An elitist-flower pollination-based strategy for constructing sequence and sequence-less t-way test suite

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

Faculty of Computer Systems and Software Engineering,
Universiti Malaysia Pahang,
26300 Kuantan, Pahang, Malaysia
Email: abdullahnasser83@gmail.com
Email: kamalz@ump.edu.my
Email: alsewari@gmail.com
*Corresponding author

Bestoun S. Ahmed

Faculty of Electrical Engineering,
Department of Computer Science,
Czech Technical University,
Karlovo n'am, 13, 121 35, Praha 2, Czech Republic
Email: albeybes@fel.cvut.cz

Abstract: In line with the upcoming of a new field called search-based software engineering (SBSE), many newly developed t-way strategies adopting meta-heuristic algorithms can be seen in the literature for constructing interaction test suite (such as simulated annealing (SA), genetic algorithm (GA), ant colony optimisation algorithm (ACO), particle swarm optimisation (PSO), harmony search (HS) and cuckoo search (CS). Although useful, most of the aforementioned t-way strategies have assumed sequence-less interactions amongst input parameters. In the case of reactive system, such an assumption is invalid as some parameter operations (or events) occur in sequence and hence, creating a possibility of bugs triggered by the order (or sequence) of input parameters. If t-way strategies are to be adopted in such a system, there is also a need to support test data generation based on sequence of interactions. In line with such a need, this paper presents a unified strategy based on the new meta-heuristic algorithm, called the elitist flower pollination algorithm (eFPA), for sequence and sequence-less coverage. Experimental results demonstrate the proposed strategy gives sufficiently competitive results as compared with existing works.

Keywords: t-way testing; flower pollination algorithm; event sequence testing; combinatorial problem; meta-heuristics; optimisation problem.

Reference to this paper should be made as follows: Nasser, A.B., Zamli, K.Z., Alsewari, A.A. and Ahmed, B.S. (2018) 'An elitist-flower pollination-based strategy for constructing sequence and sequence-less t-way test suite', *Int. J. Bio-Inspired Computation*, Vol. 12, No. 2, pp.115–127.

Biographical notes: Abdullah B. Nasser received his BSc degree in Computer Science from the Hodeidah University, Yemen in 2006, and MSc degree in Computer Science from the Universiti Sains Malaysia, Malaysia in 2014. He received his PhD in Computer Science from the Universiti Malaysia Pahang in 2017. Currently, he serves as a Research Assistant in the Universiti Malaysis Pahang focusing on software testing and optimisation algorithm.

Kamal Z. Zamli received his BSc degree in Electrical Engineering from the Worcester Polytechnic Institute, USA in 1992, MSc degree in Real-Time Software Engineering from the Universiti Teknologi Malaysia in 2000 and PhD degree in Software Engineering from the Newcastle University, UK in 2003. His main research interests include search-based software engineering, combinatorial software testing, and computational intelligence.

AbdulRahman A. Alsewari obtained his BSc in Computer Engineering from the Military College of Engineering Baghdad Iraq in 2002, MSc degree in Electronic System Design Engineering from the Universiti Sains Malaysia in 2008 and PhD in Software Engineering from the Universiti Sains Malaysia in 2012. His research interests include combinatorial testing, computational intelligence, optimization algorithms, embedded systems, and image processing.

Copyright © 2018 Inderscience Enterprises Ltd.

Bestoun S. Ahmed obtained his MSc degree from the University Putra Malaysia in 2009, and his PhD degree from the Universiti Sains Malaysia in 2012. He serves as Post-doctoral Researcher in the Swiss AI Lab Istituto Dalle Molle di Studi sull'Intelligenza Artificiale (IDSIA), Switzerland for nearly 1.5 years before joining Czech Technical University in Prague. His main research interests include software testing, search-based software engineering and quality assurance for IoT services.

This paper is a revised and expanded version of a paper entitled 'Sequence and sequence-less t-way test suite generation strategy based on the flower pollination algorithm' presented at IEEE Student Conference on Research and Development (SCOReD 2015), Kuala Lumpur, 13–14 December 2015.

1 Introduction

Recently, a new field of software engineering, termed search-based software engineering (SBSE) has been coined leveraging on the application of meta-heuristics algorithms for solving optimisation problem within software engineering applications (Harman et al., 2009). In the field of interaction testing, many meta-heuristics algorithms have been adopted as the basis of the t-way strategy implementation (where t indicates the interaction strength). Sampling of the test cases from a large set of combinatorial values based on the specified t-wise interaction, many existing works have been developed in the literature including simulated annealing (SA) (Stardom, 2001), genetic algorithm (GA) (Shiba et al., 2004; Stardom, 2001), ant colony algorithm (ACA) (Shiba et al., 2004), particle swarm optimisation (PSO) (Ahmed et al., 2012), harmony search (HS) (Alsewari and Zamli, 2012) and Cuckoo Search (CS) (Ahmed et al., 2015) respectively.

At a glance, the adoption of the aforementioned meta-heuristic-based strategies appears to be sufficiently effective for obtaining good quality solution as reported in many benchmarking experiments related to t-way testing (Ahmed et al., 2012; Alsewari and Zamli, 2012). Nonetheless, a closer look suggests two subtle limitations.

Firstly, existing meta-heuristic-based strategies assume 'sequence-less' interactions between input parameters. However, this assumption is not always true. In the real world, many systems show sequence dependencies, hence, their faults may not be exposed if the sequences of inputs are not considered. For these reasons, the adoption of meta-heuristic algorithm as the basis for t-way strategy for both sequence and sequence-less t-way testing is seen as a useful endeavour.

Secondly, as the effectiveness of any meta-heuristic-based strategy is largely dependent on their control parameter settings, it is desirable to adopt algorithm with small number of control parameters. Existing work on meta-heuristic-based t-way strategies demand extensive tuning of its main control parameters before optimal solutions can be obtained. For example, Simulated Annealing is overly sensitive on the initial and final temperature as well as its temperature reduction function. GA requires substantial tuning for population size, mutation and cross over rate. The same issue also appears in the case of PSO (Ahmed et al., 2012) which relies on population size, inertia weight, social and cognitive parameters as parameters. Similarly, HS (Geem, 2009) requires tuning of harmony size, harmony memory consideration rate, and pitch adjustment. In many cases, improper tuning for all of these control parameters undesirably increases computational efforts as well as yields sub-optimal solution.

2 Background

2.1 Theoretical background

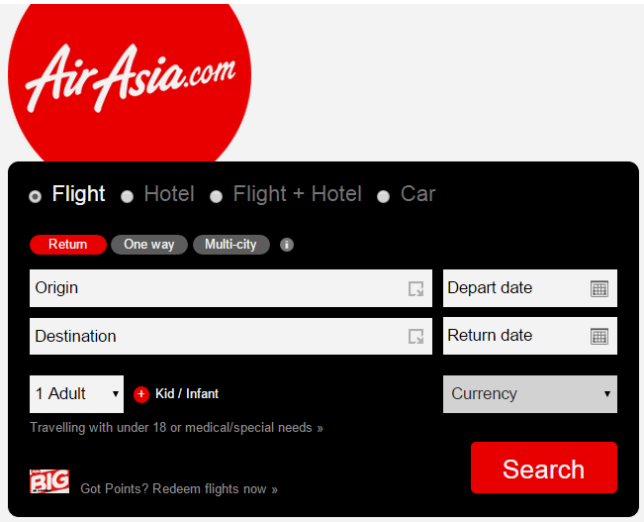
Covering array (CA) is often adopted as a mathematical object to describe and formulate the problems related to t-way testing (Burr and Young, 1998; Yilmaz et al., 2006). In general, any system under test (SUT) consists of different components, which interact with each other, called parameters with their associated values, and a number of events. Throughout this paper, the symbols v, p, t, and e are used to refer to number of parameters, associated levels, interaction strength and number of events respectively. When the number of values (v) is equal for all the parameters (p), then the CA is represented as uniform CA, CA(N; t, v^p). For example, a CA of CA(6; 2, 2^4) consists of six rows of test cases generated from four column (or parameters) with two values each. When the number of parameters are not equal (i.e., each parameters has different number of values), the CA representation takes the mixed coverage array notation of MCA(N; t, $v_1^{p_1} v_2^{p_2} v_3^{p_3} \ldots v_j^{p_j}$). For example, MCA (12; 3, 2^3 3^1) represents a test suite consisting an array of 12 rows of test cases, four columns of parameters with three parameters having two values and one parameter having three values. A sequence CA of e events with t interaction strength is represented as SCA(N; t, e). For example, SCA(8; 3, 6), represents a test suite consisting an array with 8 rows of test cases generated from four events for interaction strength equals to three.

2.2 Problem definition model

To illustrate the t-way testing problem, consider the online ticket booking system for AirAsia airline in Figure 1. This

system is used for four types of bookings: flight, hotel, flight + hotel, or car. To simplify the example, we consider only flight booking system. Flight booking system uses different components, such as journey type, origin location, destination location, number of passenger, number of kid/infant, depart date, and return date.

Figure 1 Air Asia ticket booking website (see online version for colours)



The system consists of seven parameters, each parameter has its associated configurations or values (i.e., the parameter-values of *journey type* are return, one way, or multi-city, the values of origin location are nine origins, the values of destination location are nine destinations, …, and so on). To describe the consecutive sequence of events, the term event is used to set the parameter-values of one parameter or more. For instance, the event 'select origin location' is invoked to set value for *origin location* parameter. Further examples of events include selecting a *depart date*, *destination location*, *return date*, and *clicking a search button*.

The system can be summarised as seven parameters; one parameter with three values, two parameters with nine values, two parameters with one value, two parameters with ten values (as shown in Table 1), and eight events (as shown in Table 2). In this example, exhaustive testing requires 24,300 test cases (i.e., $3 \times 9 \times 9 \times 10 \times 10 \times 1 \times 1$) for testing input parameters, and 40,320 test cases (i.e., $8 != 40{,}320$) for testing sequence events, which is hardly feasible in practice. In this case, we consider the following t-way strategies to systematically minimise the tests:

a *Sequence-less t-way strategy* generates the test suite that covers every t combination of parameter-value at least once irrespective of the parameter order. Considering our running example, the test cases are reduced to 105 test cases from 24,300 tests exhaustively if we consider $t = 2$.

b *Sequence t-way strategy* generates the sequences of events such that every combination of events is covered at least one time with respect to the order. Sequence-based t-way test suite is a set of permutation of m events such that every permutation of t events occurs at least one time. For instance, the sequence of e1, e2, e3, e4 and e5 is covered the three-way sequences of [e1, e2, e3], [e1,e2,e4], [e1,e2,e5], [e1,e3,e4], [e1,e3,e5], [e1,e4,e5], [e2,e3,e4], [e2,e3,e5], [e2,e4,e5] and [e3,e4,e5]. Here, the combinations (e1, e2) and (e2, e1) are not equivalent. In our running example, using three-way for generating sequence test suite require only ten test cases out of 40,320 test candidates.

This paper draws upon these similarities to design a unified t-way strategy for sequence and sequence-less t-way test generation. To the best of our knowledge, this work is the first attempt to address the problem of generating both sequence and sequence-less t-way test suite within the same strategy implementation.

Table 1 Air Asia ticketing system options

Journey type	*Origin location*	*Destination location*	*# passenger*	*# kid*	*Depart date*	*Return date*
One way	Malaysia	Malaysia	0	0	1	1
Return	Indonesia	Indonesia	1	1		
Multi-city	Thailand	Thailand	2	2		
	Singapore	Singapore	3	3		
	Philippines	Philippines	4	4		
	Vietnam	Vietnam	5	5		
	Cambodia	Cambodia	6	6		
	Laos	Laos	7	7		
	China	China	8	8		
			9	9		

Table 2 List of events

Event	*Return date*
e1	Selecting an origin location
e2	Selecting a depart date
e3	Selecting a destination location
e4	Selecting a return date
e5	Selecting number of passenger
e6	Selecting number of kids
e7	Selecting currency
e8	Clicking a search button

3 Related works

This section provides an overview on the existing works for generating a t-way test suite. The literature review of existing works will be grouped into two parts. The first part consists of detailed review of t-way test suite generation strategies. The second part of the literature review provides an overview sequence-based t-way test suite generation strategies.

3.1 T-way test suite generation strategies

Considering the scope of this paper is on the application of SBSE for t-way test suite generation, our review gives specific focus on the strategies that adopt meta-heuristic algorithms (i.e., search-based strategies).

Generally, most of search-based strategies start by generating the search space, which represents all t-way interaction elements. Then, the strategy applies one of the search algorithms to find the test case with highest fitness score. Finally, the covered interactions elements are removed from the search space. This process is repeated until all interaction elements are covered. Otherwise, search process is repeated again.

Hill climbing (HC) is the most basic search algorithm for constructing t-way test suite (Cohen, 2004). HC is a single-solution-based local search algorithm with low computational complexity (and parameter free), however, it is often prone to getting stuck in local minima. Unlike HC, LAHC is a population-based algorithm making late decision on the best solution as a way to avoid local minima. Complementing HC and LAHC, tabu search (TS) has also been successfully adopted for two-way testing (Stardom, 2001). Unlike HC, TS allows movement to poor solution, when the current solution is not improving. TS maintains a short-term memory (i.e., tabu list) of recent solutions that have been visited in the search space to prevent regeneration. TS exhaustively explores the neighbourhood of each potential solution, rather than random as in HC. SA is a similar algorithm to HC but allows the algorithm to move to poor solution, with a decreasing acceptance probability.

GA (Shiba et al., 2004; Stardom, 2001; Sthamer, 1995) and ACO (Shiba et al., 2004) are considered early works in adopting population-based algorithms for constructing t-way test suite. GA starts finding the optional solution from many positions, and then each solution of the population is subjected to repeated cycles of processes (i.e., selection, crossover, and mutation) in order to mimics natural selection of biological evolution. ACO mimics the behaviour of colonies of ants for finding food paths. The main advantage of GA over HC, TS and SA is that GA is not usually get stuck into local optima. Both of GA and ACO include some expensive computations such as crossover and mutation operations in GA and ant search process in ACO (Harman and Jones, 2001).

In recent work, PSO has also been adopted a t-way strategy (Ahmed et al., 2012). PSO relies on a population of particles called a swarm. PSO mimics the swarm behaviour of bird and fish swarm in searching food. Similar to PSO, HS (Alsewari and Zamli, 2012) has also been adopted for t-way test suite generation. Using HS, the t-way generation process follows the analogy of the improvisation process by a skilled musician. To explore the search spaces in an efficient manner, HS uses a probabilistic-gradient to select current solution neighbour for optimal solutions. CS is a population-based algorithm inspired from brood parasitic behaviour of cuckoos. CS has been implemented for t-way test suite generation in Ahmed et al. (2015). CS provides optimal balance between local intensification and global diversification as well as provides the capability to explore all the search space efficiently through the use of Lévy flights (Yang et al., 2014). Similar to GA, CS uses elitism mechanism to ensure that only solutions with higher fitness can proceed to next generation of iteration. Owing its effectiveness to enhance the solution, we have also opted to adopt elitism for our proposed eFPA strategy.

Recently, Zamli et al. (2016) proposes a new hyper-heuristic-based strategy called high level hyper-heuristic (HHH). In HHH, tabu search algorithm serves as the master algorithm (i.e., high level) to control other four low level algorithms (LLH); teaching learning-based optimisation, PSO, cuckoo search algorithm and global neighbourhood algorithm. To ensure high performance, HHH defines a new acceptance mechanism for the selection LLH algorithm, relies on three operations (i.e., diversification, intensification and improvement). Further experiments have been done in Zamli et al. (2017b) with new acceptance mechanism based on fuzzy inference system and new LLH operations (i.e., GA's crossover search operator, TLBO's learning search operator, FPA's global Pollination, and Jaya algorithm's search operator).

Building from earlier approach, (Zamli et al., 2017a) develop a new strategy, called adaptive teaching learning-based optimisation (ALTBO). ATLBO improves the performance of standard TLBO resulting from a good balance between diversification and intensification through the adoption of fuzzy inference rules.

3.2 Sequence t-way test suite generation strategies

Contrary to sequence-less t-way test suite generation strategies, there are much little existing works that have been published on sequence-based t-way test suite

generation. Although, sequence-based strategies have a long history, most of existing works focus on sequences derived from state transitions, program control flow graphs, and syntax expressions (Offutt et al., 2003). The most closed works to this paper are combinatorial testing and applications of CAs such as greedy-based strategies, e.g., (Chee et al., 2013; et al., 2012) or meta-heuristic-based strategies [e.g., BA (Zabil et al., 2012)].

Sequence of events in the CA has been considered as a combinatorial problem first by Kuhn et al. (2012). In their algorithm, called t-seq, a set of n events is numbered from 0 to n. Here, a small set of tests, which must cover every t-sequence at least once has to be generated. In t-seq algorithm, large number of test cases is generated, then, each test case is scored based on the number of previously uncovered sequences. Then, using a greedy algorithm, the highest score test case covering the maximum number of t-way sequence combination is chosen (Kuhn et al., 2012).

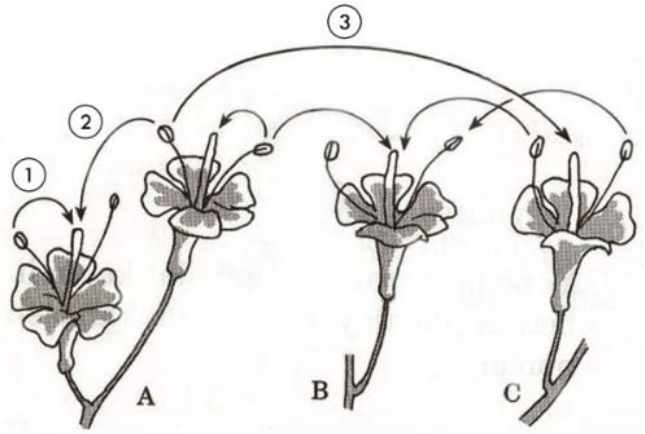
Bee algorithm is adopted in sequence t-way test suite generation strategy (Zabil et al., 2012). BA is a population-based algorithm inspired from behaviour of honey bee colonies to find food. In the strategy, t-way sequence combinations represent the food source and bees will be employed to search for the best source of food. The population of test case is subjected to repeated cycles of searching to find the best test case. This searching process is repeated till t-way sequence combinations are covered. Rahman et al. (2015) proposed the event driven input sequence testing (EDIST-SA), by using the SA algorithm. However, EDIST-SA is designed to support only for t = 2 and t = 3. Ahmad et al. (2016) develops sequence covering array generator (SCAT), which groups similar quality candidates into a pool. Effectively, SCAT delays the selection of the best candidate till the very end for each cycle.

Chee et al. (2013) exploit upper bound (U) and upper bound reversal (Ur) in order to select one permutation from t-scrambling sets of permutations. Consisting of two selections of best candidates, the first one is based on the greedy algorithm covering the most uncovered t-sequence. The second one also uses the greedy algorithm but adds the reverse of the current best test case in the final test suite. Erdem et al. (2011) adopts the concept of answer set programming (ASP), essentially a finite set of rules based on logic programming for generating sequence t-way test suite. In this approach, SCA is computed using answer-set programming where search problems are encoded using ASP, such that the solutions of a problem correspond to a set of answer models.

4 Flower pollination algorithm

Flower pollination algorithm is a recently developed meta-heuristic algorithm based on the flower pollination process. To be specific, pollination process involves transferring pollen grains from male part of flower to ovules borne in female part via pollinators such as birds, butterflies, bees, and bats. According to mechanisms of pollen transfer, pollination can take two types: biotic and abiotic. Biotic pollination refers to the transfer pollen via pollinators (i.e., involving insects or other animals). Abiotic pollination, on the other hand, does not require any pollinators to transfer pollen (i.e., using non-animal vectors, such wind and water responsible). Often, pollination can be accomplished by self-pollination or cross-pollination as shown in Figure 2. Self-pollination occurs when pollen is transferred from male parts to female parts of the same flower or to another flower of same plant. Cross-pollination refers to the case when pollen is transferred from flower of one plant to flower of another plant. Based on the pollination process described earlier, Yang et al. (2014) develops a new meta-heuristic algorithm, called the flower pollination algorithm (FPA) (Yang, 2012).

Figure 2 Flower pollination methods, (1) self pollination, (2) pollination from same plant but different flower, (3) pollination from different plant



Referring to Figure 2, FPA can be represented mathematically as two key steps: global pollination step and local pollination step. Global pollination step in FPA is represented in transferring the flower pollens, by pollinators such as insects, over a long distance. This guarantee the fittest pollens with high quality will carry over to the next reproduction. Exploiting the Lévy flight motion, global pollination step can be represented by equation (1).

$$x_i^{(t+1)} = x_i^{(t)} + \gamma \text{Lévy}(\lambda)\left(X_t - \text{gbest}\right) \tag{1}$$

γ is step size to control the Lévy flight, $\gamma > 0$. Lévy (λ) is a function to produce step size. The Lévy flight essentially is a random walk interspersed by long jumps with various distance steps. Lévy (λ) represents the flight behaviour of pollinators based on the Lévy distribution:

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

λ is distribution factor and $\Gamma(\lambda)$ is standard gamma function. This distribution is valid for large steps $s > 0$.

Flower pollination process can also involve self-pollination where pollen's flower can successfully pollinate the same flower or another flower in the same

plant or with another flower in different plants (i.e., termed local pollination step). Local pollination step is formulated by following equation:

$$x_i^{(t+1)} = x_i^{(t)} + \epsilon\left(x_j^{(t)} - x_k^{(t)}\right) \quad (3)$$

where $x_j^{(t)}$ and $x_k^{(t)}$ are pollens selected randomly from different flowers, ϵ is random number obeying uniform distribution in [0,1]. Here, a switch condition p can be used to alternate between common global pollination and intensive local pollination. FPA can be summarised as shown in Figure 3.

Figure 3 Pseudocode of flower pollination algorithm

```
1.  Objective function f(x), x = (x_1, ..., x_d);
2.  Initialize a population of n pollen with random solutions.
3.  Find the best solution gbest in the initial population
4.  Define a switch probability pa ∈ [0, 1]
5.  while (t <MaxGeneration)
6.    for i = 1 : n (all n flowers in the population)
7.      if ( rand < pa )
8.            Global pollination via x_t^{i+1} = x_t^i + L(g^best − x_t^i)
9.      else
10.           Randomly choose j and k among all the solutions
11.           Do local pollination via x_t^{i+1} = x_t^i + ϱ(x_t^j − x_t^k)
12.     End if
13.     Evaluate new solutions
14.     If new solutions are better, update them.
15.   End for
16. Find the current best solution gbest
17. End while
```

Source: Yang (2012)

5 Flower strategy test generation

As part of our work, we develop a unified elitist flower pollination algorithm (eFPA) strategy, for sequence and sequence-less t-way test generation problems. The complete pseudo-code of eFPA algorithm can be represented in Figure 4.

In line 1, eFPA generates interaction elements list (L) based on the *Interaction_Type* variable selection (i.e., sequence-less or sequence). L represents all t interaction possibilities of combinations between input values.

In case of sequence-less interaction elements, for a set of parameters and their values v, all t-combinations of P that met the interaction strength requirement are generated. Then, the interaction elements are generated based on these combinations. Consider a system CA with CA (N; 2, 2^3 3^1). The t-way combinations of this system are P_1P_2, P_1P_3, P1P4, P_2P_3, and P_2, P_4, where P = [P_1, P_2, P_3, P_4]. For our running example, P_1, P_2, and P_3 having two values (i.e., 0 and 1), and P_4 has three values (0, 1, and 2). For each t-combination, all possible combinations of corresponding parameters' values are added in the L. For instance, P_1P_2 has 2 × 2 possible interaction elements (i.e., 0:0, 0:1, 1:0, and 1:1), P_1P_4 has 2 × 3 possible interaction elements (i.e., 0:0, 0:1, 1:0, 1:1, 2:0, and 2:1). The total interaction elements are (2 × 2) + (2 × 2) + (2 × 3) + (2 × 2) + (2 × 3) = 24.

Figure 4 Pseudo-code of EFPA

```
Input:
    Interaction_Type: [sequence-less, sequence]
    Elitism: Boolean
    (P, v, n):  Set of parameters P, its parameter-value v and number
  of events.
    t:  interaction strength.
  Output:
    Final test suite List FTS;
  Begin
1.  Initialize interaction elements list L.
2.  Initialize pollen_size, sp, and ep.
3.  Generate population of pollens, M={ M_1, M_2 ... M_pollen_size
4.  While L is not empty
5.    While t < MaxGeneration
6.     For i = 1 : n (all n pollens in the population)
7.       If ( rand < sp )
8.         Global pollination via x_t^{i+1} = x_t^i + L(gbest − x_t^i)
9.       Else
10.        Randomly choose j and k among all the solutions
11.        Do local pollination via x_t^{i+1} = x_t^i + ϱ(x_t^j − x_t^k)
12.      End if
13.      Evaluate new solutions
14.      If new solutions are better
15.      update them in the population
16.   End for
17. // decision  based on elitism
18.  If (Elitism== true)
19.    For i = 1 : n (all n pollens in the population)
20.      If random < 0.25 then
21.       Get the worst pollen index, say j.
22.       Replace j by randomly new solution;
23.      End if
24.    End For
25.  End if
26.  Find the current best solution gbest
27.  End while
28.  Add the best test case, gbest, into FTS.
29.  Remove covered interactions elements from L.
30. End while
31. End
```

Concerning the generation of sequenced interaction elements, every t-way combination of events is covered at least once with respect to the particular order. For given n events (i.e., 1, 2 … n), we generate sequence interaction elements list based on t-combinations set. For each combination, all possible permutations of corresponding events are added into L. For instance, for SCA(N; 4, 3), the all three-combinations set is (i.e., $P_1P_2P_3$, $P_1P_2P_4$, $P_1P_3P_4$, $P_2P_3P_4$). Here, the combination P1P2P3 has 6 permutations (i.e., 1:2:3", 1:3:2, 2:1:3, 2:3:1, 3:1:2, and 3:2:1), while $P_1P_2P_4$ six permutations (i.e. 0:1:3, 0:3:1, 1:0:3, 1:3:0, 3:0:1, and 3:1:0). The total interaction elements are 3! +3! +3! +3! = 24.

Line 2 marks the start of the test suite generation algorithm. Here, eFPA specifies initial values for population size *pollen_size*, switch probability (*sp*), elitism probability (*ep*) and stopping criteria (i.e., maximum iteration for improvement). Line 3 initialises random population of pollens $M = [M_1, M_2 \ldots M_{pollen\_size}]$, $M_i = (x_1^i, x_2^i, \ldots, x_{p-1}^i, x_p^i)$, and calculates the pollen's fitness. Here, the pollen's fitness is the number of covered interaction

elements by the pollen. In lines 4–24, eFPA subjects M to repeated cycles of the search process in order to cover maximum number of interactions in L. Based on the value of sp, in line 7, eFPA selects one of the two core operations of FPA. Global pollination (in line 8) permits the adoption of Lévy flight to produce new potential pollen, $x^{new} = (x_1^{new}, x_2^{new}, ..., x_{n-1}^{new}, x_n^{new})$ using equation (1). Using the local pollination (in lines 10–11), eFPA selects two existing pollens, $x_j^{(t)}$ and $x_k^{(t)}$ randomly from the population to generate now pollen using equation (2). In lines 13 and 14, eFPA evaluates the fitness of the generated pollens. If the new pollen fitness is better than current one, the new pollen replaces the current pollen.

Lines 18–23 introduce the elitism into eFPA if the variable *elitism* is set to be true. Here, a fraction of worse pollens are replaced by new pollens (lines 20 and 21). Elitism ensures that only pollens with high fitness are passed to next generation. For sampling, the value of elitism probability (*ep*) depends on the user setting.

The search process is repeated until the maximum number of improvement has been satisfied (in lines 5–27). Then, the eFPA adds the best test case into the final test suite (line 27), and then the covered interactions elements are removed from the interaction list (line 28). After that, interaction elements list is checked, once all interaction elements are covered (i.e., the interaction list is empty), the iteration stops. Otherwise, the overall iteration is repeated.

6 Evaluation of eFPA

The evaluation of eFPA is divided into two parts. The first part evaluates eFPA with the original FPA in terms of convergence rate. The second part deals with comparative studies with existing sequence and sequence-less t-way strategies, in terms of the generated test suite size. To perform the experiments, we adopt the Intel (R) Core (TM) i7-3770 CPU@ 3.40 GHz – 3.40 GHz, 4GB of RAM, Windows 7 Professional, and 32-bit operating system. For parameters adjustment, we have adopted Pa = 0.8, maximum iteration = 700, pollen size = 50 and switch probability = 0.25, based on the experiments recommended values by Ahmed et al. (2015), Nasser et al. (2015) and Yang (2012). Each cell displays the minimum test suite size for each the existing strategies where the best (marked as shaded cell) and the average test suite size is recorded. Cells marked as NA denote unavailability of the results in the literature. To allow meaningful comparison between each strategy (with both FPA and eFPA), we define the following relationship:

$$\Delta = \beta - \alpha \tag{4}$$

where

β our solution (including FPA and eFPA)

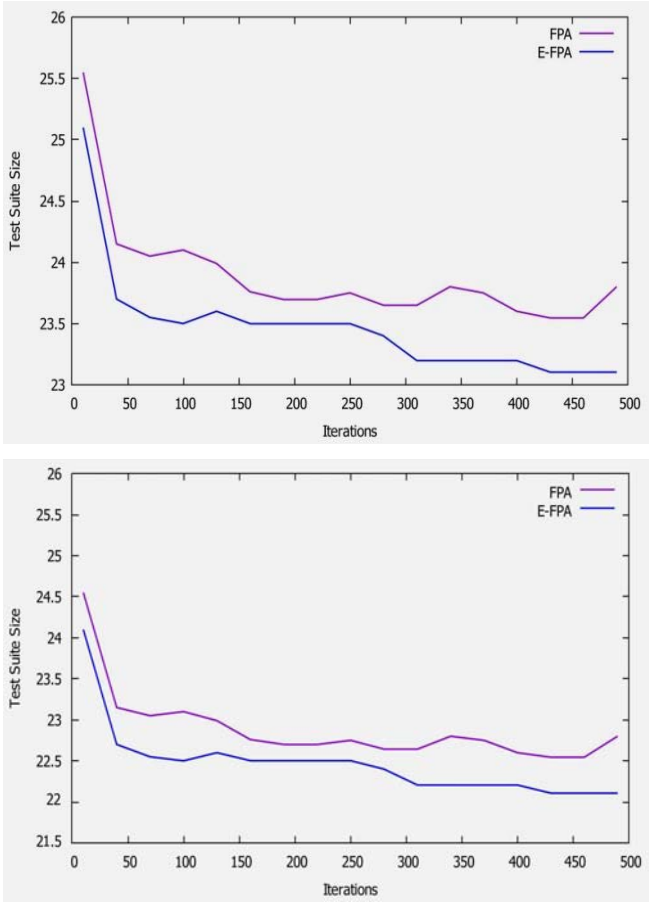
α the best known solution from existing strategies.

If $\Delta < 0$, our solution is better than the known best solution. By the same token, if $\Delta > 0$, then our solution is poorer than that of the known best. Similarly, when $\Delta = 0$, our solution matches with the known best solution.

6.1 Convergence rate comparison eFPA versus FPA

The convergence rates of the proposed strategy with elitism (i.e., referred to as eFPA) and without elitism (i.e., referred to as FPA) are compared using the sequence-less t-way CA(N; 2, 105), and the sequence t-way SCA(N; 20, 3). FPA and eFPA are executed with different values of the iteration (i.e., 5, 10, 20, 30, 40, 50, 100, 200, 300, 400, and 500). The average values of 20 runs for the two CAs are shown in Figure 5.

Figure 5 Convergence rate of eFPA versus FPA for CA(N; 2, 105), and SCA(N; 20, 3) (see online version for colours)



Referring to figure, eFPA outperforms FPA for the two sequences-less and sequence t-way problems. In this case, introducing elitism into eFPA increases the randomisation of population. Hence, the quality of solutions for eFPA and the convergence rate improve.

6.2 Benchmarking with existing t-way strategies

Our benchmark consists of sequence-less and sequence-based experiments as follows.

6.2.1 Benchmarking with sequence-less t-way strategies

Our sequence-less benchmark comparison is based on the following experiments.

- *Experiment 1*: comparison with existing strategies as published in (Ahmed et al., 2015; Alsewari and Zamli, 2012) for 13 different configurations.
- *Experiment 2*: comparison with existing strategies for CA (N; t, 2^{10}), t varied from 2 to 10.
- *Experiment 3*: comparison with existing strategies for CA(N; 4, v^{10}), v varied from 2 to 7.
- *Experiment 4*: comparison with existing strategies for CA(N; 4, 5^{p}), p varied from 5 to 10

The results in Table 3 deals with small interaction strength (i.e., t = 2 and t = 3). Referring to the Δ values in Table 3, FPA and eFPA contribute in terms of generating the best overall test suite size. Comparatively, judging by the Δ values, eFPA generally outperforms FPA. As for the overall performance, SA appears to produce the best overall test suite size with ACA, GA, and eFPA coming in as the closest runner up. TVG perform the poorest overall.

One subtle observation is the fact that meta-heuristic-based strategies always outperform the general computational strategies with the exception of the case involving AETG with CA (N; 2, 3^{13}) and mAETG with CA (N; 3, 10^{6}) respectively.

Table 3 Comparison with existing strategies for experiment 1

System configurations	*Computational-based strategies*					*Meta-heuristic strategies*					
	mAETG	*AETG*	*IPOG*	*Jenny*	*TVG*	*SA*	*ACA*	*GA*	*PSO*	*HSS*	*CS*
CA(N; 2, 3^4)	9	9	9	10	11	9	9	9	9	9	9
CA(N; 2, 3^{13})	17	15	20	20	19	16	17	17	17	18	20
CA(N; 2, 10^{10})	NA	NA	176	157	208	NA	159	157	NA	155	NA
CA(N; 2, 5^{10})	NA	NA	50	45	51	NA	NA	NA	45	43	NA
CA(N, 2, 8^{10})	NA	NA	117	104	124	NA	NA	NA	109	105	NA
CA(N; 2, 15^{10})	NA	NA	373	336	473	NA	NA	NA	NA	342	NA
CA(N; 3, 3^6)	38	47	53	51	49	33	33	33	42	39	43
CA(N; 3, 4^6)	77	105	64*	112	123	64	64	64	102	70	105
CA(N; 3, 5^6)	194	NA	216	215	234	152	125	125	NA	199	NA
CA(N; 3, 5^7)	218	229	274	236	271	201	218	218	229	236	233
CA (N; 3, 10^6)	1,426	1,496	NA	1,572	1,919	1,508	1,501	1,473	1,506	1,505	NA
MCA(N; 2, 5^1 3^8 2^2)	20	19	19	23	22	15	16	15	NA	20	21
MCA(N; 3, 5^2 4^2 3^2)	114	NA	111	131	136	100	106	108	NA	120	NA

System configurations	*Meta-heuristic strategies*				$\Delta = \beta - \alpha$	
	FPA		*eFPA*			
	Avg	*Best*	*Avg*	*Best*	*Δ FPA*	*Δ eFPA*
CA(N; 2, 3^4)	10.6	9	10.0	9	0	0
CA(N; 2, 3^{13})	19.9	18	18.2	17	+3	+2
CA(N; 2, 10^{10})	156.7	153	153.9	150	–3	–5
CA(N; 2, 5^{10})	43.5	42	42.8	42	–1	–1
CA(N, 2, 8^{10})	105.55	103	105.34	101	Δ 1	–3
CA(N; 2, 15^{10})	348.7	345	339.1	333	+12	–3
CA(N; 3, 3^6)	45.5	44	46.3	41	+11	+8
CA(N; 3, 4^6)	105.0	101	103.8	93	+37	+29
CA(N; 3, 5^6)	199.0	195	198.6	194	+70	+69
CA(N; 3, 5^7)	222.3	220	221.9	217	+19	+16
CA (N; 3, 10^6)	1,484.2	1,478	1,475.0	1,470	+52	+44
MCA(N; 2, 5^1 3^8 2^2)	22.0	21	21.5	20	+6	+5
MCA(N; 3, 5^2 4^2 3^2)	123.0	118	122.5	113	+18	+13

Table 4 Comparison with existing strategies for experiment 2

t	*Computational-based strategies*								
	IPOG	*TVG*	*Jenny*	*TConfig*	*PICT*	*GTWay*	*ITCH*	*CTE-XL*	*MIPOG*
2	10	10	10	9	NA	NA	6	NA	NA
3	19	17	18	20	NA	NA	18	NA	NA
4	49	41	39	45	NA	NA	58	NA	NA
5	128	84	87	95	NA	NA	NS	NA	NA
6	352	168	169	183	NA	NA	NS	NA	NA
7	NS	302	311	NS	NA	NA	NS	NA	NA
8	NS	514	521	NS	NA	NA	NS	NA	NA
9	NS	651	788	NS	NA	NA	NS	NA	NA
10	NS	NS	1,024	NS	NA	NA	NS	NA	NA

t	*Meta-heuristic strategies*							$\Delta = \beta - \alpha$	
	PSO	*HSS*	*CS*	*FPA*		*eFPA*			
				Avg	*Best*	*Avg*	*Best*	*ΔFPA*	*ΔeFPA*
2	8	7	8	8.7	8	8.6	8	+2	+2
3	17	16	16	16.7	16*	17.1	16	0	0
4	37	37	36	38.4	35	39.0	36	–1	0
5	82	81	79	82.8	81	84.0	80	+2	+1
6	158	158	157	160.5	158	159.4	157	+1	0
7	NS	298	NS	295.0	292	293.4	290	–6	–8
8	NS	498	NS	504.1	500	505.2	495	+2	–3
9	NS	512	NS	671.8	592	705.6	621	+80	+109
10	NS	1,024	NS	1,024	11,024	1,024	1,024	0	0

Table 5 Comparison with existing strategies for experiment 3

P	*Computational-based strategies*								
	IPOG	*TVG*	*Jenny*	*TConfig*	*PICT*	*GTWay*	*ITCH*	*CTE-XL*	*MIPOG*
2	49	40	39	45	43	46	58	NA	43
3	241	228	221	235	231	224	336	NA	217
4	707	782	703	718	742	621	704	NA	637
5	1,965	1,917	1,719	1,878	1,812	1,714	1,750	NA	1,643
6	3,935	4,159	3,519	NA	3,735	3,514	NA	NA	3,657
7	7,061	7,854	6,462	NA	NA	6,459	NA	NA	5,927

P	*Meta-heuristic strategies*							$\Delta = \beta - \alpha$	
	PSO	*HSS*	*CS*	*FPA*		*eFPA*			
				Avg	*Best*	*Avg*	*Best*	*Δ FPA*	*ΔeFPA*
2	34	37	28	36	36	39.9	36	+2	+2
3	213	211	211	212.3	211	209.8	208	0	–3
4	685	691	698	662.6	661	659.8	657	+24	+20
5	1,716	1,624	1,731	1,603.5	1,605	1,592.4	1,592	–19	–32
6	3,880	3,475	3,894	3,354.0	3,354	3,310.4	3,310	–130	–165
7	NA	6,398	NA	6,215.4	6,205	6,211.8	6,095	+288	+168

Table 6 Comparison with existing strategies for experiment 4

P	*Computational-based strategies*								
	IPOG	*TVG*	*Jenny*	*TConfig*	*PICT*	*GTWay*	*ITCH*	*CTE-XL*	*MIPOG*
5	908	849	837	773	810	731	625	NA	625
6	1,239	1,128	1,074	1,092	1,072	1,027	625	NA	625
7	1,349	1,384	1,248	1,320	1,279	1,216	1,750	NA	1,125
8	1,792	1,595	1,424	1,532	1,468	1,443	1,750	NA	1,384
9	1,793	1,795	1,578	1,724	1,643	1,579	1,750	NA	1,543
10	1,965	1,971	1,791	1,878	1,812	1,714	1,750	NA	1,643
11	2,091	2,122	1,839	2,038	1,957	1,852	1,750	NA	1,722

P	*Meta-heuristic strategies*							$\Delta = \beta - \alpha$	
	PSO	*HSS*	*CS*	*FPA*		*eFPA*			
				Avg	*Best*	*Avg*	*Best*	*Δ FPA*	*ΔeFPA*
5	779	751	776	788.2	784	786.5	778	+159	+153
6	1,001	990	991	991.4	988	994.6	985	+363	+360
7	1,209	1,186	1,200	1,170.0	1,164	1,169.4	1,166	+39	+41
8	1,417	1,358	1,415	1,330.4	1,329	1,324.8	1,319	–29	–39
9	1,570	1,530	1,562	1,478.0	1,478	1,466.4	1,465	–52	–65
10	1,716	1,624	1,731	1,603.5	1,605	1,592.4	1,592	–19	–23
11	1,902	1,860	2,062	1,742.0	1,739	1,732.3	1,719	+17	–3

Table 7 Comparison with existing sequence-based t-way strategies for experiment 5

System configurations	*U*	*Ur*	*BA*	*SCAT*	*T-SEQ*	*FPA*		*eFPA*		$\Delta = \beta - \alpha$	
						Avg	*Best*	*Avg*	*Best*	*Δ FPA*	*ΔeFPA*
SCA(N;3, 4)	12	12	6	8	NA	7.2	6	6.25	6	0	0
SCA(N;3, 5)	17	16	8	10	8	8.6	7	8.0	7	–1	–1
SCA(N;3, 6)	20	18	9	12	10	9.8	9	9.4	8	0	–1
SCA(N;3, 7)	23	22	10	12	12	10.7	10	10.0	10	0	0
SCA(N;3, 8)	26	24	11	12	12	11.5	11	11.1	10	0	–1
SCA(N;3, 9)	28	26	13	14	14	12.3	12	12.1	11	–1	–2
SCA(N;3, 10)	30	28	14	16	14	13.3	13	12.8	12	–1	–2
SCA(N;3, 11)	32	30	NA	16	14	13.8	13	13.6	13	–1	–1
SCA(N;3, 12)	33	30	NA	16	16	14.8	14	14.0	13	–2	–3
SCA(N;3, 13)	35	32	NA	18	16	15.4	15	15.6	15	–1	–1
SCA(N;3, 14)	36	34	NA	18	16	16.53	16	16.0	16	0	0
SCA(N;3, 15)	37	34	NA	20	18	16.8	16	16.4	16	–2	–2
SCA(N;3, 16)	39	36	NA	18	18	17.2	17	17.0	17	–1	–1
SCA(N;3, 17)	40	36	NA	20	20	18.0	18	17.6	17	–2	–3
SCA(N;3, 18)	41	38	NA	20	20	18.8	18	18.6	18	–2	–2
SCA(N;3, 19)	42	38	NA	20	22	19.6	19	19.0	19	–3	–3
SCA(N;3, 20)	42	38	NA	22	22	21.25	20	20.33	20	–2	–2

Table 8 Comparison with existing sequence-based t-way strategies for experiment 6

System configurations	*U*	*Ur*	*BA*	*SCAT*	*T-SEQ*	*FPA*		*eFPA*		$\Delta = \beta - \alpha$	
						Avg	*Best*	*Avg*	*Best*	*Δ FPA*	*ΔeFPA*
SCA(N;4, 5)	54	54	28	24	26	29.8	29	29.1	28	+5	+4
SCA(N;4, 6)	79	78	36	36	36	36.8	36	36.0	36	0	0
SCA(N;4, 7)	98	96	45	46	46	43.6	43	42.2	42	–3	–4
SCA(N;4, 8)	114	112	55	54	50	50.4	50	48.6	48	0	–2
SCA(N;4, 9)	128	126	62	62	58	57.6	57	55.5	54	–1	–8
SCA(N;4, 10)	140	138	71	64	66	64.71	63	62.5	61	–1	–3
SCA(N;4, 11)	151	148	NA	72	70	68.2	67	68.8	68	–3	–2
SCA(N;4, 12)	160	158	NA	82	78	74.8	74	74.2	74	–4	–4
SCA(N;4, 13)	169	166	NA	86	86	79.8	79	80.0	79	–7	–7
SCA(N;4, 14)	177	174	NA	90	90	85.4	84	85.8	84	–6	–6
SCA(N;4, 15)	184	180	NA	90	96	90.0	90	90.5	89	–6	–7
SCA(N;4, 16)	191	188	NA	96	100	97.0	97	97.0	97	–3	–3
SCA(N;4, 17)	197	194	NA	104	108	103.67	101	101.0	103	–3	–1
SCA(N;4, 18)	203	203	NA	106	112	106.67	106	105.67	105	0	–1
SCA(N;4, 19)	209	209	NA	114	114	111.0	109	110.0	110	–5	–4
SCA(N;4, 20)	214	214	NA	112	120	115.0	114	115.45	115	+2	+3

Table 9 Comparison with existing sequence-based t-way strategies for experiment 7

System configurations	*U*	*Ur*	*BA*	*SCAT*	*T-SEQ*	*FPA*		*eFPA*		$\Delta = \beta - \alpha$	
						Avg	*Best*	*Avg*	*Best*	*Δ FPA*	*ΔeFPA*
SCA(N;5, 6)	294	294	159	154	NA	151.4	148	154.8	152	–6	–2
SCA(N;5, 7)	437	436	212	212	NA	200.0	199	197.0	194	–3	–8
SCA(N;5, 8)	552	550	271	264	NA	249.8	247	241.4	240	–17	–24
SCA(N;5, 9)	648	646	329	324	NA	295.5	295	291.3	283	–29	–41
SCA(N;5, 10)	731	728	383	368	NA	349.5	344	345.75	344	–24	–24
SCA(N;6, 7)	NA	NA	NA	NA	NA	982.3	980	963.4	960	-	-
SCA(N;6, 8)	NA	NA	NA	NA	NA	1,311.5	1,301	1,282.34	1,274	-	-
SCA(N;6, 9)	NA	NA	NA	NA	NA	1,639.11	1,636	1,633.34	1,628	-	-
SCA(N;6, 10)	NA	NA	NA	NA	NA	1,998.0	1,998	2,164.3	2,161	-	-

Unlike Table 3, Table 4 deals with fixing CA (N; t, 2^{10}) whilst varying *t* from 2 to 10. Referring to the Δ values, FPA and eFPA also contribute in terms of generating the best overall test suite size. Again, based on the Δ values, eFPA generally outperforms FPA. However, there is a large difference in Δ values for the case involving CA (N; t, 2^{10}) for FPA and eFPA. In this case, it appears that elitism element within eFPA is counter-productive. We perceive this case as outlier owing to the fact that meta-heuristics can be influenced by chance – as the algorithm relies heavily on randomness to generate the solution. Concerning the overall performance, eFPA performs the best followed by HSS and CS. As with Table 3, meta-heuristic-based strategies outperform the computational-based ones.

Table 5 shows the results of the comparison with existing strategies when v varied from 2 to 7 for CA(N; 4, v^{10}). Referring to the Δ values, eFPA is also outperforming its FPA counterparts. As for the overall performance, eFPA outperform other strategies with three cases (i.e., with v = 3, v = 5, and v = 6) with MIPOG comes as the runner up.

Table 6 reports results for CA(N; 4, 5^P) where P is varied from 5 to 12. Based on Δ values, eFPA outperforms its FPA counterparts in most cases. In fact, eFPA outperforms all the existing strategies when P is varied from 8 to 11. MIPOG outperforms the existing strategies when P equal to 5, 6, 7 and 12. In fact, MIPOG optimises both the vertical and horizontal extensions which improve the test suite size.

6.2.2 Benchmarking with sequence t-way strategies

In this section, we compare FPA and eFPA with available results for sequence-based t-way strategies involving U, Ur, BA, SCAT, and T-SEQ. The experiments were adopted from Ahmad et al. (2016), Chee et al. (2013), Kuhn et al. (2012) and Zabil et al. (2012). Our benchmark comparison is based on the following experiments.

- *Experiment 5*: comparison with existing strategies for SCA(N;t, e), t =3 and e varied from 4 to 20.
- *Experiment 6*: comparison with existing strategies for SCA(N;t, e), t =4 and e varied from 5 to 20
- *Experiment 7*: comparison with existing strategies for SCA(N;t, e), t =5 and t = 6, with e varied from 6 to 10, and 7 to 10, respectively.

Table 7 shows that FPA and eFPA have good overall performance as compared to existing work. Most Δ columns give negative values indicating that most of the results for FPA and eFPA outperform others. In fact, eFPA and FPA generates nearly half of the test suite size as compared to the cases of U and Ur.

Table 8 reports comparison results for SCA(N;4, e) where e is varied from 5 to 20. In this case, the performance of eFPA and FPA is similar. In fact, FPA and eFPA outperform the existing strategies almost all test cases except when e = 5 and e = 20. In these two cases, SCAT has outperformed FPA and eFPA. Again, U and Ur perform the poorest in all cases. Referring to Table 9, all Δ columns give negative values indicating that most of the results for FPA and eFPA outperform other strategies. We note that all existing strategies do not give the support for t = 5 and t = 6.

7 Discussion

Finding a general strategy for generating sequence and sequence-less test suite is difficult as both are NP-hard problems. Sequence t-way test generation focuses on obtaining sequences of events (as test cases) such that every combination of events is covered at least once with respect to the order. For sequence-less t-way test generation, the interaction elements must be covered at least once whilst the order of coverage is not important. When dealing these two issues, developing a unified strategy for both sequence and sequence-less less t-way test generation dictates factoring out the way that t-way coverage is handled. The work on FPA and its variant are the first of its kind as most existing strategies support either but not both cases involving sequence and sequence-less t-way test generation.

Concerning the overall performance of eFPA for each sequence and sequence-less, eFPA strategy produces sufficiently competitive results against existing strategies in most of cases with more superiority for sequence results. Comparing to FPA, eFPA is able to generate better results in most of cases due to its enhanced exploration capability (i.e., via elitism).

8 Conclusions and further works

Summing up, this paper has discussed and evaluated the unified eFPA supporting sequence and sequence-less t-way coverage. Our experiences with eFPA have been encouraging. As part of the future work, we are looking to support constraints as well as adopt eFPA for software product lines applications.

Acknowledgements

The work undertaken in this paper is supported by the Fundamental Research Grant: Reinforcement Learning Sine Cosine based Strategy for Combinatorial Test Suite (RDU170103) from Ministry of Higher Education Malaysia.

References

Ahmad, M.Z.Z., Othman, R.R. and Ali, M.S.A.R. (2016) 'Sequence covering array generator (scat) for sequence based combinatorial testing', *International Journal of Applied Engineering Research*, Vol. 11, No. 8, pp.5984–5991.

Ahmed, B.S., Abdulsamad, T.S. and 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*, Vol. 66, No. C, pp.13–29, doi: 10.1016/ j.infsof.2015.05.005.

Ahmed, B.S., Zamli, K.Z. and Lim, C.P. (2012) 'Application of particle swarm optimization to uniform and variable strength covering array construction', *Applied Soft Computing*, Vol. 12, No. 4, pp.1330–1347.

Alsewari, A.R.A. and Zamli, K.Z. (2012) 'Design and implementation of a harmony-search-based variable-strength t-way testing strategy with constraints support', *Information and Software Technology*, Vol. 54, No. 6, pp.553–568.

Burr, K. and Young, W. (1998) 'Combinatorial test techniques: table-based automation, test generation and code coverage', Paper presented at the *Proceedings of the International Conference on Software Testing Analysis and Review*.

Chee, Y.M., Colbourn, C.J., Horsley, D. and Zhou, J. (2013) 'Sequence covering arrays', *SIAM Journal on Discrete Mathematics*, Vol. 27, No. 4, pp.1844–1861.

Cohen, M.B. (2004) *Designing Test Suites for Software Interaction Testing*, Doctoral dissertation, University of Auckland [online] https://cse.unl.edu (accessed December 2017).

Erdem, E., Inoue, K., Oetsch, J., Pührer, J., Tompits, H. and Yılmaz, C. (2011) 'Answer-set programming as a new approach to event-sequence testing', Paper presented at the *International Conference on Advances in System Testing and Validation Lifecycle*.

Geem, Z.W. (2009) *Music-inspired Harmony Search Algorithm: Theory and Applications*, Vol. 191, Springer, Berlin Heidelberg.

Harman, M. and Jones, B.F. (2001) 'Search-based software engineering', *Information and Software Technology*, Vol. 43, No. 14, pp. 833–839.

Harman, M., Mansouri, S.A. and Zhang, Y. (2009) *Search based Software Engineering: a Comprehensive Analysis and Review of Trends Techniques and Applications*, (TR-09-03).

Kuhn, D.R., Higdon, J.M., Lawrence, J.F., Kacker, R.N. and Lei, Y. (2012) 'Combinatorial methods for event sequence testing', Paper presented at the *IEEE 5th International Conference on Software Testing, Verification and Validation*.

Nasser, A.B., Alsewari, A.R.A. and Zamli, K.Z. (2015) 'Tuning of cuckoo search based strategy for t-way testing', Paper presented at the *International Conference on Electrical and Electronic Engineering*.

Offutt, J., Liu, S., Abdurazik, A. and Ammann, P. (2003) 'Generating test data from state-based specifications', *Software Testing, Verification and Reliability*, Vol. 13, No. 1, pp.25–53.

Rahman, M., Othman, R.R., Ahmad, R.B. and Rahman, M.M. (2015) 'Event driven input sequence t-way test strategy using simulated annealing', Paper presented at the *5th International Conference on Intelligent Systems, Modelling and Simulation (ISMS)*.

Shiba, T., Tsuchiya, T. and Kikuno, T. (2004) 'Using artificial life techniques to generate test cases for combinatorial testing', Paper presented at the *28th Annual International Computer Software and Applications Conference*.

Stardom, J. (2001) *Metaheuristics and the Search for Covering and Packing Arrays*, Simon Fraser University, Canada.

Sthamer, H-H. (1995) *The Automatic Generation of Software Test Data Using Genetic Algorithms*, PhD thesis, University of Glamorgan.

Yang, X-S. (2012) 'Flower pollination algorithm for global optimization', Paper presented at the *International Conference on Unconventional Computing and Natural Computation*.

Yang, X-S., Deb, S. and Fong, S. (2014) 'Metaheuristic algorithms: optimal balance of intensification and diversification', *Applied Mathematics and Information Sciences*, Vol. 8, No. 3, p.977.

Yilmaz, C., Cohen, M.B. and Porter, A.A. (2006) 'Covering arrays for efficient fault characterization in complex configuration spaces', *IEEE Transactions on Software Engineering*, Vol. 32, No. 1, pp.20–34.

Zabil, M.H.M., Zamli, K.Z. and Othman, R.R. (2012) 'Sequence-based interaction testing implementation using bees algorithm', Paper presented at the *IEEE Symposium on Computers and Informatics*.

Zamli, K.Z., Alkazemi, B.Y. and Kendall, G. (2016) 'A tabu search hyper-heuristic strategy for t-way test suite generation', *Applied Soft Computing*, Vol. 44, pp.57–74.

Zamli, K.Z., Din, F., Baharom, S. and Ahmed, B.S. (2017a) '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.

Zamli, K.Z., Din, F., Kendall, G. and Ahmed, B.S. (2017b) 'An experimental study of hyper-heuristic selection and acceptance mechanism for combinatorial t-way test suite generation', *Information Sciences*, Vol. 399, pp.121–153.