An improved Jaya Algorithm-based Strategy for T-way Test Suite Generation

Abdullah B. Nasser, Fadhl Hujainah, AbdulRahman A. Al-Sewari and Kamal Z. Zamli

Faculty of Computer Systems and Software Engineering,
Universiti Malaysia Pahang, 26300 Kuantan, Pahang, Malaysia
abdullahnasser@ump.edu.my

Abstract. In the field of software testing, several meta-heuristics algorithms have been successfully used for finding an optimized t-way test suite (where t refers to covering level). T-way testing strategies adopt the meta-heuristic algorithms to generate the smallest/optimal test suite. However, the existing t-way strategies' results show that no single strategy appears to be superior in all problems. The aim of this paper to propose a new variant of Jaya algorithm for generating t-way test suite called Improved Jaya Algorithm (IJA). In fact, the performance of meta-heuristic algorithms highly depends on the intensification and diversification capabilities. IJA enhances the intensification and diversification capabilities by introducing new operators search such lévy flight and mutation operator in Jaya Algorithm. Experimental results show that the IJA variant improves the results of original Jaya algorithm, also overcomes the problems of slow convergence of Jaya algorithm.

Keywords: T-way testing, Meta-heuristics, Jaya Algorithm, improved Jaya Algorithm.

1 Introduction

Optimization is a mathematical method used for solving complex problems many disciplines, including engineering, biology, physics, business, and economics, by finding the best/optimal solution from the alternative set of solutions. Many meta-heuristic algorithms have been used successfully for finding an optimal or near optimal solution within reasonable time including Simulated Annealing (SA),Tabu Search(TS) Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Differential Evolution (DE), Sine Cosine Algorithm (SCA), Flower Pollination Algorithm (FPA), Ant Colony Optimization (ACO), Jaya Algorithm (JA), Optimization Algorithm (TLBO) Cuckoo Search (CS) and Teaching–Learning-Based [1].

The performance of meta-heuristic methods highly depends on the intensification and diversification of the search process. Intensification explores the promising regions in the hope to find better solutions. On the other hand, diversification ensures that all regions have been visited, which allows the algorithm to jump out of local optimum [2].

In the field of software testing, several meta-heuristics algorithms have been successfully used for finding an optimized t-way test suite based on defined interaction strength (t)[3]. Nowadays, many researchers turn into t-way testing. T-way testing is a very efficient approach for minimizing the test cases based on the t interaction coverage. Finding an optimized set of test cases is a non-deterministic polynomial-time hard problem where increasing software inputs lead to an increase in computational time in an exponential manner and the complexity as well[4]. As a consequence, much existing research adopts meta-heuristic algorithms as the basis of their implementation including TS, SA, GA, ACA, PSO[5], HS[6] CS[7] and FPA [8].

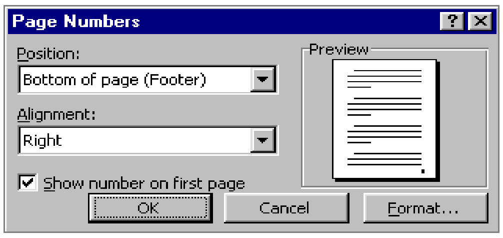
This paper proposes a t-way test suite generation method, called IJA based on an improved of Jaya Algorithm [9] by enhanced its search capabilities. First, the proposed method utilizes lévy flight for enhanced its exploration part. In fact, lévy flight consists of random walks that are interspersed by long jumps which can help to get out of local minima [10]. Second, a mutation operator is added to improve the convergence rate which to measure how fast JA converge.

The rest of this paper is organized as follows. An overview of t-way testing will highlight in Section 2 while Section 3 reviews existing t-way strategies. Section 4 elaborates the description of the proposed strategy. Section VI highlights and discusses the results. Lastly, Section VI gives the conclusion and future work.

2 Overview of T-way Testing

T-way testing ensures that every t-combinations (where t refers to interaction strength) of the inputs are covered in the final test suite. For mere understanding, consider the following insert page numbers dialog box in Microsoft word processor. In this example, three are parameters (with ignoring the buttons) which are:

1. Position has two values (top and bottom).
2. Alignment has three values (Left, center and Right).
3. Show-number-on-first-page has two values (Selected/Yes and Unelected/No).


Page Numbers

Position: Bottom of page (Footer)

Alignment: Right

☑ Show number on first page

Fig. 1. Page Numbers dialog box.

For testing such a simple dialog box, ideally, all the combinations of the inputs are considered. Here are 12 test cases need to be tested that can cover all the interactions. However, by considering 2-way interaction, test suite size can be minimizing from 12 to 6 test cases. Two-way interaction testing ensures that every two-combination of parameters is covered in the final test suite at least one time. It should be noted that there is 50 percent reduction test suite size however the faults detection rate can be reached to 90 percent. Similar to two-way testing, three-way testing increase the faults detection reach to 99 percent [11].

3 Related Works

In the literature review, T-way Testing has been generally treated as an algebraic problem. In this approach, lightweight algebraic computations are adopted for generating the t-way test suite. Generating t-way test suite using the algebraic approach used two methods. The first method is based on mathematical functions where each cell of the test suite is computed directly based on the corresponding row and column using mathematical functions. Whereas the second one employs a recursive process to construct test suite by constructing a large test set from small test sets such as CA, MCA and TConfig, are limited to small configurations [12].

Computational approaches adopt a greedy algorithm to generate the test suite. Each iteration of the generation process tries to cover as many t-way combinations as possible, by generating one parameter at a time(OPAT) or one test at a time(OTAT). OPAT strategies start by generating the test suite for the smallest t-combinations. Then the test suite is extended horizontally by adding a parameter per iteration until all the parameters are covered. Examples of such approach are IPO, IPOG, IPOG-D, IPOF and IPAD2 [13].

OTAT strategies start generating one test case per iteration involved all the parameters that cover the maximum number of combinations. The iteration continues until all the t-combinations are covered. Due to its efficiency, there are many strategies that adopt OTAT techniques such as AETG [14], Jenny [15], TConfig [16], and WHITCH [17]. Recently, many OTAT based strategies adopt meta-heuristic algorithms for generating t-way test suite Applying a meta-heuristic algorithm, or in general search algorithms, in the field of testing also called Search-based software Testing (SBST).

The first publication on SBST was in 1976 however it was completely different from techniques that were used at the time. It was simply generating the test suite that consists of floating-point inputs [18]. Recently, the role of the search algorithm in SBST is to guide the search for a better solution within a limited time.

In the literature, there are many meta-heuristic algorithms has been used successfully for generating t-way test suite such as Genetic Algorithm(GA) [14], Ant Colony Algorithm(ACA)[19], Particle Swarm Optimization[20], Harmony Search[6], Flower Pollination Algorithm(FPA)[21], and Cuckoo Search (CS) [22].

Unlike the aforementioned strategies, new strategies based on more than single search engine methods have been proposed [3, 4, 23-25]. The new strategies attempt to improve the search process for generating test suite by adopting hybridization methods such as hybrid-FPA (eFPA) [3], Tabu search hyper-heuristic (HHH) [4], Q-learning sine-cosine [25], and fuzzy-TLBO (ATLBO) [23].

4 Proposed Strategy

4.1 Original Jaya Algorithm

Jaya Algorithm [9] is one of the population-based algorithms, designed for solving constrained and unconstrained problems. The main idea behind the algorithm is that any candidate solution should approach to the best solution and evade worse solution at the same time. Therefore, for updating current solution $X_{i,j}$, Jaya Algorithm needs only to update the best and worse solutions then update the current solution using the following equation:

$$X'_{i,j} = X_{i,j} + Rnd_1 (X_{best,j} - |X_{i,j}|) - Rnd_2 (X_{\mathrm{worst,j}} - |X_{i,j}|) \quad (1)$$

Where X_{best}, X_{worst} are the best and worst obtained solutions, and $X'_{I,j}$ is the new candidate solution.

In term of parameter tuning, Jaya Algorithm is free parameters algorithm and need only to define common parameters such as population size and iteration number. Fig. 2 show the summary of Jaya Algorithm.

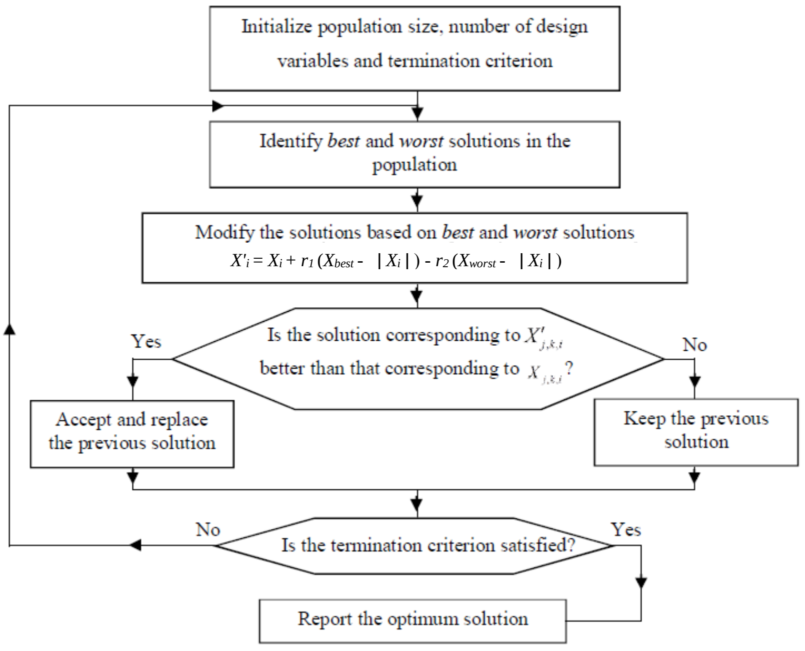

Initialize population size, number of design variables and termination criterion
Identify *best* and *worst* solutions in the population
Modify the solutions based on *best* and *worst* solutions
$X'_i = X_i + r_1 (X_{best} - |X_i|) - r_2 (X_{worst} - |X_i|)$
Is the solution corresponding to $X'_{j,k,i}$ better than that corresponding to $X_{j,k,i}$?
Yes
No
Accept and replace the previous solution
Keep the previous solution
Is the termination criterion satisfied?
No
Yes
Report the optimum solution

Fig. 2. Original Jaya Algorithm

4.2 Improved Jaya Algorithm based Strategy for T-way Test Suite Generation

Improved Jaya Algorithm (IJA) strategy is a t-way testing strategy for generating the test suite. It uses improved Jaya algorithm as core implementation for finding optimal test cases. The strategy starts generating all the t-combinations of the inputs, then using improved Jaya algorithm, IJA attempts to cover all those t-combinations using the smallest test suite.

As shown in Fig. 2, after generating all the t-combinations, improved Jaya algorithm generates its random population, then update the population using Equation 1. To improve the diversification of generated solutions, lévy flight (i.e. Eq. 2) has been deployed on the population. Based on the current solution, a new solution is generated using lévy flight. After updating some solution using lévy flight, mutation operation is applied as well. The mutation operator used to improve convergence speed and maintains the diversity solution of the population from one generation to the next one (i.e. as one or more parts of the solution values are changed subtly).

To maintain the behavior of original Jaya algorithm a portion of the population is updated. Here, we applied elitism technique that used in GA, HS and CS, to ensure the high quality solution is passed to next generation; only a fraction of worse solutions are updated using lévy flight or mutation while the elite solutions (i.e. quality solutions) passed to next generation. In our proposed, we applied a simple mutation operator that randomly changes one parameter of the solution per iteration. Finally, all solutions of the population are evaluated based on fitness function, and the best solution, which cover the maximum number of t-combination, is selected to be added to the final test suite.

$$L\acute{e}vy(\lambda) = \frac{\lambda\Gamma(\lambda)\,sin\left(\frac{\pi\lambda}{2}\right)}{\pi}\,\frac{1}{s^{1+\lambda}},\,(\,s \gg\, s_0 > 0) \tag{2}$$

The recommended value of the distribution factor λ is selected from 0.3 to 1.99 [26]. Γ(λ) is standard gamma function. This distribution is valid for large steps s > 0. Fig. 2 illustrates the proposed improved Jaya algorithm-based strategy for t-way test suite generation (IJA).

```
1:  Input:   p: Parameter number,
             $V: = [v0 .. vj]$ parameter-values and
             $t$: interaction level
             P ∈ [0, 1];
2:  Output: Final test suite $F$;
3:  Let X be a set of population (candidate test cases);
4:  Generate t-combination L based on $P, V$ and t
5:  Generate initial random population
6:  while L is not empty do
7:     while stop criterion not meet do
8:        For each $X_i$ of the population do
9:           Update $X_i$ position using Eq. 1
10:       End For
    // Find a fraction of worse solutions and update them using lévy flight or mutation
11:       For each $X_i$ of the population do
12:        If (random > p)
13:          If (random > 0.5)
14:              Generate $X_{i+1}$ using lévy flight on $X_i$ (Eq. 2)
15:          Else
16:              Apply mutation operator on $X_i$
17:          End if
18:        End if
19:       End for
20:       Evaluate the solutions
21:       Find the best solution
22:    End while
23: Add the best solution to F.
24: Remove covered t-combinations from L
25: End while
26: End-Procedure
```

Fig. 3. Improved Jaya Algorithm Pseudocode for t-way Test Suite Generation

5 Results and Discussion

The proposed strategy is compared with existing t-way strategies such as PSO, CS and original Jaya using different t-way testing problems as shown in Table 1. In order to assess the enhancement of Jaya algorithm, the results of both the original Jaya algorithm and its proposed improvement are involved in the comparison. In addition, we compare the convergence rate of Jaya algorithm against the improved Jaya algorithm. Based on some existing works, the parameters of IJA are set at population size = 50 and the maximum number of improvements = 300 [6, 21, 24]. Different systems are used in this experiment as shown in Table 1. Systems configuration column describes the system such that y^x means the system consists x parameters each parameter has y values.

5.1 Performance Evaluation

In this section, IJA's results are compared with existing meta-heuristic strategies published in [5-7].

Table 1. Comparison with Existing Strategies

System#	**Systems configuration**	**t**	**PSO**	**CS**	**Jaya**	**IJA**
1	2^{10}	2	8	8	10	**8**
2	3^{10}	2	**17**	**17**	23	**17**
3	2^{10}	3	17	**16**	18	**16**
4	3^{6}	3	**42**	43	44	**42**
5	5^{7}	4	1209	1200	1195	**1178**
6	5^{8}	4	1417	1415	1345	**1331**
7	2^{10}	5	82	**79**	85	**79**
8	3^{7}	5	441	439	449	**435**
9	2^{10}	6	158	**157**	163	159
10	3^{7}	6	977	973	**971**	**971**
11	2^{10}	7	NA	NA	312	**297**
12	2^{10}	8	NA	NA	519	**502**
13	2^{10}	9	NA	NA	680	**584**
14	2^{10}	10	NA	NA	**1024**	**1024**

Table 1 presents the results of the comparison; each cell presents the best (i.e. smallest) test suite size obtained by t-way testing strategies while NA indicates that corresponding results not available. The results in Table 1 show that IJA performs better than PSO, CS, and original Jaya and obtains the best results (marks by bold font cell) while PSO obtains the worst results. Comparing only IJA and its counterpart, the results show that IJA performs much better than original Jaya, IJA outperforms original Jaya in almost all cases with exception of cases #10 and #14 where both strategies obtain the same results.

IJA strategy appears to obtains the optimum results in most of the cases owing to enhancement of its lévy flight which help improve the diversification and intensification since the lévy flight can be seen as random walk interrupted by long jump. Another point worth to highlight is that during the experiments, the results are improved after applying mutation operator, that's because the behaviour of mutation operator and how it works are totally different than how Jaya algorithm works. In mutation operator, during updating the solutions, only one input or more are changed while Jaya algorithm updates all inputs.

5.2 5.1 Convergence Rate Analysis

To assess the modification of IJA strategy, the convergence rates of both IJA and original Jaya strategies are studied. The convergence rates use to measure the speedup of meta-heuristic algorithm to reach the optimal solutions. We apply Jaya and IJA on two systems (Table 1) using different values of iteration (i.e. 5, 10, 20, 30, 40, 50, 100, 200, 300, 500, and 1000).

Table 2: Description of Two Problems

No.	Systems	*t*	System's Description
Systems #1	10^5	2	5 parameters each with 10 values
Systems #2	5^2, 7^3, 3^1	3	6 parameters (i.e. 2 parameters with 5 values, 3 parameters with 2 values, and 1 parameter has 3 values.

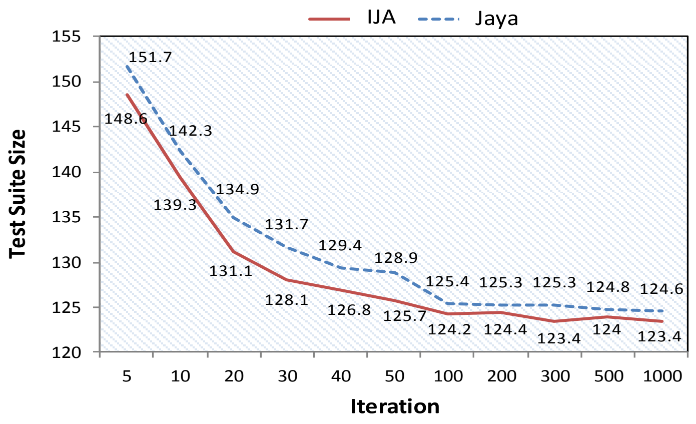


Fig. 4: Convergence rate of Jaya and IJA for System #1

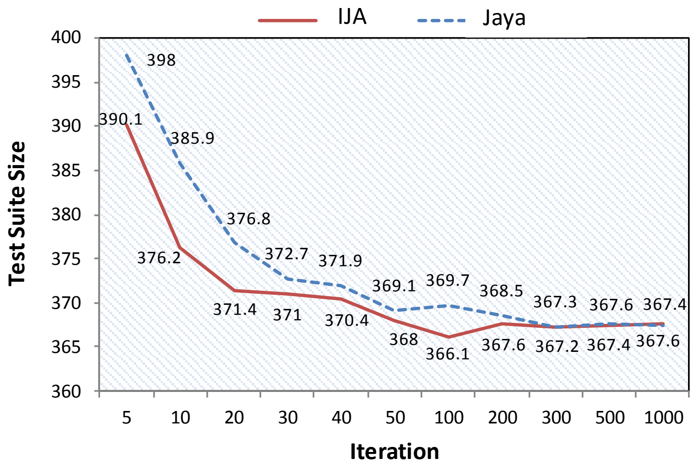


Fig. 5: Convergence rate of Jaya and IJA for System #2

Fig. 4 and 5 show the convergence rate of Jaya and IJA for the two problems. As the figures show, convergence rates of IJA are faster than convergence rates of Jaya in the two problems. From the figures, we can see IJA reach to the optimal solution before the original Jaya due to its improvements by lévy flight and mutation operator.

6 Conclusion

In this paper, an improvement of Jaya algorithm for t-way test generation, called IJA, has been elaborated. IJA enhances the intensification and diversification capabilities by introducing new operators search such lévy flight and mutation operator into Jaya Algorithm. The experimental results show that introducing the new operators into IJA improves the performance of the original Jaya algorithm, also overcomes the problems of slow convergence of Jaya algorithm. As part of future work, we are planning to extend the experimental study by study the diversifications of IJA and include other existing t-way strategies such as eFPA, HHH, and ATLBO which consider as enhanced meta-heuristic based strategies.

Acknowledgements

The work reported in this paper is funded by Universiti Malaysia Pahang (UMP) grants "Prioritized *t*-way Test Suite Generation Strategy Based on Chaotic Flower Pollination Algorithm", RDU190372. We thank UMP for the contribution and supports.

References

1. X.-S. Yang and L. Press, *Nature-inspired metaheuristic algorithms second edition*. Luniver Press, 2010.
2. X. S. Yang, *Nature-Inspired Optimization Algorithms*. Elsevier, 2014.
3. A. B. Nasser, K. Z. Zamli, A. A. Alsewari, and B. S. Ahmed, "Hybrid flower pollination algorithm strategies for t-way test suite generation," *PloS one,* vol. 13, no. 5, p. e0195187, 2018.
4. K. Z. Zamli, B. Y. Alkazemi, and G. Kendall, "A tabu search hyper-heuristic strategy for t-way test suite generation," *Applied Soft Computing,* vol. 44, pp. 57-74, 2016.
5. B. S. Ahmed, K. Z. Zamli, and C. P. Lim, "Constructing a t-way interaction test suite using the particle swarm optimization approach," *International Journal of Innovative Computing, Information and Control,* vol. 8, no. 1, pp. 431-452, 2012.
6. A. R. A. Alsewari and K. Z. Zamli, "Design and implementation of a harmony-search-based variable-strength t-way testing strategy with constraints support," (in English), *Information and Software Technology,* vol. 54, no. 6, pp. 553-568, Jun 2012.
7. B. S. Ahmed, T. S. Abdulsamad, and M. Y. Potrus, "Achievement of minimized combinatorial test suite for configuration-aware software functional testing using the cuckoo search algorithm," (in English), *Information and Software Technology,* vol. 66, pp. 13-29, Oct 2015.
8. A. B. Nasser, A. A. Alsewari, N. M. Tairan, and K. Z. Zamli, "Pairwise test data generation based on flower pollination algorithm," *Malaysian Journal of Computer Science,* vol. 30, no. 3, pp. 242-257, 2017.

9. R. Rao, "Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems," *International Journal of Industrial Engineering Computations,* vol. 7, no. 1, pp. 19-34, 2016.
10. X.-S. Yang and S. Deb, "Cuckoo search via lévy flights," in *World Congress on Nature & Biologically Inspired Computing*, 2009, pp. 210-214: IEEE.
11. D. R. Kuhn, R. N. Kacker, and Y. Lei, "Practical combinatorial testing," *National Institute of Standards and Technology (NIST) Special Publication,* vol. 800, p. 142, 2010.
12. A. Hartman and L. Raskin, "Problems and algorithms for covering arrays," *Discrete Mathematics,* vol. 284, no. 1, pp. 149-156, 2004.
13. Y. Lei, R. Kacker, D. R. Kuhn, V. Okun, and J. Lawrence, "IPOG: A general strategy for t-way software testing," in *Engineering of Computer-Based Systems, 2007. ECBS'07. 14th Annual IEEE International Conference and Workshops on the*, 2007, pp. 549-556: IEEE.
14. D. M. Cohen, S. R. Dalal, M. L. Fredman, and G. C. Patton, "The AETG system: An approach to testing based on combinatorial design," *Software Engineering, IEEE Transactions on,* vol. 23, no. 7, pp. 437-444, 1997.
15. B. Jenkins, "Jenny download page [Online]," p. Available : http://www.burtleburtle.net/bob/math. [Accessed 16 Dec 2014]. 2003.
16. A. Williams, "TConfig download page [Online]," p. University of Ottawa. Available: http://www.site.uottawa.ca/~awilliam/[Accessed 23 Dec 2014]. 2008.
17. A. Hartman, T. Klinger, and L. Raskin, "IBM intelligent test case handler," *Discrete Mathematics,* vol. 284, no. 1, pp. 149-156, 2010.
18. W. Miller and D. L. Spooner, "Automatic generation of floating-point test data," *IEEE Transactions on Software Engineering,* no. 3, pp. 223-226, 1976.
19. T. Shiba, T. Tsuchiya, and T. Kikuno, "Using artificial life techniques to generate test cases for combinatorial testing," in *Computer Software and Applications Conference, 2004. COMPSAC 2004. Proceedings of the 28th Annual International*, 2004, pp. 72-77: IEEE.
20. B. S. Ahmed and K. Z. Zamli, "A review of covering arrays and their application to software testing," *Journal of Computer Science,* vol. 7, no. 9, pp. 1375-1385, 2011.
21. A. B. Nasser, K. Z. Zamli, A. A. Alsewari, and B. S. Ahmed, "An elitist-flower pollination-based strategy for constructing sequence and sequence-less t-way test suite," *International Journal of Bio-Inspired Computation,* vol. 12, no. 2, pp. 115-127, 2018.
22. A. B. Nasser, A. R. A. Alsewari, and K. Z. Zamli, "PairCS: A new Approach of Pairwise Testing based on Cuckoo Search Algorithm," presented at the SOFTEC Asia 2015, Kuantan, Malaysia, 2015.
23. K. Z. Zamli, F. Din, S. Baharom, and B. S. Ahmed, "Fuzzy adaptive teaching learning-based optimization strategy for the problem of generating mixed strength t-way test suites," *Engineering Applications of Artificial Intelligence,* vol. 59, pp. 35-50, 2017.
24. K. Z. Zamli, F. Din, G. Kendall, and B. S. Ahmed, "An experimental study of hyper-heuristic selection and acceptance mechanism for combinatorial t-way test suite generation," *Information Sciences,* vol. 399, pp. 121-153, 2017.
25. K. Z. Zamli, F. Din, B. S. Ahmed, and M. Bures, "A hybrid Q-learning sine-cosine-based strategy for addressing the combinatorial test suite minimization problem," *PloS one,* vol. 13, no. 5, p. e0195675, 2018.
26. I. Pavlyukevich, "Lévy flights, non-local search and simulated annealing," *Journal of Computational Physics,* vol. 226, no. 2, pp. 1830-1844, 2007.