach for mutation testing of feature models." *Journal of Software Engineering Research and Development* 4, no. 1 (2016): 1-29. https://doi.org/10.1186/s40411-016-0030-9
[43] Tuglular, Tugkan, Mutlu Beyazıt, and Dilek Öztürk. "Featured event sequence graphs for model-based incremental testing of software product lines." In *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)*, vol. 1, pp. 197-202. IEEE, 2019. https://doi.org/10.1109/COMPSAC.2019.00035
[44] Tuglular, Tugkan, and Sercan Şensülün. "SPL-AT Gherkin: A Gherkin Extension for Feature Oriented Testing of Software Product Lines." In *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)*, vol. 2, pp. 344-349. IEEE, 2019. https://doi.org/10.1109/COMPSAC.2019.10230
[45] CHEN, Xiang, Ji-Hong CHEN, Xiao-Lin JU, and Qing GU. "Survey of Test Case Prioritization Techniques for Regression Testing."
[46] Sahid, Mohd Zanes, Abu Bakar Md Sultan, Abdul Azim Abdul Ghani, and Salmi Baharom. "Combinatorial Interaction Testing of Software Product Lines: A Mapping Study." *J. Comput. Sci.* 12, no. 8 (2016): 379-398. https://doi.org/10.3844/jcssp.2016.379.398
[47] Halim, Khadijah Mohd, and Wan Khadijah. "Hospital Bed Allocation using Three-Stage Weighted Optimization Method for Government Hospital in Pulau Pinang." *Journal of Advanced Research in Applied Sciences and Engineering Technology* 31, no. 1 (2023): 90-98. https://doi.org/10.37934/araset.31.1.9098
[48] Maddina, Suresh Babu, R. Thirunavukkarasu, and N. Karthik. "Optimization of Energy Storage Unit Size and Location in a Radial Distribution Network to Minimize Power Loss Using Firefly Algorithm." *Journal of Advanced Research in Applied Sciences and Engineering Technology* 31, no. 3 (2023): 25-42. https://doi.org/10.37934/araset.31.3.2542
[49] Majid, Hanafi, Syahid Anuar, and Noor Hafizah Hassan. "TPOT-MTR: A Multiple Target Regression Based on Genetic Algorithm of Automated Machine Learning Systems." *Journal of Advanced Research in Applied Sciences and Engineering Technology* 30, no. 3 (2023): 104-126. https://doi.org/10.37934/araset.30.3.104126
[50] Omar Alhazmi. "A Survivable Internet of Things Scheme" *Journal of Advance Research in Computing and Applications* 13, no. 1 (2018): 19-26
[51] Rahman, Mizanur, Kamal Z. Zamli, Md Abdul Kader, Roslina Mohd Sidek, and Fakhrud Din. "Comprehensive Review on the State-of-the-arts and Solutions to the Test Redundancy Reduction Problem with Taxonomy." *Journal of Advanced Research in Applied Sciences and Engineering Technology* 35, no. 1 (2024): 62-87. https://doi.org/10.37934/araset.34.3.6287