2015 IEEE International Conference on Control System, Computing and Engineering, 27 - 29 November 2015, Penang, Malaysia

Assessing Optimization Based Strategies for t-way Test Suite Generation: The Case for Flower-based Strategy

Abdullah B. Nasser, Yazan A. Sariera, AbdulRahman A. Alsewari, and Kamal Z. Zamli

Faculty of Computer Systems and Software Engineering,
Universiti Malaysia Pahang, 26300 Kuantan, Pahang, Malaysia

***Abstract*— Exhaustive testing is extremely difficult to perform owing to the large number of combinations. Thus, sampling and finding the optimal test suite from a set of feasible test cases becomes a central concern. Addressing this issue, the adoption of t-way testing (where t indicates the interaction strength) has come into the limelight. In order to summarize the achievements so far and facilitate future development, the main focus of this paper is, first, to present a critical comparison of adoption optimization algorithms (OA) as a basis of the t-way test suite generation strategy and, second, to propose a new t-way strategy based on Flower Pollination Algorithm, called Flower Strategy (FS). Analytical and experimental results demonstrate the applicability of FS for t-way test suite generation.**

***Key words:* Analysis of t-way Strategies, Flower Pollination Algorithm, Combinatorial Testing Problem.**

I. INTRODUCTION

Exhaustively testing all possible input/output possibilities are extremely difficult in practice owing to the huge number of combinations needs to be considered. A sampling mechanism is needed to find a subset of effective test case for testing consideration. Random testing provides a useful alternative but , there is no guarantee of fair distribution on the overall search space [1]. While partition testing (e.g. equivalence partitioning, boundary value analysis, and cause-effect graphing) is also useful, no provision is provided to detect faults due to interaction [2]. Parallel testing can enhance random testing and partitioned testing and potentially saves time [3] but there is a limit on how much parallel can be achieved in reality.

Recently, researchers are turning into t-way testing strategy as complementary alternatives to the aforementioned sampling techniques. In a nut shell, t-way testing requires that each t-way combination is to be covered at least one time resulting into significant reduction of test suite size. Many studies show that many software failures are often associated with unnecessary interactions between parameters. Kuhn et al [4] reported that 76% of bugs can be detected by only two-way testing. Study conducted by Brownlie et al [5] reported the usefulness of two-way testing for finding interaction bugs for the AT&T's PMX/StarMAIL system reducing the original test cases from 1500 to 422.

The central concern of many researchers into t-way testing is on how to obtain the minimal number of test cases. Viewed as optimization problem, searching for the optimal set of test cases is also an NP-hard (Non-deterministic Polynomial-time hard) problem.[6-8]. To address this issue, many t-way testing strategies have been designed and implemented in the scientific literature. One of the most recent emerging approaches in this field on the application of Optimization Algorithms (OA) as part of concerted efforts related to Search based Software Engineering agenda. Many OA-based strategies have been implemented as a result. In order to summarize the achievements so far and facilitate future development, the main focus of this paper is, first, to present a critical comparison of adoption optimization algorithms (OA) as a basis of the t-way test suite generation strategy and, second, to propose a new t-way strategy based on Flower Pollination Algorithm (FPA), called Flower Strategy (FS).

The rest of this paper is organized as follows. Section II gives an overview of t-way testing. Detailed reviews of OA strategies are provided in Section III. A briefed dissection is introduced in Section IV. In Section V a new OA-based strategy for t-way testing is proposed. Section VI highlights the early results. Lastly, Section VI gives the conclusion and future work.

II. BACKGROUND ON T-WAY TESTING

In general, any system is made up of a number of components, which interact with each other (called parameters with their associated values). To illustrate the concept of t-way testing, consider a Simple Find Dialog example as given in Fig. 1.

The dialog box takes an input text string and returns all the matching records. Here, this dialog box consists of four inputs (i.e. Text string, Match whole word, case sensitivity and direction of the search). By ignoring illegal values for simplicity; the values for each parameter are as the follows: “Find What” text string: uppercase / lowercase / mixed, “Match whole word only”: match / do not match, “Match case”: match / do not match and “Direction”: Up / Down. In order to test this system exhaustively, there is a need for 24 test cases.

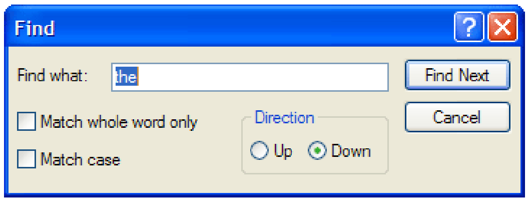

Fig. 1: Simple Find Dialog Example

By using two-way testing, there are 30 pairs that need to be covered. By any greedy strategy (e.g. FA), the first test case can cover 6 pairs. In fact, it can be shown that all the 24 pairs can be covered by 6 test cases.

The interesting property of t-way test, is that any *t* columns contain all possible combinations of these columns, occur somewhere in any order (see Fig. 2).

A	B	C	D
0	0	1	1
0	1	0	0
1	0	1	0
1	1	0	1
2	1	1	0
2	0	0	1

Fig. 2. Two-way test suite generation for Find Dialog example, the only pairs of AB, AC, and CD are highlighted

III. RELATED WORKS

This section is intended to provide an overview of the existing works for constructing t-way test suite. The existing approaches can be classified into two main categories: algebraic approaches, and computational approaches [9]. In the Algebraic Approaches, test data sets are constructed without enumerate any combinations. Hence, the computations associated with algebraic approaches tend to be lightweight. On the negative note, the aforementioned approaches are often restricted to small configurations. Strategies adopting algebraic approaches include orthogonal Latin squares (OLS), CA, MCA and TConfig. Computational approaches use a greedy algorithm to construct the test cases. Each step tries to cover as many combinations as possible uncovered combinations. Generating test set is accomplished by either using one-test-at-a-time strategy (OTAT) or one-parameter-at-a-time strategy (OPAT).

OTAT based strategies build one complete test case per iteration and checks if this test case is the best test case to cover the most uncovered interaction. The same procedure is repeated until all the combinations are covered. In the literature, there are many strategies and tools have developed uses OTAT techniques such as AETG [10], TConfig [11], Jenny [12], and WHITCH [13].

One-parameter-at-a-time (OPAT) strategy starts by building a completed test suite for the first two parameters, or the smallest number of components, then extends horizontal by adding one parameter per iteration, and sometimes, extends vertically until all the parameters is covered. The most well-known strategy in OPAT is in-parameter-order (IPO) strategy [7]. On the basis of IPO strategy, many improvements of IPO have been proposed such as IPOG [14], IPOG-D [6], IPOF and IPAD2 [15] to obtain optimal test size and fast execution time.

Optimization Algorithm (OA) based strategies are one of the most emerging technologies for the last 20 years. The next section provides a detailed review of the existing OA-based strategies for t-way testing. In doing so, this paper highlights and analyses the strengths and the limitations for each OA based strategy. In order to have objective assessment, the analysis focus on the tuning issues, parameter setting requirements as well as the search procedure in each strategy and how each algorithm balances between local intensification and global diversification.

A. Hill Climbing

HC, or local search, is a single-solution based algorithm because it starts finding a solution from one position in the search space, therefore, HC is a good for local optimum problems. As Fig. 3a indicates, HC perform well on problems with only one peak but with other problems that encompass many peaks, HC fails to find global optimal solution [16]. On a positive note, HC is a parameter free algorithm that does not require any tuning of tis parameter.

As a t-way strategy, HC randomly starts finding a test case *x* from one position in the search space and then select any new test case *x',* from *x,* that improves the current solution [17]. Study conducted by Cohen show that HC and SA produce similar results in many cases. However, there are cases where HC fail to find optimal test suite size. To increase the chance of finding a good solution, it may be necessary to repeat the algorithm many time, each time with different initial configurations [8].

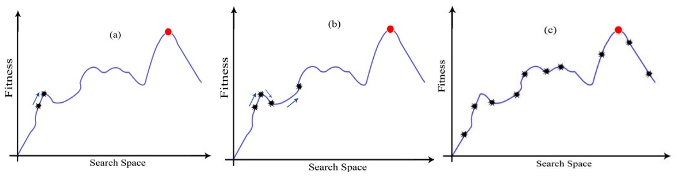

Fig. 3. (a) Hill climbing problem. (b). Simulated Annealing and Tabu Search problem, (c) GA, PSO and HS problem

B. Simulated Annealing

SA is a similar to HC, but SA allows worsening moves with decreasing probability to avoid getting stuck in a local minimum solution as shown in Fig. 3b. The acceptance probability function *P(e, e', T)* decides to move to the new solution x′ or stay in current solution x (where *e= E(x)* is the energy of current solution x, *e'= E(x')* is new candidate solution *x* and *T* controlling temperature) [18]. In order to construct the t-way test suite, SA has been implemented in[8, 19]. The algorithm used randomized searches to generating a test suite by finding the best test case per iteration until cover all interaction elements [20].

As a t-way strategy, SA starts from one position in the search space and then employ local/neighborhood search in a local region (see Fig. 3b). SA provides reasonable balance between local intensification and global diversification by accepting any solution that increases the fitness and sometime accepts a solution with lower fitness. SA needs few parameters (e.g. initial temperature, and cooling rate) [21]. Comparative experiments for t-way test generation between SA, TS and GA demonstrate that SA performs better than TS and GA [8]. On a negative note, apart from being a single solution algorithm, SA depends heavily on the neighborhood structure (but there is no standard structure) to get the most optimal solution [16].

C. Tabu Search

Tabu Search is proposed by Glover for solving combinatorial optimization problems [22]. Similar to SA, TS is a single solution based algorithm that adopts neighborhood structure to perform the search. In Tabu search, worsening moves can be accepted if no improving move is available. TS exploits adaptive forms of memory structures (termed Tabulist) to remember the visited solution (i.e. so that the algorithm does not consider the same possibility repeatedly) [23, 24]. Concerning t-way testing, Stardom [24] point out that TS is effective for constructing test suite and is able to outperform GA. Similar results can be obtain from the work of Nurmela [25].

D. *Genetic Algorithm*

Genetic algorithm is a population-based algorithm that mimics the natural selection process. The main advantage of GA over HC, TS, and SA is the fact that it is not usually get stuck into local optima [26]. To ensure that the solution space is adequately explored, GA preserves the population diverse solutions during the evolution. Hence a proper tuning for GA parameters, (i.e. GA requires the tuning of mutation rate, Crossover rate, number of iteration and population size), lead to good balance between local intensification and global diversification (i.e. genetic population diversity and selective pressure) [27, 28]. Roeva, Fidanova et al. observe that optimal population size is 100 chromosomes for 200 generations [29]. Other studies by Schaffer et al.[30] and Grefenstette [31] suggest that 30 chromosomes are sufficient in many cases.

GA can be considered as an early works in adopting population-based search algorithms for constructing t-way test suite[32]. Therefore, GA starts finding a solution from many positions as Fig. 3c show. In GA, each chromosome represents a test case. The population of chromosome is subjected to repeated cycles of process: (1) Selection: GA selects two chromosomes randomly and then a copy of the winner is deposited mating pool. (2) Crossover: is taking two chromosomes and produce a child by exchanges between the two chromosomes values with probability. (3) Mutation process replaces the value of chromosomes randomly. The fitness function with the best chromosomes is selected at each generation cycle, and added to the final test suite [33].

E. *Ant Colony Algorithm*

Ant Colony algorithm (ACA) mimics the behavior of colonies of ants for finding food paths. ACA use probabilistic technique to control search operation via pheromone trails. In general, the key points of ACA are pheromone update and edge selection. Edge selection in ACA based on probability which is controlled by a number of parameters such as amount of pheromone (□), trail level (□), pheromone coefficient (α) to controls the amount of pheromone □, and heuristic coefficient (β) to control the influence of trail level value □. Pheromone is updated when ant travel from one node to another based on pheromone evaporation coefficient (ρ) and pheromone deposit [32, 34].

For t-way test generation, the places of food represent the parameter and the food represents the value of the parameter [32]. Each test case represents the quality of the paths to the food. The paths to the food are evaluated based on the quantity of pheromones which is reinforced by the ants. Over time, the density gets higher for some paths which are chosen by many ants. By comparison, the best path is selected to be added to final test cases.

F. *Particle Swarm Optimization*

Particle Swarm Optimization (PSO) is population-based algorithm, which is based on swarm movement (flocking) behavior of bird or fish [35]. PSO requires 5 control parameters consisting of cognitive parameters (C1 and C2), inertial weight as well as population size and iteration size [36]. A study conducted by Pedersen demonstrates that the basic of PSO can achieve a good performance if its parameters are properly tuned [37].

Concerning t-way testing, each test case is represented as one particle [9, 36, 38] in PSO. The algorithm starts by initializing the swarms search space and velocity of each particle. At each generation, the velocities are updated according to the best test case as the particle moves around the search space, and make a move to the new test case. The particle continues its motion until termination criteria is met. Here, the best test case is selected in each iteration cycle and is added to final test suite.

G. *Harmony Search*

Harmony Search (HS) is population-based algorithm, mimics the improvisation process of skilled musicians [39]. To explore search space, HS uses a probabilistic-gradient to select the current solution, and adopts mathematical equations to move to better solution. HS requires 4 control parameters namely harmony memory size (HMS), Improvisation/Iteration, pitch adjustment rate (par) and harmony memory consideration rate (HMCR) [40].

Concerning t-way testing, HS has been successfully adopted in harmony search algorithm-based strategy (HSS). The core part of the HSS is improvising a new harmony based on the HMCR. By doing so, the new test case can either be chosen from harmony memory (HM) or randomly. Here, if the new value is coming from HM, the second probability based on pitch adjusting rat (PAR) determines whether or not to improve the current solution [41].

H. *Cuckoo Search*

Cuckoo Search (CS) is a population based algorithm inspired from brood parasitic behavior for some birds such as the Ani and Guira cuckoos [42]. CS provides optimal balance between local intensification and global diversification by intensifies the search for solutions in the neighborhood of incumbent solution as well as explore all the search space efficiently through the use of lévy flights [43] (i.e. as a chaotic search of values). CS has only one parameter *pa* that needs to be tuned, less than the number of parameters in PSO and GA [44].

For t-way test generation, the algorithm begins by generating an initial population of nest. Each nest represents a test case. In each generation of the algorithm, there are two operations are performed. First, generate a new nest by performing a lévy flight, and then the new nest is evaluated and replaces the current nest if it obtains a better evaluation. The second part of the algorithm discovers and removes the worse nests with probability *pa* [45].

IV. Discussion

In the previous section, we reviewed the current literature of the existing strategies that adopt OA algorithms such as HC, SA, TS, GA, ACA, PSO, HS, and CS. In a critical manner, we highlighted and analyzed the strengths and the limitations for each strategy. Here are some observations that we can draw from our analysis:

- No single OA-based t-way strategy has strong dominance over the others in terms of obtaining the optimal test suite size. As such, the search of new OA implementation is still relevant particularly involving newly developed algorithms.
- There is still lack of work on hybridization of OA as a t-way strategy. With hybridization of two or more algorithms, the intensification and diversification of the main OA algorithm of interest can be improved, that is, by exploiting the search operator of the other algorithms.
- Some strategies are computationally heavy and require many parameters to be tuned to achieve performance. If new strategies are to be developed, there is a need for strong consideration on the tuned parameters.

Flower Pollination Algorithm (FPA) is one of the latest OAs developed for solving global optimization problems. FPA is a population based algorithm, inspired by the pollination behavior of flowering plants. FPA has the following advantages [46, 47]:

- Important advantages of FPA are simplicity and flexibility.
- Unlike GA, and PSO, FPA offers lightweight computation relying only on one parameters *Pa*.
- FPA offers balance intensification and diversification of solutions through the adoption of lévy flight. Specifically, lévy flight consists of random walks that are interspersed by long jumps which are heavy tailed according to a power law distribution. In this manner, FPA often can sufficiently explore regions of interests.

In addition to the aforementioned advantages, FPA has yet to be adopted for t-way testing. Thus, we propose a new testing strategy for generating t-way test suite based on FPA, called Flower Strategy (FS).

V. Flower Strategy

This section is intended to give a brief description for our proposed, Flower Strategy (FS). FS is a new t-way strategy for test suite generation based on FPA.

Mathematically, FPA can be represented as two core parties: Global Pollination step and Local Pollination step. In global pollination, the flower pollen is transferred by pollinators such as insects, over a long distance using lévy flight (see equation 1). Local pollination transfer pollen from male parts to female parts in the same flower (see equation 2).

$$x_i^{(t+1)} = x_i^{(t)} + \gamma L\acute{e}vy\,(\lambda)\big(x_i^{(t)} - gbest\big) \qquad (1)$$

$$x_i^{(t+1)} = x_i^{(t)} + \epsilon(x_j^{(t)} - x_k^{(t)}) \qquad (2)$$

In FS, each test case represents as a flower or pollen (i.e. solutions). The fitness function for any pollen is the number of interactions that the pollen can cover, therefore, FS begins by initializing the interaction elements list according to input values (i.e. interaction strength *t*, number of parameters and parameters values). Then, FS randomly generates a population of pollen. During the iteration of the algorithm, each pollen repeatedly subjects to generating new pollen phrase by performing the global pollination or local pollination. The new pollens will be chosen as current pollens if the new pollens are better than current pollens. The generation of new pollen is repeated until the maximum number of improvement has been satisfied. Here, the current best pollen is added to final test suite, and covered interaction elements are deleted from the interaction elements list. This cycle continues all the interaction elements are covered. Fig. 4 describes pseudo code of Flower Strategy.

```
Input: P: Parameter number n, and
       V:  set of values for each feature V = [$v_0$ ..$v_j$];
Output: Final test suite List FTS;
  Let FTS be a set of candidate tests;
  Generate all possible interactions elements IEL based on P and V
  Generate initial population of pollens randomly
  while IEL is not empty do
    while t <MaxGeneration  or  stop criterion do
      for i = 1 : n (all n pollens in the population)
        if ( rand < pa )
          Global pollination via $x_t^{i+1} = x_t^i + \mathrm{L}(\mathrm{gbest} - x_t^i)$
        Else
          Do local pollination via $x_t^{i+1} = x_t^i + \square(x_t^j - x_t^k)$
        End if
        Evaluate new solutions
        If new solutions are better, update
        them in the population
      End for
      Find the current best solution gbest
    End while
    Add the best test case, gbest, into FTS.
    Remove covered interactions elements from IEL.
  End while
End-Procedure
```

Fig. 4. Flower Strategy pseudo code

VI. Early Results

In order to evaluate the performance of FS against OA-based strategies, early results of FS is compared with results of existing OA as published in [9, 41]. In this comparison, different systems configurations have been used (see Table 1).

TABLE 1 : Comparison with Existing OA-Based Strategies

t	Systems	SA	ACA	GA	PSO	HSS	CS	FS
2	3^3	9	9	9	9	9	9	9*
2	3^{13}	16	17	17	17	18	20	17
2	10^{10}	NA	159	157	NA	155	NA	153*
2	15^{10}	NA	NA	NA	NA	342	NA	344*
2	5^{10}	NA	NA	NA	45	43	NA	42*
3	3^6	33	33	33	42	39	43	42
3	4^6	64	64	64	102	70	105	101
3	5^6	152	125	125	NA	199	NA	194
3	6^6	300	330	331	338	336	350	333
3	5^7	201	218	218	229	236	233	219
3	10^6	1508	1501	1473	1506	1505	NA	1470*
2	$5^1\,3^8\,2^2$	15	16	15	NA	20	21	20
3	$5^2\,4^2\,3^2$	100	106	108	NA	120	NA	117
4	2^{10}	NA	NA	NA	37	37	36	36*
5	2^{10}	NA	NA	NA	82	81	79	75*
9	2^{10}	NA	NA	NA	NA	512	NA	571
10	2^{10}	NA	NA	NA	NA	1024	NA	1024*

y^x: means that this system has x parameters, each parameter with y values.

In our comparison, FS parameters are set at *Pa* = 0.8, pollen population size = 30 and maximum number of improvement = 500. Table 1 highlights the results. The cell marked as NA indicates "Not Available results".

From the initial results in Table 1, FS has the potential to outperform the existing strategies. In fact, in the current experiments, FS is able to produce the optimal test size in 8 cases (as shown in shaded cells), even when FS is not the best, the results are still within reasonable values.

VII. Conclusion and Future Work

Summing up, we have given a survey on the recent OA-based strategies adopted for t-way testing. The survey highlights and analyse the strengths and the limitations for each strategy. As a result, a new t-way strategy, called Flower Strategy (FS), is proposed. The new strategy uses Flower Pollination Algorithm as a core implementation. Initial results show that FS is able to outperform some of existing OA-based strategies.

As part of the future work, we plan to improve the design of FS in two directions. Firstly, we intend to address high interaction parameters. Secondly, we plan to hybridize FPA with another OA to improve its overall search capabilities.

Acknowledgment

The work reported in this paper is funded by MOSTI eScience fund for the project titled: Constraints T-Way Testing Strategy with Modified Condition/Decision Coverage from the Ministry of Science, Technology, and Innovation, Malaysia. We thank MOSTI for the contribution and support. Mr. Abdullah B. Nasser is the recipient of the Graduate Research Scheme (GRS) from Universiti Malaysia Pahang.

References

[1] A. Arcuri and L. Briand, "Formal analysis of the probability of interaction fault detection using random testing," *Software Engineering, IEEE Transactions on,* vol. 38, pp. 1088-1099, 2012.

[2] I. Burnstein, *Practical software testing: a process-oriented approach*. New York: Springer Inc, 2003.

[3] M. F. Klaib, S. Muthuraman, and A. Noraziah, "A Parallel Tree Based Strategy for T-Way Combinatorial Interaction Testing," in *Software Engineering and Computer Systems*, ed: Springer, 2011, pp. 91-98.

[4] D. R. Kuhn, D. R. Wallace, and J. AM Gallo, "Software fault interactions and implications for software testing," *Software Engineering, IEEE Transactions on,* vol. 30, pp. 418-421, 2004.

[5] R. Brownlie, J. Prowse, and M. S. Phadke, "Robust Testing of AT&T PMX/StarMAIL Using Oats," *AT&T Technical Journal,* vol. 71, pp. 41-47, 1992.

[6] Y. Lei, R. Kacker, D. R. Kuhn, V. Okun, and J. Lawrence, "IPOG/IPOG‐D: efficient test generation for multi‐way combinatorial testing," *Software Testing, Verification and Reliability,* vol. 18, pp. 125-148, 2008.

[7] Y. Lei and K.-C. Tai, "In-parameter-order: A test generation strategy for pairwise testing," in *Proceedings of The 3rd IEEE International Symposium on HighAssurance Systems Engineering*, 1998, pp. 254-261.

[8] C. J. Colbourn, M. B. Cohen, and R. Turban, "A deterministic density algorithm for pairwise interaction coverage," in *IASTED Conf. on Software Engineering*, 2004, pp. 345-352.

[9] 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, pp. 431-452, 2012.

[10] 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, pp. 437-444, 1997.

[11] A. Williams. TConfig download page [Online]. Available: http://www.site.uottawa.ca/~awilliam/ [Accessed 23 Dec 2014]

[12] B. Jenkins, "Jenny Tool download page [Online]," Available : http://www.burtleburtle.net/bob/math. [Accessed 16 Dec 2014], 2003.

[13] A. Hartman, T. Klinger, and L. Raskin, "IBM intelligent test case handler," *Discrete Mathematics,* vol. 284, pp. 149-156, 2010.

[14] Y. Lei, R. Kacker, D. R. Kuhn, V. Okun, and J. Lawrence, "IPOG: A general strategy for t-way software testing," in *14th Annual IEEE International Conference and Workshops on the Engineering of Computer-Based Systems*, 2007, pp. 549-556.

[15] M. Forbes, J. Lawrence, Y. Lei, R. N. Kacker, and D. R. Kuhn, "Refining the in-parameter-order strategy for constructing covering arrays," *Journal of Research of the National Institute of Standards and Technology,* vol. 113, pp. 287-297, 2008.

[16] D. Beasley, R. Martin, and D. Bull, "An overview of genetic algorithms: Part 1. Fundamentals," *University Computing,* vol. 15, pp. 58-58, 1993.

[17] M. B. Cohen, P. B. Gibbons, W. B. Mugridge, and C. J. Colbourn, "Constructing test suites for interaction testing," in *Software Engineering, 2003. Proceedings. 25th International Conference on*, 2003, pp. 38-48.

[18] E. Aarts and J. Korst, "Simulated annealing and Boltzmann machines: a stochastic approach to combinatorial optimization and neural computing," 1988.

[19] M. Patil and P. Nikumbh, "Pair-wise testing using simulated annealing," *Procedia Technology,* vol. 4, pp. 778-782, 2012.

[20] K. J. Nurmela and P. R. Östergård, *Constructing covering designs by simulated annealing*: Helsinki University, Digital Systems Laboratory, 1993.

[21] F. Busetti, "Simulated annealing overview," *World Wide Web URL www. geocities. com/francorbusetti/saweb. pdf,* 2003.

[22] F. Glover, "Tabu search-part I," *ORSA Journal on computing,* vol. 1, pp. 190-206, 1989.

[23] F. Glover and M. Laguna, *Tabu search*: Springer, 1999.

[24] J. Stardom, "Metaheuristics and the search for covering and packing arrays," Trent University, 2001.

[25] K. J. Nurmela, "Upper bounds for covering arrays by tabu search," *Discrete applied mathematics,* vol. 138, pp. 143-152, 2004.

[26] H. Ishibuchi and T. Murata, "Multi-objective genetic local search for minimizing the number of fuzzy rules for pattern classification problems," in *Fuzzy Systems Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*, 1998, pp. 1100-1105.

[27] D. Gupta and S. Ghafir, "An overview of methods maintaining diversity in genetic algorithms," *International Journal of Emerging Technology and Advanced Engineering,* vol. 2, pp. 56-60, 2012.

[28] L. J. Eshelman and J. D. Schaffer, "Preventing Premature Convergence in Genetic Algorithms by Preventing Incest," in *ICGA*, 1991, pp. 115-122.

[29] O. Roeva, S. Fidanova, and M. Paprzycki, "Influence of the population size on the genetic algorithm performance in case of cultivation process modelling," in *Computer Science and Information Systems (FedCSIS), 2013 Federated Conference on*, 2013, pp. 371-376.

[30] J. D. Schaffer, R. A. Caruana, L. J. Eshelman, and R. Das, "A study of control parameters affecting online performance of genetic algorithms for function optimization," in *Proceedings of the third international conference on Genetic algorithms*, 1989, pp. 51-60.

[31] J. J. Grefenstette, "Optimization of control parameters for genetic algorithms," *Systems, Man and Cybernetics, IEEE Transactions on,* vol. 16, pp. 122-128, 1986.

[32] 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.

[33] C. Nie and H. Leung, "A survey of combinatorial testing," *ACM Computing Surveys (CSUR),* vol. 43, p. 11, 2011.

[34] M. Grindal, J. Offutt, and S. F. Andler, "Combination testing strategies: a survey," *Software Testing, Verification and Reliability,* vol. 15, pp. 167-199, 2005.

[35] H. Huang, A. Hoorfar, and S. Lakhani, "A comparative study of evolutionary programming, genetic algorithms and particle swarm optimization in antenna design," in *Antennas and Propagation Society International Symposium, 2007 IEEE*, 2007, pp. 1609-1612.

[36] B. S. Ahmed, K. Z. Zamli, and C. Lim, "The development of a particle swarm based optimization strategy for pairwise testing," *Journal of Artificial Intelligence,* vol. 4, pp. 156-165, 2011.

[37] M. E. H. Pedersen, "Good parameters for particle swarm optimization," *Hvass Lab., Copenhagen, Denmark, Tech. Rep. HL1001,* 2010.

[38] X. Chen, Q. Gu, J. Qi, and D. Chen, "Applying particle swarm optimization to pairwise testing," in *Computer Software and Applications Conference (COMPSAC), 2010 IEEE 34th Annual*, 2010, pp. 107-116.

[39] Z. W. Geem, J. H. Kim, and G. Loganathan, "A new heuristic optimization algorithm: harmony search," *Simulation,* vol. 76, pp. 60-68, 2001.

[40] Z. W. Geem, "Optimal cost design of water distribution networks using harmony search," *Engineering Optimization,* vol. 38, pp. 259-277, 2006.

[41] 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," *Information and Software Technology,* vol. 54, pp. 553-568, 2012.

[42] X.-S. Yang and S. Deb, "Cuckoo search via Lévy flights," in *Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on*, 2009, pp. 210-214.

[43] X. S. Yang, S. Deb, and S. Fong, "Metaheuristic algorithms: optimal balance of intensification and diversification," 2013.

[44] X.-S. Yang and S. Deb, "Cuckoo search: recent advances and applications," *Neural Computing and Applications,* vol. 24, pp. 169-174, 2014.

[45] 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," *Information and Software Technology,* 2015.

[46] X.-S. Yang, "Flower pollination algorithm for global optimization," in *Unconventional computation and natural computation*, ed: Springer, 2012, pp. 240-249.

[47] X.-S. Yang, M. Karamanoglu, and X. He, "Flower pollination algorithm: a novel approach for multiobjective optimization," *Engineering Optimization,* vol. 46, pp. 1222-1237, 2014.