JSCDM

JOURNAL OF SOFT COMPUTING AND DATA MINING
e-ISSN: 2716-621X

Vol. 7 No. 1 (2026) 127-140
https://publisher.uthm.edu.my/ojs/index.php/jscdm

A Variable Strength Interaction Test Suites Generation Strategy using Sewing Training Optimization

Murad Muhammad Hasan Salih Al-Walidi[1]*, Ahmad Ashraf Abdul Halim[1], Rozmie Razif Othman[1], Mohd Zamri Zahir Ahmad[1], Kentaro Go[2], Muhammad Aiman Mohd Asyraf[1]

[1] *Centre of Excellence for Advanced Computing (AdvComp), Universiti Malaysia Perlis, MALAYSIA*
[2] *Department of Computer Science and Engineering, University of Yamanashi, JAPAN*

*Corresponding Author: muradmuhammad@studentmail.unimap.edu.my
DOI: https://doi.org/10.30880/jscdm.2026.07.01.010

Article Info

Received: 30 October 2025
Accepted: 23 February 2026
Available online: 10 April 2026

Keywords

Software testing, variable strength interaction, T-way testing, metaheuristic optimization, combinatorial testing, Sewing Training Based Optimization (STBO)

Abstract

This paper reports on a study in which a new model for the development of variable strength (VS) interaction test suits, identified as Sewing Training Based Optimization (STBO). This approach is based on the principle of how tailor apprentices learn through a stage of imitation, a stage of practice, and a stage of evaluation, which together bring about a balance between wide exploration and focused search. To evaluate its performance in terms of test suite size reduction and scalability, STBO was applied to commonly used base models. The results indicate that STBO produced 26 optimal solutions, outperforming nearly 90% of existing methods. Consistent superiority over traditional algorithms was observed, while performance remained comparable to that of more complex metaheuristic approaches. In addition, a real-time software system case study was conducted, in which STBO demonstrated effective fault detection capability, thereby indicating that it is a reliable and adaptive strategy for software testing.

1. Introduction

Software testing is a key element in the reliability, efficiency and correctness of modern software systems [1]. As these systems grow in complexity a great increase in the number of input combinations is observed which in turn causes a combinatorial explosion. Also, it is indicated that most software defects are a result of the interaction of only a few input variables [2] hence it is not practical to test all of them out [3]. Past methods of reducing tests like equivalence partitioning and boundary value analysis made great test suits in making the input sets smaller but by only presenting representative values [4]. In previous studies, presenting combinations of many input variables that interact with each other was not as expected. In order to fill that gap Combinatorial Interaction Testing (CIT) which includes t-way testing has grown in popularity. The way T-way testing works is to check that any interactions between any set of the variables are included at least once. This in turn makes it easier to identify the bugs and also greatly reduces the size of the test suite [5].

T-way testing does indeed work but in the real world it introduces other issues. It is observed that variables which are put forward often do not pan out and that the strength of interactions between parameters is variable which in turn requires termed variable strength testing. Also, there is the issue of creating test suites which is an

NP hard issue that requires approaches which reduce the size of the test suite, which at the same time will follow the rules and also find many faults [6]. To handle this complexity that is being observed is an increase in the use of metaheuristic algorithms. These algorithms allow for large and irregular solution spaces to be explored quickly and easily [7]. In the area of testing that is being seen thus far there are Simulated Annealing, Genetic Algorithms, Ant Colony Optimization and Particle Swarm Optimization which have reported very good results in terms of scale up and constraint handling [4]. There is also a call out for new methods which do a better job of balancing exploration, exploitation and adaptation in the process of putting together tests.

STBO brings a new perspective to the issue at hand. In STBO there is a structure which presents simple skills as the basis for complex knowledge. This is from the systematized training of tailoring apprentices. Also, this is played out in the optimization process which improves candidate solutions over time [8]. STBO has performed well in many areas of optimization such as mechanical design, scheduling and resource allocation. It is known for putting out scalable high-quality solutions. It offers an organized yet flexible approach which balances practice with assessment. This makes it capable of solving complex issues of generating test suites. This study shows that STBO is adapted to the software testing field in the context of variable-strength, constraint aware t-way test suite development. STBO makes test optimization easier by approaching a structured training process. This in turn allows large search spaces to be navigated easily, faulty combinations to be eliminated, redundancy to be reduced, and compact yet very effective test suites to be built [9]. Its proven growth ability also makes it very likely that it will do well with the increasing complexity of present-day software systems.

This study reports on the application of STBO to software testing, extending beyond studies of engineering issues which have been tried out previously. Key issues of scalability and constraint management, which usually make combinatorial testing a tough reality in practice, are addressed by introducing STBO to put forth variable strength, constraint aware t-way test suites. An in-depth experimental study is also presented and reported to demonstrate the ability of STBO to put forth compact, efficient and fault tolerant test suites, while performing very well in competition with state-of-the-art metaheuristic approaches.

The present study reports as follows. In Section 2 there are terms of evaluating search algorithms for variable strength covering arrays in t-way testing is examined. Section 3 presents strategies that are put forth with regard to the STBO technique and it can create test packages. Section 4 reports on the experiments that were conducted and their results. Section 5 presents the statistical analysis which supports the data. Lastly, Section 6 draws conclusions and puts forth future research ideas.

2. Related Work

2.1 Variable Strength Covering Array

The Covering Array (CA) has been put forth as a standard in combinatorial testing for its ability to systemically cover all t-way interactions among input parameters. it covers all the bases yet at the same time it produces extra test cases when the interaction strength is high, which in turn causes great computing issues [10]. Also, in conventional CA the same interaction strength is used for all parameters, which does not always represent the importance of different elements in real world systems [11]. To that end, there has been the introduction of the Variable Strength Covering Array (VSCA) [12]. VSCA deals with tests differently as it allows for different interaction levels to be used within subsets of parameters, thus achieving a balance between fault detection and test suite size [13].

Mathematically, VSCA is expressed as (1):

$$S = VSCA(N, t, C, R) \tag{1}$$

where N is the test suite size, t the dominant interaction strength, and C the parameter configuration values, expressed as $v_1^{(p_1)}, v_2^{(p_2)}, \ldots, v_n^{(p_n)}$. The set R defines subsets of parameters with different interaction strengths, which lets you create mixed-strength arrays that are perfect for your system. Metaheuristic algorithms have been widely used to reduce suite size while keeping coverage because making an optimal VSCA is an NP-hard problem [14].

2.2 Computational Approaches

Computers have been the base for the development of t-way test suites. The use of algebraic models and heuristic rules in the design of covering arrays is observed, which is a very methodical approach [15]. The main advantage of them is that it is a very predictable and deterministic process which in turn means the test sets which come out of it may be reproduced and checked. But as system elements grow in size and complexity these methods have issue with scale which in turn produces large test sets that are expensive to run [16].

Some of the most used methods are TVG [17], Density [18], SCIPOG-VS [19], IPOG [20], GVS [21] and DA-FO [22]. While they do a good job with small to medium scale issues, they fall short in large scale, variable and complex constraint settings which in turn limits their application in practical testing.

2.3 Metaheuristic Approaches

Metaheuristic algorithms are seen as adaptive search-based methods which have extensive research as a solution to scale. They start out with a random solution which is improved step by step via guided exploration and exploitation [23]. Analysed is their strength in that they are able to put forth nearly perfect solutions quickly in large scale testing environments which also have many constraints [24]. Also, it is their ability to learn and adapt that allows them to navigate high dimensional search spaces [25]. Metaheuristic methods have been successful in many fields, such as Ant Colony Optimisation (ACO), which is a type of combinatorial optimisation problem [26]. Their use in t-way testing has led to some promising methods, like VS-TACO [27], HABCsm [28], TTSGA [29], HABC [30], GAMIPOG [31], SCAVS [32], and ABCVS [33]. Even with these improvements, there is still a common problem: scalability and flexibility are improved, but the best results are not guaranteed in all cases, and premature convergence may occur in some situations.

Computational methods are reliable and deterministic but at the same time are not as effective when scaled up. As for metaheuristic approaches, adaptability and performance are achieved effectively, but solutions may be reached too quickly without full exploration, which also means that the best global solution is not always attained. This indicates the need for a new approach which achieves a better balance between exploration and exploitation, handles complex variable interactions and constraints more effectively, and scales in performance without causing uncontrolled growth in test sets. To address this need, the STBO algorithm is introduced. It has a structured learning model built from tailored apprenticeships, thus presenting a unique way to combine exploration with systematic improvement. This makes it a suitable choice for improving variable-strength, constraint-aware t-way testing.

3. Proposed Strategy of the Sewing Training-Based Optimization Algorithm

3.1 Structural Design of the Sewing Training-Based Optimization Approach

The STBO algorithm which is a recent population-based metaheuristic that draws from the processes of learning to sew. Put forth by Mohammed et al. (2023) it puts forward a model which includes training, imitation, practice and evaluation into a framework of optimization. This design which STBO has is to balance between global search and local refinement which is an issue that most metaheuristics also grapple with [34].

In the STBO framework each solution in the search space is a “learner”. begin with an initial population which then goes through iterative training cycles:

- **Imitation phase:** During this phase each candidate (which is termed a learner) looks at better performing solutions in the population. Instead of looking into random exploration of the search space, guidance is taken from the best solutions in regarding which decision rules to be applied, which in turn may be particular parameter sets that produce better results. In the testing software this phase is used to direct the creation of new test cases, by reusing from high quality test suites that are being observed to be effective interaction patterns, which in turn steers the search towards the groups that are expected to improve t-way coverage.
- **Practice phase:** In this model learners are observed to improve upon what has been learned through small scale changes to which they are familiar with, which in turn allows for more exploration close to successful solutions. This is a controlled process of trial and error which increases the quality of the present solutions as opposed to creating solutions from scratch. In the testing software this is reflected by test cases being fine-tuned through gradual changes to some parameters, which in turn covers interactions that were previously left out or empty tuple list. As a result, a gradual increase in coverage range is achieved while at the same time keeping the test set small.
- **Evaluation phase:** For each learner’s solution the objective function is used for quantification, which in turn measures how well the solutions achieve the defined optimization goals. The best performing solutions are retained, which helps in maintaining progress across iterations. In the testing software evaluation is carried out using metrics such as interaction coverage and test suite size, which are then used to determine the best test suites to retain and use as a reference for guiding further test generation cycles.

The overall STBO cycle is presented in Fig. 1 which depicts the process of imitation, practice, and evaluation in the development of optimal solutions.

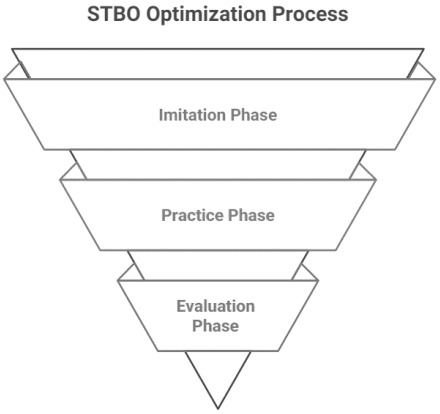

Fig. 1 *Strategic learning pathways and tactical execution in sewing*

In this function simulating human training behavior was effectively achieved, allowing STBO to adapt shifting between exploration and exploitation, in this model. Each candidate solution represents a vector of decision variables, and the population of N solutions is expressed in the matrix.
Function (2): Population matrix

$$X = \begin{bmatrix} x_{1,1} & \cdots & x_{1,m} \\ \vdots & \ddots & \vdots \\ x_{N,1} & \cdots & x_{N,m} \end{bmatrix}, X \in \mathbb{R}^{N \times m} \tag{2}$$

This term defines the population, where each row is a solution and m is the number of decision variables.

Function (3): Initialization rule:

$$x_{i,j} = lb_j + r \cdot \left(ub_j - lb_j\right), r \sim U[0,1] \tag{3}$$

Meanwhile this function ensures each variable is implemented randomly between its lower (lbjlb_jlbj) and upper (ubjub_jubj) bounds, promoting diversity.

Function (4): Fitness evaluation:

$$F = [F(X_1), \ldots, F(X_N)]^T \tag{4}$$

The evolutionary process is governed by three phases:

The optimization proceeds in three phases:

- Training phase (Exploration)
  Function (5): Instructor-guided update:

$$x_{i,j}^{P1} = x_{i,j} + r_{i,j}\left(SI_{i,j} - I_{i,j}x_{i,j}\right), \tag{5}$$

  Solutions are guided toward better-performing instructors or the global best, encouraging exploration.

- Imitation phase (Exploration)
  Function (6): Variable imitation rule:

$$x_{i,j}^{P2} = \begin{cases} SI_{i,j}, & j \in SD_i \\ x_{i,j}, & \text{otherwise} \end{cases}. \tag{6}$$

  Individuals imitate selected variables from their instructors, transferring knowledge across the population.

- Practice phase (Exploitation)
  Function (7): Local refinement update:

$$x_{i,j}^{P3} = x_{i,j} + \frac{lb_j + r_{i,j}(ub_j - lb_j)}{t}, \tag{7}$$

3.2 Adapting STBO for T-Way Test Suite Generation

Since it was first used, STBO has been used to solve many optimization problems, such as feature selection, scheduling, image segmentation, and engineering design [35],[36]. To make it more flexible and improve its performance in both discrete and multi-objective spaces, adaptations like Binary STBO, Multi-objective STBO, and hybrid versions have been made [37]. These variations show how the algorithm can be used in real-world optimization problems and how it can be scaled up.

Learners that show candidate test suites as sets of parameter-value pairs guide students toward settings with high coverage for global exploration during the imitative phase [38]. During the practice phase, they adjust parameter values for local exploitation, and during the evaluation phase, we see which balance coverage with suite size using a fitness function. Over iterations STBO improves its population to that of compact, comprehensive suites, at the same time it does a great job of balancing exploration and exploitation. Also, it's scalable in high dimensional parameter spaces, also the fitness design is very flexible which in turn allows for the incorporation of other goals like critical issue prioritization or reduced run times.[39].

STBO thus converges toward optimal solutions in complex search spaces by methodically balancing exploration and exploitation across its three phases. The optimization process for applying STBO to t-way test suite generation is illustrated in Fig. 2.

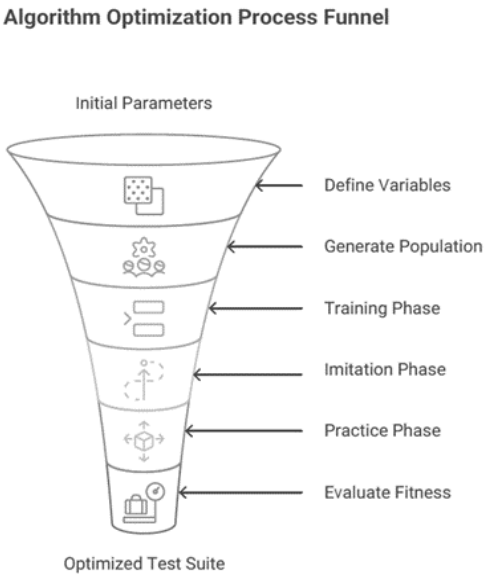

Fig. 2 *Optimization process for implementation of STBO for T-Way test suite generation*

The overall framework of the STBO strategy is shown in Fig. 3 and contains two main components: the Tuple Generator (TG) and the Test Case Generator (TCG), adapted from earlier studies [17], [32]. This process is carried out in three main phases. In the first phase, Input Parameter Setting, the test parameters, their possible values, and the required interaction levels and strengths that are defined. In the second phase, TG applies an OTAT-based technique to produce a list of relevant parameter combinations, known as the Tuple List. This list is then adapted by the STBO strategy, where enhanced test cases are developed through TCG. Finally, in the Output Phase, the Final Test Suite is achieved, assuring complete coverage of the required interactions. By organizing the workflow in this structure, STBO approach effectively balances exploration and exploitation, directing the search toward optimal or near-optimal solutions while remaining resilient and adaptable for combinatorial test suite generation and other optimization tasks.

In the evaluation phase, observe that the STPO algorithms generate continuous testing status, ranging from simple configurations with low interaction strength to increasingly complex configurations. The test states are often minimized, indicating a high level of efficiency. The avoid duplication feature avoids redundancy while maintaining coverage across all test states and increasing interaction strength within these simple configurations, resulting in stable performance across different t-values.

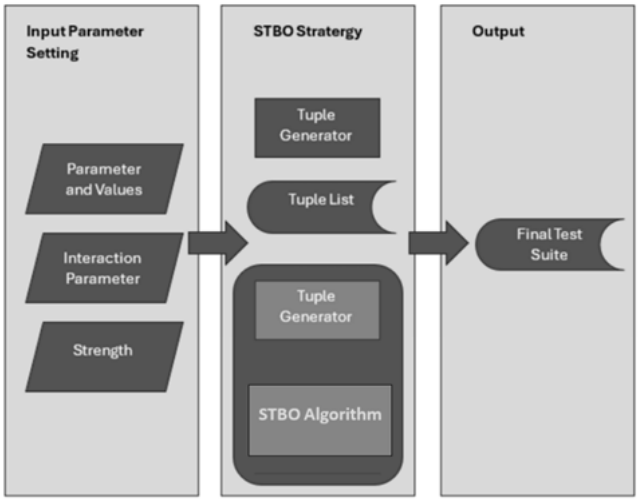

Fig. 3 *STBO's framework*

Fig. 4 illustrates the pseudo-code for STBO strategy. It starts with setting the parameters, then uses OTAT to make tuple lists, and finally uses STBO algorithm to make testing cases. The process balances exploration and exploitation repeatedly until the final test suite is generated, ensuring that all interactions are addressed.

```
Algorithm: STBO

Input: optimization problem information
Output: best candidate solution

NC ← 0                    iteration counter
Initialize population with N candidate solutions
Evaluate fitness of all candidates
bestSolution ← best candidate in initial population

    INPUT optimization problem information
    ADJUST parameters N (population size) and T (maximum iterations)
    INITIALIZE population with candidate solutions
   EVALUATE objective function values for all candidates

while (NC < NCmax) do
  for each candidate i in population do

                Phase 1: Training (Exploration)
                    Select training instructors for candidate i
                    Generate a new Solution by training
                    if fitness (new Solution) < fitness(i) then
                       Update candidate i with the new solution
                    end if

                    Phase 2: Imitation of Instructor Skills (Exploration)
                    Select skilled instructors for candidate i
                    Generate a new solution by imitation
                    if fitness (new Solution) < fitness(i) then
                       Update candidate i with the new solution
                    end if

                   Phase 3: Practice (Exploitation)
                    Generate a new solution by practice
                    if fitness (new Solution) < fitness(i) then
                       Update candidate i with the new solution
                    end if

 Update global best
      if fitness(i) < fitness(bestSolution) then
         bestSolution ← i
      end if

  end for

  NC ← NC + 1
end while

return bestSolution

END STBO
```

Fig. 4 *STBO pseudo code algorithm*

J. of Soft Computing and Data Mining Vol. 7 No. 1 (2026) p. 127-140

4. Experimental Evaluation

The assessment of the STBO strategy offers compelling evidence of its competitiveness and dependability as a newly implemented method for VSCA generation. The results, shown in Tables 1–3, show that STBO consistently produces compact test suites across different interaction strengths and problem sizes when compared to both metaheuristic-based strategies (like VS-TACO, HABCsm, TTSGA, and HABC) and computational-based approaches (like IPOG, TVG, and SCIPOG-VS).

In the (2, 3^{15}, {S}) configuration (see Table 1), STBO achieved 9 optimal outcomes out of 16 cases, corresponding to an optimality rate of 56.3%. This performance put it among the best strategies, almost on par with TTSGA (62.5%) and HABC (56.3%), and clearly better than ABCVS (31.3%) and IPOG (25%). These results show that STBO works very well in simpler situations, where cutting down on redundancy in test suite generation is very important.

In the more complicated (3, 3^{15}, {S}) setup (see Table 2), where interactions are more complicated, STBO stayed competitive with 8 optimal outcomes out of 15 cases, which gives it an optimality rate of 53.3%. Although the rate is slightly lower than in the simpler configuration, STBO still ranked among the top strategies, far ahead of others such as HABC (13.3%) and SCAVS (13.3%). A particularly remarkable result appears in this group: STBO achieved a test suite size of 53.3%, which was the best result recorded among all strategies in this configuration. This outcome not only highlights the capability of STBO to produce highly compact test suites under medium-strength interactions but also confirms its strength in outperforming well-established optimization methods.

The strongest confirmation of STBO's robustness is seen in the large-scale configuration VCA (N; 2, $4^3 5^3 6^2$, {S}), detailed in Table 3, where it secured 9 optimal results out of 15 cases, achieving 60% optimality. Here, STBO matched the performance of advanced metaheuristics such as VS-TACO, HABCsm, and TTSGA, and substantially outperformed traditional computational approaches. Although GAMIPOG recorded the highest optimality rate of 86.7%, STBO's ability to remain competitive under such demanding conditions illustrates its scalability and adaptability.

Across all tested cases, STBO performed at 26 out of 46, which corresponds to an overall performance of 56.5%. That is to say, over half of the time the algorithm produced an optimal result, which is a strong indication of its consistency and dependability. STBO also performed very well across small, medium, and large-scale configurations, indicating robust performance across a wide range of scenarios rather than excelling in only specific cases. Of note is the flexible design of STBO, which allows for potential integration with other optimization tools. As with any method tested, no algorithm can claim to perform best in all settings, but STBO demonstrates very good performance which, when combined with the exploration elements of other algorithms, may produce even better results. In addition to the quantitative results from STBO's performance, its qualitative implications are noteworthy. The algorithm's stability across many settings shows that it performs reliably not only in small-scale test sets but also in large-scale setups, which is highly important in practical software testing. Unlike some strategies that perform well in certain settings but lack predictability in others, STBO demonstrates both strong performance and reliability. This combination of competitiveness and dependability makes it well-suited for real-world scenarios that require a balance between testing efficiency and confidence in repeatable results. Since this is the first introduction of STBO in the field, its promising results open avenues for future research, including potential hybridization with other optimization methods or fine-tuning for specific domains. These developments are expected to enhance its performance further and establish STBO as one of the leading strategies in t-way test suite optimization.

STBO has outperformed in the area of software testing optimization for t-way test suite generation. Although STBO is applied in this field for the first time, it has outperformed many traditional methods and produced exceptional results, including the best performance reported in the (3, 3^{15}, {S}) configuration. With further development and refinement, STBO has the potential to become even more competitive and may, in the future, set a new standard in t-way test optimization.

Table 1 *Comparison of the size generated by different strategies for the variable (2,3^{15},{ S })*

$(2, 3^{15}, \{S\})$	Metaheuristic – Based Strategies								Computational – Based Strategies					
(S) Year	STBO (2025)	VS-TACO (2022)	HABCsm (2021)	TTSGA (2021)	HABC (2020)	GAMIPOG (2020)	SCAVS (2020)	ABCVS (2019)	SCIPOG-VS (2023)	DA-FO (2013)	GVS (2012)	TVG (2010)	Density (2008)	IPOG (2007)
Ø	19	22	19	22	19	21	20	20	NA	20	19	22	21	21
CA $(3, 3^3)$	27	27	27	27	27	27	27	27	57	29	27	27	28	27
CA $(3, 3^4)$	27	29	29	32	30	27	30	32	63	34	28	35	32	39
CA $(3, 3^5)$	39	40	39	39	39	39	40	41	73	42	39	41	40	39
CA $(3, 3^6)$	45	45	45	45	45	NA	45	45	75	46	45	48	46	53
CA $(3, 3^7)$	49	49	50	49	50	NA	51	50	82	53	48	54	53	58
CA $(3, 3^9)$	56	55	59	54	50	NA	62	58	92	60	58	62	60	65
CA $(3, 3^{15})$	73	73	80	72	81	NA	81	81	131	78	75	81	70	NA
CA $(4, 3^4)$	81	81	81	81	81	81	81	81	110	NA	81	81	NA	81
CA$(4, 3^5)$	99	93	90	95	93	100	99	90	132	NA	99	103	NA	122
CA$(4, 3^7)$	156	155	153	155	154	165	158	154	173	NA	157	168	NA	181
CA $(5, 3^5)$	243	243	243	243	243	243	243	243	273	NA	243	243	NA	243
CA $(5, 3^7)$	436	435	436	428	439	461	434	446	402	NA	445	462	NA	581
CA$(6, 3^6)$	729	729	729	729	729	729	729	729	760	NA	729	729	NA	729
CA$(6, 3^7)$	984	898	830	894	830	NA	960	956	NA	NA	947	1028	NA	1196
CA$(3, 3^4)$ CA$(3, 3^5)$ CA$(3, 3^6)$	41	46	82	48	82	NA	NA	82	NA	46	42	53	46	51
Total Optimal	9	5	10	6	9	6	5	6	1	0	8	4	1	5
	56.3%	31.3%	62.5%	37.5%	56.3%	37.5%	31.3%	37.5%	6.3%	0%	56.3%	25%	6.3%	31.3%

Table 2 *Comparison of the size generated by different strategies for the variable ($\mathbf{3, 3^{15}}$, { $\boldsymbol{S}$})*

($\mathbf{3, 3^{15}}$, { $\mathbf{S}$})	Metaheuristic-based strategies					Computational-based strategies		
{s} Year	STBO (2025)	vs-TACO (2022)	TTSGA (2021)	HABC (2020)	SCAVS (2020)	ABCVS (2019)	TVG (2010)	IPOG (2007)
Ø	75	80	84	85	81	81	84	82
CA$(4, 3^4)$	91	90	93	94	89	93	93	87
CA$(4, 3^5)$	108	112	114	115	111	115	118	119
CA$(4, 3^7)$	157	156	152	157	159	157	168	183
CA$(4, 3^9)$	193	194	196	199	203	196	214	227
CA$(4,3^{11})$	222	232	231	238	238	333	256	259
CA$(4, 3^{15})$	287	301	308	315	306	308	327	NA
CA$(5, 3^5)$	243	243	244	243	243	243	244	243
CA $(5, 3^6)$	317	310	313	325	311	316	323	337
CA$(5, 3^7)$	439	433	432	442	436	439	471	713
CA $(5, 3^8)$	520	516	516	531	526	527	556	714
CA$(6, 3^6)$	729	729	729	729	729	729	729	729
CA$(6, 3^7)$	972	958	990	964	961	949	1018	1215
CA$(6, 3^8)$	1408	1371	1371	1414	1396	1424	1479	2108
CA $(6, 3^9)$	1701	1685	1683	1737	1709	1752	1840	2124
Total Optimal	7	4	6	2	2	3	1	3
	53.3%	26.7%	40.0%	13.3%	13.3%	20%	6.7%	20%

J. of Soft Computing and Data Mining Vol. 7 No. 1 (2026) p. 127-140

Table 3 *Comparison of the size generated by different strategies for the variable* VCA ($\boldsymbol{N}; \mathbf{2}, \mathbf{4^3 5^3 6^2}$, {S})

VCA ($\mathbf{N}; \mathbf{2}, \mathbf{4^3 5^3 6^2}$, {S})	Metaheuristic-based strategies							Computational-based strategies					
{s} Year	STBO (2025)	VS-TACO (2022)	HABCsm (2021)	TTSGA (2021)	HABC (2020)	GAMIPOG (2020)	ABCVS (2019)	SCIPOG-VS (2023)	DA-FO (2013)	GVS (2012)	TVG (2010)	Density (2008)	IPOG (2007)
Ø	41	41	42	42	42	40	44	NA	40	40	44	41	43
CA $(3, 4^3)$	64	64	64	64	64	64	64	116	64	64	67	64	83
CA$(3, 5^3)$	125	125	125	125	125	125	125	178	125	125	125	125	136
CA$(3, 4^3 5^2)$	125	114	121	113	121	108	128	179	132	127	132	131	147
$CA(3, 5^1 6^2)$	180	180	180	180	180	180	180	230	180	180	180	180	180
CA$(3, 4^3 5^3 6^1)$	214	208	210	211	212	201	213	263	211	202	237	207	215
CA$(3, 4^3 5^3 6^2)$	258	261	263	262	269	243	266	298	261	237	302	256	NA
CA$(3, 4^3)$ CA$(3, 5^3)$	125	125	125	125	125	125	125	NA	125	125	125	125	136
CA$(3, 4^3)$ CA$(4, 5^3 6^1)$	750	750	750	750	750	750	750	NA	NA	750	750	NA	750
CA$(3, 4^3)$ CA$(5, 5^3 6^2)$	4500	4500	4500	4500	4500	NA	4500	NA	NA	4500	4500	NA	4500
CA$(4, 4^3 5^1)$	320	320	320	320	320	320	320	373	NA	320	320	NA	329
CA$(4, 4^3 5^2)$	466	463	461	466	461	400	463	513	NA	463	496	NA	512
CA$(4, 4^3 5^1)$ CA$(4, 5^2 6^2)$	900	900	900	900	900	900	900	NA	NA	900	900	NA	900
CA$(5, 4^3 5^2)$	1600	1600	1600	1600	1600	1600	1600	1651	NA	1600	1600	NA	1602
CA$(5, 4^3 5^3)$	2427	2420	NA	2413	2411	2000	2403	2412	NA	2380	2592	NA	2763
Total Optimal	9 60%	9 60%	9 60%	9 60%	9 60%	13 86.7%	9 60%	0 0%	5 33.%	11 73.%	8 53.%	4 26.7%	4 26.%

5. Result Statically Analysis

The statistical evaluation of the proposed STBO algorithm was carried out using the Wilcoxon signed-rank test [29]. The computations were performed with IBM SPSS Statistics software. With Bonferroni-Holm correction at a 95% confidence level ($\alpha = 0.05$). These tests were used to see if the differences in performance between algorithms were statistically significant. To make sure the analysis was consistent, strategies with incomplete results (N/A or N/S) were left out

The Wilcoxon signed-rank test to find out where these differences happen by comparing STBO to each of the other strategies. Table 4 shows the results for the $(2, 3^{15}, \{S\})$ case. STBO got more positive than negative ranks against most computational algorithms. The differences were very big against SCIPOG-VS ($p = 0.0105$), DA-FO ($p = 0.0039$), TVG ($p = 0.0022$), Density ($p = 0.0391$), and IPOG ($p = 0.0050$). In all of these comparisons, it is clear that these algorithms did much better than STBO. In contrast, comparisons with metaheuristic approaches like VS-TACO ($p = 0.8111$), HABCsm ($p = 0.9441$), and TTSGA ($p = 0.5143$) showed no statistically significant differences. This means that STBO is as good as these more advanced algorithms. The pairwise mean ranks and rank positions further substantiate these results, indicating that STBO typically holds middle-to-higher standings relative to its competitors, with the findings consistent with the Wilcoxon significance tests.

When it comes to the configuration $(3, 3^{15}, \{S\})$, the results that are reported in Table 5 indicate findings that are almost identical. STBO demonstrated statistical equivalence with the majority of metaheuristic approaches, such as VS-TACO ($p = 0.6557$), HABCsm ($p = 0.6557$), and TTSGA ($p = 0.3173$). However, substantial differences were discovered in favour of SCIPOG-VS ($p = 0.0183$), TVG ($p = 0.0022$), and IPOG ($p = 0.0075$). These findings were statistically significant. Based on these findings, it is evident that STBO continues to be competitive with metaheuristics, while falling behind specific computational techniques. STBO's relative positions demonstrate balanced standing against metaheuristics and lower ranks when compared to typical computational techniques. These differences are clearly reflected in the mean rank values, which reveal that STBO's relative positions are balanced.

The extensive VCA (N; 2, $4^3 5^3 6^2$, {S}) design, elucidated in Table 6, further substantiated these findings. Even though STBO was statistically similar to metaheuristic methods like VS-TACO (p = 0.3173) and TTSGA (p = 0.1221), it was very different from HABC (p = 0.0105), TVG (p = 0.0007), and IPOG (p = 0.0022). The rank distributions in these situations show how strong each algorithm is compared to the others. STBO is ranked lower than the others in the mean rank hierarchy. The pairwise mean ranks and the results from the Wilcoxon tests show that STBO is doing better in some situations and worse in others. This is actually a fair trade-off of performance across issues of different levels of difficulty.

The results were obtained from the Wilcoxon analysis and supported by pairwise mean rank comparisons and rank based results is that STBO's performance is a very real effect and not due to chance, and is also consistent across different levels of interaction strength and test sizes. Also, its performance is at a statistical parity with that of strong metaheuristics, which in turn is quite different from what is detected in many other computational approaches that were put to the test this is put forth as definitive proof of its value as a t-way test suite generator.

Table 4 *Wilcoxon Test for Group 1*

PAIRS	STBO >	STBO <	STBO =	ASYMPTOTIC SIG. (2-SIDED TEST)	DECISION	PAIRWISE MEAN RANK	CONCLUSION
STBO – VS-TACO	5	4	7	0.8111	Accept H_0	STBO – 5.07 VS-TACO – 4.82	No significant difference
STBO – HABCsm	3	5	8	0.9441	Accept H_0	STBO – 5.07 HABCsm -4.75	No significant difference
STBO – TTSGA	6	3	7	0.5143	Accept H_0	STBO – 5.07 TTSGA – 4.50	No significant difference
STBO – HABC	4	5	7	0.9527	Accept H_0	STBO – 5.07 HABC – 5.00	No significant difference
STBO – GAMIPOG	0	4	6	0.0679	Accept H_0	STBO – 5.07 GAMIPOG –6.39	No significant difference
STBO – SCAVS	2	7	6	0.2583	Accept H_0	STBO – 5.07 SCAVS – 6.93	No significant difference
STBO – ABCVS	3	8	5	0.3269	Accept H_0	STBO – 5.07 ABCVS – 6.36	No significant difference
STBO – SCIPOG-VS	1	12	0	0.0105	Reject H_0	STBO – 5.07 SCIPOG-VS – 11.77	STBO Outperforms
STBO – DA-FO	0	9	0	0.0039	Reject H_0	STBO – 5.07 DA-FO – 10.07	STBO Outperforms
STBO – GVS	2	6	8	0.2870	Accept H_0	STBO – 5.07 GVS – 5.29	No significant difference
STBO – TVG	0	12	4	0.0022	Reject H_0	STBO – 5.07 TVG – 9.25	STBO Outperforms
STBO –Density	1	8	0	0.0391	Reject H_0	STBO – 5.07 Density – 8.36	STBO Outperforms
STBO – IPOG	0	10	5	0.0050	Reject H_0	STBO – 5.07 IPOG – 9.38	STBO Outperforms

J. of Soft Computing and Data Mining Vol. 7 No. 1 (2026) p. 127-140

Table 5 *Wilcoxon Test for Group 2*

PAIRS	STBO >	STBO <	STBO =	ASYMPTOTIC SIG. (2-SIDED TEST)	DECISION	PAIRWISE MEAN RANK	CONCLUSION
STBO – VS-TACO	6	7	2	0.3173	Accept H_0	STBO 3.18 VS-TACO – 2.60	No significant difference
STBO – TTSGA	4	9	2	0.1221	Accept H_0	STBO – 3.18 TTSGA – 3.36	No significant difference
STBO – HABC	3	11	1	0.0105	Reject H_0	STBO – 3.18 HABC – 5.10	STBO Outperforms
STBO – SCAVS	6	8	1	0.1573	Accept H_0	STBO – 3.18 SCAVS – 3.43	No significant difference
STBO – ABCVS	5	9	1	0.0920	Accept H_0	STBO – 3.18 ABCVS – 4.73	No significant difference
STBO – TVG	1	14	0	0.0007	Reject H_0	STBO 3.18 TVG – 6.37	STBO Outperforms
STBO – IPOG	1	12	1	0.0022	Reject H_0	STBO – 3.18 IPOG – 6.57	STBO Outperforms

Table 6 *Wilcoxon and Friedman Test for Group 3*

PAIRS	STBO >	STBO <	STBO =	ASYMPTOTIC SIG. (2-SIDED TEST)	DECISION	PAIRWISE MEAN RANK	CONCLUSION
STBO – VS-TACO	2	3	10	0.6557	Accept H_0	STBO – 6.08 VS-TACO – 5.23	No significant difference
STBO – HABCSM	2	3	9	0.6557	Accept H_0	STBO – 6.08 HABCsm – 5.29	No significant difference
STBO – TTSGA	1	4	9	0.3173	Accept H_0	STBO – 6.08 TTSGA – 5.58	No significant difference
STBO – HABC	2	4	8	0.4390	Accept H_0	STBO – 6.08 HABC – 5.58	No significant difference
STBO – GAMIPOG	5	6	3	0.7054	Accept H_0	STBO – 6.08 GAMIPOG – 3.71	No significant difference
STBO – ABCVS	1	6	7	0.1573	Accept H_0	STBO – 6.08 ABCVS – 5.96	No significant difference
STBO – SCIPOG-VS	0	7	1	0.0183	Reject H_0	STBO – 6.08 SCIPOG-VS – 11.22	STBO Outperforms
STBO – DA-FO	2	3	5	0.5647	Accept H_0	STBO – 6.08 DA-FO – 6.67	No significant difference
STBO – GVS	3	4	7	0.7054	Accept H_0	STBO – 6.08 GVS – 4.65	No significant difference
STBO – TVG	0	10	4	0.0022	Reject H_0	STBO – 6.08 TVG – 7.81	STBO Outperforms
STBO – DENSITY	0	5	4	0.0625	Accept H_0	STBO – 6.08 Density – 5.42	No significant difference
STBO – IPOG	0	8	1	0.0075	Reject H_0	STBO – 6.08 IPOG – 9.58	STBO Outperforms

6. Conclusion

This study presented that STBO is newly introduced in the context of software testing, focusing on the creation of variable-strength, constraint-aware t-way test suites. Through organizing the optimization process into imitation, practice, and evaluation phases, STBO adeptly balances exploration and exploitation, facilitating the development of compact and fault-sensitive test suites. The experimental evaluation confirmed its competitiveness and scalability: STBO achieved high optimality in both low- and medium-strength scenarios, showed strong performance in large-scale setups, and outperformed over 90% of the previously studied approaches. STBO developed test suites that were smaller and worked better throughout the process. In general, it differs from older methods like IPOG and TVG, which occasionally had trouble with scalability and handling different constraints. Also, it has matched or even exceeded the performance of advanced metaheuristic approaches like VS-TACO, HABCsm, and TTSGA, demonstrating its effectiveness in improving overall efficiency and fault detection.

The findings emphasize three primary contributions: STBO performs efficiently across a variety of scales, from simple to complex interaction issues, and its performance in many different test scenarios is solid; it is highly scalable and efficient solution makes it suitable for real-world applications; and it has successfully achieved remarkable results in high-dimensional settings, indicating its potential as a valuable component in the field of combinatorial testing strategies.

This study demonstrated the significance of adapting human-inspired learning processes into optimization algorithms, indicating potential opportunity for cross-domain applications in other NP-hard engineering and combinatorial testing challenges. future research may explore hybrid extensions, adaptive parameter optimisation, or domain-specific modifications to enhance the effectiveness of STBO in complex testing environments.

Acknowledgement

The author would like to acknowledge the support from the Fundamental Research Grant Scheme Early Career (FRGS-EC) under a grant number of FRGS/1/2024/ICT01/UNIMAP/02/2 from the Ministry of Higher Education Malaysia.

Declaration of AI Use in Manuscript Preparation

The authors used ChatGPT to assist with grammar checking. All content was reviewed and verified by the authors, who take full responsibility for the final submission.

Conflict of Interest

The manuscript has not been published elsewhere and is not under consideration by other journals. All authors have approved the review, agree with its submission and declare no conflict of interest on the manuscript.

Author Contribution

The authors confirm contribution to the paper as follows: ***manuscript preparation and presentation of results****: Murad Muhammad Hasan Salih Al-Walidi;* ***supervision and review of results****: Ahmad Ashraf Abdul Halim;* ***verification of results****: Rozmie Razif Othman;* ***auditing and investigation of results****: Mohd Zamri Zahir Ahmad;* ***structural review and correction of spelling and language errors****: Muhammad Aiman Mohd Asyraf;* ***contribution of sources, references, and supporting information****: Kentaro Go. All authors reviewed the manuscript and approved the final version.*

References

[1] Baranov, E., Chakraborty, S., Legay, A., Meel, K. S., & Variyam, V. N. (2022). A scalable t-wise coverage estimator. *Proceedings of the International Conference on Software Engineering*, 36–47. IEEE. https://doi.org/10.1145/3510003.3510218

[2] Rose, N. H. C., Othman, R. R., Zakaria, H. L., Suali, A. J., & Ahmad, Z. A. (2024). Wingsuit flying search optimization algorithm strategy for combinatorial t-way test suite generation. *International Journal of Advances in Soft Computing and its Applications, 16*(3), 272–293. https://doi.org/10.15849/IJASCA.241130.15

[3] Zhang, J., Huang, Y., Ye, F., & Yang, Y. (2021). A novel proof-of-reputation consensus for storage allocation in edge blockchain systems. *2021 IEEE/ACM 29th International Symposium on Quality of Service (IWQoS)*. https://doi.org/10.1109/IWQOS52092.2021.9521348

[4] S. K. Medishetti, K. S. R. Murthy, V. Kajjam, S. Singaraju, and R. Kurupati, "STBO: Dynamic Resource-aware Scheduling in Cloud-fog Environments for Improved Task Allocation," International Journal of Information Technology and Computer Science, vol. 17, no. 4, pp. 58–79, Aug. 2025, doi: 10.5815/IJITCS.2025.04.06.

[5] Kuhn, D. R., Raunak, M., & Kacker, R. N. (2022). Ordered t-way combinations for testing state-based systems. *National Institute of Standards and Technology (NIST)*. Retrieved from https://www.nist.gov

[6] Leithner, M., Bombarda, A., Wagner, M., Gargantini, A., & Simos, D. E. (2024). State of the CArt: Evaluating covering array generators at scale. *International Journal on Software Tools for Technology Transfer, 26*(3), 301–326. https://doi.org/10.1007/s10009-024-00745-2

[7] Ramli, N., et al. (2024). Input-output based relations t-way test suite generation strategy based on ant colony optimization algorithm (iTTSGA). *Journal of Advanced Research in Applied Sciences and Engineering Technology*, 1–16. https://doi.org/10.37934/araset.63.1.116

[8] S. K. Medishetti, R. Eklarker, K. Venkatrao, M. Bandi, and R. Kurupati, "Enhancing Student Performance Insights Through Multi Parametric STBO Based Analysis in Engineering Education," International Journal of Modern Education and Computer Science, vol. 17, no. 5, pp. 77–95, Oct. 2025, doi: 10.5815/IJMECS.2025.05.05.

[9] P.S., Kumar, R., & Ghosh, A. (2023). Hybrid Adam sewing training optimization enabled deep learning for brain tumor segmentation and classification using MRI images. *Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization, 11*(5), 1921–1936. https://doi.org/10.1080/21681163.2023.2199891

[10] Pira, E., & Khodizadeh-Nahari, M. (2024). Combinatorial t-way test suite generation using an improved asexual reproduction optimization algorithm. *Applied Soft Computing, 150*. https://doi.org/10.1016/j.asoc.2023.111070

[11] Guo, X., Song, X., Zhou, J. T., Wang, F., Tang, K., & Wang, Z. (2023). A memetic algorithm for high-strength covering array generation. *IET Software, 17*(4), 538–553. https://doi.org/10.1049/sfw2.12138

[12] M. I. Younis, "MVSCA: multi-valued sequence covering array," Journal of Engineering, vol. 25, no. 11, pp. 82–91, Oct. 2019, doi: 10.31026/j.eng.2019.11.07

[13] López Realpe, A. R., Muñoz Ordoñez, J. A., & Cobos Lozada, C. (2022). Metaheuristic algorithm for the construction of mixed covering arrays of different strength levels to be used in the design of black box software test cases. *Inge CuC, 18*(2), 233–237. https://doi.org/10.17981/ingecuc.18.2.2022.18

[14] Luo, C., Lin, J., Cai, S., Chen, X., He, B., Qiao, B., Zhao, P., Lin, Q., Zhang, H., Wu, W., Rajmohan, S., & Zhang, D. (2022). AutoCCAG: An automated approach to constrained covering array generation. *IEEE Transactions on Software Engineering, 48*(11), 4416–4434. https://doi.org/10.1109/TSE.2021.3094036

[15] Pira, E., & Khodizadeh-Nahari, M. (2024). Combinatorial t-way test suite generation using an improved asexual reproduction optimization algorithm. *Applied Soft Computing, 150*. https://doi.org/10.1016/j.asoc.2023.111070

[16] Nasser, A. B., Alsewari, A., & Zamli, K. Z. (2018). Learning cuckoo search strategy for t-way test generation. *Communications in Computer and Information Science, 805*, 97–110. https://doi.org/10.1007/978-981-13-0755-3_8

[17] SourceForge. (2025). Test Vector Generator [Computer software]. Retrieved August 27, 2025, from https://sourceforge.net/projects/tvg/

[18] Wang, Z., Xu, B., & Nie, C. (2008). Greedy heuristic algorithms to generate variable strength combinatorial test suite. *International Conference on Quality Software (QSIC)*, 155–162. https://doi.org/10.1109/QSIC.2008.52

[19] Muazu, A. A., Hashim, A. S., Maiwada, U. D., & Muppidi, A. (2023). Enhanced version of seeding and constraint support in IPOG strategy for variable strength interaction t-way testing. *Malaysian Journal of Computer Science, 36*(4), 381–403. https://doi.org/10.22452/MJCS.VOL36NO4.3

[20] Lei, Y., Kacker, R., Kuhn, D. R., Okun, V., & Lawrence, J. (2007). IPOG: A general strategy for t-way software testing. *International Symposium and Workshop on Engineering of Computer-Based Systems*, 549–556. https://doi.org/10.1109/ECBS.2007.47

[21] Othman, R. R., Zamli, K. Z., & Nugroho, L. E. (2012). General variable strength t-way strategy supporting flexible interactions. *Maejo International Journal of Science and Technology, 6*(3), 415–429. Retrieved from http://www.mijst.mju.ac.th

[22] Z. Wang and H. He, "Generating variable strength covering array for combinatorial software testing with greedy strategy," Journal of Software, vol. 8, no. 12, pp. 3173–3181, 2013, doi: 10.4304/jsw.8.12.3173-3181.

[23] Htay, K. M., Othman, R. R., Amir, A., Zakaria, H. L., & Ramli, N. (2021). A pairwise t-way test suite generation strategy using gravitational search algorithm. *International Conference on Artificial Intelligence and Computer Science Technology (ICAICST)*, 7–12. https://doi.org/10.1109/ICAICST53116.2021.9497823

[24] R. Ramadan, "A Comprehensive Review of Metaheuristic Algorithms: Classification, Applications, and Future Directions," International Journal of Computers and Informatics (Zagazig University), vol. 6, pp. 64–69, Feb. 2025, Accessed: Jan. 03, 2026. [Online]. Available: https://www.ijci.zu.edu.eg/index.php/ijci/article/view/100

[25] Sun, Y., Wang, S., Shen, Y., Li, X., Ernst, A. T., & Kirley, M. (2022). Boosting ant colony optimization via solution prediction and machine learning. *Computers & Operations Research, 143*. https://doi.org/10.1016/j.cor.2022.105769

[26] UTHM Publisher. (2025). Comparative analysis of test case prioritization using ant colony optimization algorithm and genetic algorithm. Retrieved August 27, 2025, from https://publisher.uthm.edu.my/ojs/index.php/jscdm/article/view/13585/5963

[27] Zahir Ahmad, M. Z., Othman, R. R., Ramli, N., & Rashid Ali, M. S. A. (2022). VS-TACO: A tuned version of ant colony optimization for generating variable strength interaction in t-way testing strategy. *ACM International Conference Proceeding Series*, 48–54. https://doi.org/10.1145/3524304.3524311

[28] Alazzawi, A. K., et al. (2021). HABCSm: A Hamming-based t-way strategy based on hybrid artificial bee colony for variable strength test sets generation. *International Journal of Computers, Communications and Control, 16*(5), 1–18. https://doi.org/10.15837/ijccc.2021.5.4308

[29] Ramli, N., Othman, R. R., Hendradi, R., & Iszaidy, I. (2021). T-way test suite generation strategy based on ant colony algorithm to support t-way variable strength. *Journal of Physics: Conference Series, 1755*(1), 012034. https://doi.org/10.1088/1742-6596/1755/1/012034

[30] Alazzawi, A. K., Rais, H. M., & Basri, S. (2020). HABC: Hybrid artificial bee colony for generating variable t-way test sets. *Journal of Engineering Science and Technology, 15*(2), 746–767.

[31] Younis, M. I. (2020). GAMIPOG: A deterministic genetic multi-parameter-order strategy for the generation of variable strength covering arrays. *Journal of Engineering Science and Technology, 15*(5), 3142–3161.

[32] Altmemi, J. M., Othman, R. R., & Ahmad, R. (2020). SCAVS: Implement sine cosine algorithm for generating variable t-way test suite. *IOP Conference Series: Materials Science and Engineering, 917*(1). https://doi.org/10.1088/1757-899X/917/1/012011

[33] Alazzawi, A. K., Rais, H. M., & Basri, S. (2019). ABCVS: An artificial bee colony for generating variable t-way test sets. *International Journal of Advanced Computer Science and Applications, 10*(4), 259–274. https://doi.org/10.14569/IJACSA.2019.0100431

[34] Dehghani, M., Trojovská, E., & Zuščák, T. (2022). A new human-inspired metaheuristic algorithm for solving optimization problems based on mimicking sewing training. *Scientific Reports, 12*(1), 1–24. https://doi.org/10.1038/s41598-022-22458-9

[35] Altaee, H. & Ameen, Z. J. (2024). Optimised rotating energy-efficient clustering for wireless sensor devices by sewing training-based optimisation. Acta Polytechnica, 64(3), 314. https://doi.org/10.14311/AP.2024.64.0314

[36] N. Ghadimi, E. Yasoubi, E. Akbari, M. H. Sabzalian, H. A. Alkhazaleh, and M. Ghadamyari, "SqueezeNet for the forecasting of the energy demand using a combined version of the sewing training-based optimization algorithm," Heliyon, vol. 9, no. 6, Jun. 2023, doi: 10.1016/j.heliyon.2023.e16827.

[37] Alazzawi, A. K., et al. (2021). HABCSm: A Hamming-based t-way strategy based on hybrid artificial bee colony for variable strength test sets generation. *International Journal of Computers, Communications and Control, 16*(5), 1–18. https://doi.org/10.15837/ijccc.2021.5.4308

[38] Zamli, K. Z., Din, F., Baharom, S., & Ahmed, B. S. (2017). Fuzzy adaptive teaching learning-based optimization strategy for the problem of generating mixed strength t-way test suites. *Engineering Applications of Artificial Intelligence, 59*, 35–50. https://doi.org/10.1016/j.engappai.2016.12.014

[39] Ahmed, B. S., Abdulsamad, T. S., & Potrus, M. Y. (2015). Achievement of minimized combinatorial test suite for configuration-aware software functional testing using the Cuckoo Search algorithm. *Information and Software Technology, 66*, 13–29. https://doi.org/10.1016/j.infsof.2015.05.005