African Buffalo Optimization Algorithm Based T-Way Test Suite Generation Strategy for Electronic-Payment Transactions

Julius B. Odili[1(✉)], Abdullah B. Nasser[2], A. Noraziah[2,3], Mohd Helmy Abd Wahab[4], and Mashuk Ahmed[2]

[1] Faculty of Science and Science Education, Anchor University Lagos, Lagos, Nigeria
[2] Faculty of Computing, Universiti Malaysia Pahang, 26300 Kuantan, Malaysia
{abdullahnasser,noraziah}@ump.edu.my
[3] Centre for Software Development and Integrated Computing, Pekan, Malaysia
[4] Advanced Communication Research Center, Universiti Tun Hussein Onn Malaysia, Parit Raja, Malaysia
MCS20004@sudent.ump.edu.my

Abstract. The use of meta-heuristics in Combinatorial Interaction Testing (CIT) is becoming more and more popular due to their effectiveness and efficiency over the traditional methods especially in authenticating electronic payment (e-payment) transactions. Concomitantly, over the past two decades, there has been a rise both in the development of metaheuristics and their application to diverse theoretical and practical areas including CIT in e-payments. In the implementation of t-way strategies (the *t* is used to represent the interaction strength), mixed results have been reported; some very exciting but, in other cases, the performance of metaheuristics has been, to say the least, below par. This mixed trend has led many researchers to explore alternate ways of improving the effectiveness and efficiency of metaheuristics in CIT, hence this study. It must be emphasized, however, that available literature indicates that no particular metaheuristic testing strategy has had consistent superior performance over the others in diverse testing environments and configurations. The need for effectiveness, therefore, necessitates the need for algorithm hybridization to deploy only the component parts of algorithms that have been proven to enhance overall search capabilities while at the same time eliminating the demerits of particular algorithms in the hybridization procedure. In this paper, therefore, a hybrid variant of the African Buffalo Optimization (ABO) algorithm is proposed for CIT. Four hybrid variants of the ABO are proposed through a deliberate improvement of the ABO with four algorithmic components. Experimental procedures indicate that the hybridization of the ABO with these algorithmic components led to faster convergence and greater effectiveness superior to the outcomes of existing techniques, thereby placing the algorithm among the best when compared with other methods/techniques.

Keywords: African Buffalo Optimization · e-payment system · T-way testing · Software testing · Combinatorial problem · Optimization techniques

M. Al-Emran et al. (Eds.): ICETIS 2021, LNNS 299, pp. 160–174, 2022.
https://doi.org/10.1007/978-3-030-82616-1_15

1 Introduction

With the massive deployment of electronic payment systems, especially during this period of Lockdown across many countries of the world as a result of the New Corona Virus, otherwise called COVID-19, the need to examine the software used in e-payments have not been greater in human history than at present. Different money-transaction organizations, the world over have developed several payment platforms/software, many of these software have, to say the least, been quite successful. However, the need for continuous improvements cannot be over-emphasized. The need to ensure foolproof transactions in our monetary systems has necessitated several research investigations to ensure stability, reliability and security of financial transactions in our financial institutions. This has led to the involvement of artificial intelligence techniques to assist is ascertaining the reliability of existing financial software. One of the areas attracting researcher's attention in this regard is the use of optimization algorithms is software testing.

Optimization has gradually been entrenched in virtually all aspects of Software Engineering ranging from software requirements engineering to software management, software testing and software refactoring respectively. Optimization, fundamentally, is sometimes defined as the application of economic principles to science and engineering which basically deals with generating the highest amount of output with as little input as possible [1]. In other words, optimization is the process of discovering the best alternative out of a given number of alternatives utilizing limited inputs as much as possible [2]. In the past few decades, there has been a marriage of convenience between optimization and metaheuristic algorithms. Some of the popular metaheuristic algorithms include: Simulated Annealing (SA) [3]; Tabu Search (TS) [5], African Buffalo Optimization [6], Great Deluge (GD) [4];; Ant Colony Optimization (ACO) [3]; and Particle Swarm Optimization (PSO) [7], etc.

Optimization becomes very important in Combinatorial Interaction Testing (CIT) in order to identify the optimal cases from a large range of combinatorial values based on the specified interaction strength (t) since the major task of CIT is to determine the optimal test cases from among a large test suite. Since it may not be possible or efficient to test every possible test combination in a large test suite, there is the need for combinatorial optimization [8]. Combinatorial optimization is relevant in a situation where an exhaustive search is practically impossible or totally inefficient when a choice has to be made, thereby emphasizing the need for optimization search algorithms [9].

In most testing situations (similar to our focus in this study), CIT is a (NP-hard (Non-Polynomial-hard) problem. The term NP-hard problem refers to classes of problems that there exists no algorithm that provides a solution to the problem as well as the verification of such solutions in a polynomial time. Polynomial time is simply a linear time. This further underscore the need for metaheuristics. In response, today, the research community has designed several metaheuristics which have been applied to several NP-hard problems such as the travelling salesman's problem [10]; N-Queen problems; numerical function optimization [11]; and software testing [12] etc. With particular reference to software testing, the PSO, SA, and (HS) [13] have proven to be quite successful. It must, however, be emphasized that in spite of the huge contributions of the above algorithms, there still exists the need for further improvements, hence this paper.

The remainder of this paper is structured thus: section two gives a general overview of t-way testing in addition to discussing its theoretical background; section three presents a review of some existing t-way strategies; section four examines the hybridization of variants of the ABO algorithm for an appropriate t-way test suite generation of an e-payment system; section five is concerned with the experimental investigation of the ABO algorithm for a t-way test suite of an e-payment system and discusses the experimental outcomes while section six presents the conclusions drawn from the study.

2 Explanation of Background and History

2.1 The Need for T-Way Test Suite Generation for E-payment Platforms

Of the several CIT techniques in use, the t-way testing has proven to be one of the most effective combinatorial techniques that can dramatically reduce the size of the test parameters without compromising effectiveness. T-way testing, basically, is a sampling mechanism used to generate test samples with a clear focus on the attitude of interaction system's component parts. To further elaborate on the need for t-way testing in test suite reduction, consider the online payment systems such as MasterCard, Visa etc. Online Payment systems enable customers to carry out banking transactions electronically.

To carry out an electronic payment transaction, a customer is required to simply fill out the online Payment Details form and then submit to the merchant's website. Usually, the transaction form consists of six different inputs, namely: The Payment System (Visa, MasterCard, PayPal, American Express etc.); Name on Card; Expiration Date (MM and YY); Card Number; Card and Card's CVV. In the Transaction Method, a customer is required to choose the particular Payment method from among a number of choices including: "Discover"; "Visa Card"; "American Express"; "MasterCard"; and "PayPal". The next input data is the Name on Card, then the Card Number which takes numeric string values. Following this is the Card Expiration Date, which requires further numeric inputs (i.e. MM takes values 1 … 12, and YY takes values 16 … 31). The last compulsory input is the Card CVV which requires another numeric value. An example of the online payment system is presented in Fig. 1.

Fig. 1. Electronic payment system

To exhaustively test the electronic payment system, 900 test cases are required [14]. Nevertheless, utilizing a two-way test suite reduces it to only 180 test cases [15]. In

general, the test suite T is $n \times m$ array which consists of n rows of the generated test cases and m columns in such a way that each test case becomes a combination of m input values. Thus, whereas the t-way test suite T1 covers all valid pair values of input parameters, one test case incorporates many pairs of input parameter values. Therefore, a good software tester would need to discover an effective test suite T1 which is a subset of T that should have the smallest number of rows and columns with the embedded capacity of unravelling a defect wherever one exists.

In scientific literature, there exist many test suite generation strategies such as: partition testing [16]; random testing [17]; combinatorial interaction testing [19]; etc.

2.2 Theoretical Foundation of T-Way Test Suite Generation

To formulate CIT, many researchers adopt Covering Array (CA). Basically, from literature, any system under test (SUT) is made up of a number of component parts which interact with each other parts /parameters cooperatively with each component making available its values to other components in the system. Please note that throughout this paper, the symbols p, t, and v refer to number of parameters, interaction strength and values, respectively [21, 22].

Similarly, it is important to emphasize that whenever the number of values (v) is the same with the number of parameters (p), the covering array is assumed to be a uniform Covering Array, CA (N, t, v^p). For instance, anytime a covering array of CA (6; 2, 2^4) is made up of six (6) rows of the test cases being generated from the four (4) columns of the appropriate parameters with only two (2) values, then it is a case of a uniform Covering Array. On the other hand, anytime the number of parameters being investigated does not correlate (that is, when. each parameter under consideration generates a different number of values), then such covering array is a the Coverage Array symbol of MCA (N, t, $v_1{}^{p1}$ $v_2{}^{p2}$ $v_3{}^{p3}$…..$v_j{}^{pj}$).

3 A Detailed Explanation of Recent Findings

3.1 Hybrid African Buffalo Optimization for T-Way Test Suite Generation

The primary objective of this section is to present an efficient, effective cum fast t-way strategy for a good test suite generation based on a meta-heuristic algorithm for the authentication of an electronic payment system. The need for hybridization emanates from the effectiveness of hybrid algorithm over the classical algorithms because hybrid algorithm draws from the strength of each of the algorithms in the hybrid [23]. Hybridization is a very successful concept in the development and use of meta heuristic algorithms. In literature, many hybridization methods have been proposed by simply selecting two algorithms to a new one [24]. Even though the exceptional performance of a new hybrid is not guaranteed, some hybridization has been very successful [24].

Therefore, finding an effective way for algorithm-hybridization is still an open issue for researchers [25]. To design an effective hybrid algorithm, there is the need for a thorough grasp of the algorithms' components and employ the concept of component grafting [26]. In this study, first, a new t-way suite test generation strategy is proposed; called Buffalo Strategy (BS) based on the ABO algorithm [27, 28]. The ABO has been improved

to obtain several hybridization variants of the original ABO. Lastly, the hybrid variants of BS have been evaluated and their performance compared with existing strategies.

The ABO is one of the 21st century swarm intelligence-based algorithm. The ABO was inspired by the unique intelligence displayed by migrant buffalos searching for food in the African continent. ABO being is a swarm-based technique where simulates the aggregate intelligence of individual buffalo as they optimize their search for food sources [29]. ABO attempts to model three qualities of the African Buffalo as follows:

A. Extensive memory capacity of African Buffalos. These animals keep track of their migrant paths for hundreds of miles;
B. Democratic nature: The major decisions of the herd are determined by "election";
C. Cooperative behavior of buffalos: African Buffalos have two sounds: the alert '/waaa/' sound prompting the herd to move since their present because location is dangerous or unfavorable for some reasons; while the sound "/maaa/" requests other Buffalos to stay in the same location [30].

The ABO algorithm starts by initializing a number of buffalos and then updates each buffalo's fitness using Eq. 1. At this point, the algorithm saves the fitness of each buffalo *bp* and the global fitness overall of the herd b*g*. *lpr1* and *lpr2* are learning factors. The three parts of Eq. 1 (i.e. w^t, lpr1 $\times$ $(bg - w^t)$, and lpr2 $\times$ $(bg - w^t)$ represent: memory capacity; the cooperative behavior of buffalos; and the intelligence part of the buffalos, respectively. Equation 2, basically, is used to update the buffalo's locations. Please note that λ is a random number. The ABO algorithm searching strategy is governed by the equations [31]:

$$m^{(t+1)} = w^{(t)} + lpr1 * (bp - w^{(t)}) + lpr2 * (bg^{(t)} - w^{(t)}) \tag{1}$$

and

$$W_{k'} = (W_k + m_k)/\lambda \tag{2}$$

3.2 Buffalo Strategy T-Way Test Suite Generation for E-payment

In this section, our concern with the presentation of the Buffalo Strategy (BS) for the design and implementation of a tway test suite generator [32]. BS uses the original ABO in generating optimized test suite through a diligent search for particular test cases which incorporates the maximum number of t-combinations of inputs at least once. It treats a buffalo (that is, a feasible solution and interaction elements) within the search space. Basically, BS starts by first generating all interaction elements that are within the present search space into a list and representing them as component units of a population of buffalos.

At the evaluation loop, which is the next step, the buffalos are repeatedly subjected to a cycle of ABO search procedures in order to generate the expected final optimized test suite. In general, the BS comprises of two steps:

- Produce interaction test elements;

- Produce T-way Test Suite by diligent search procedures of the ABO to discover the optimal test cases. Please see Fig. 2.

1. ***Producing Interaction Element Step:***
   Let IE be a set of Interaction Elements
   B = generate the smallest binary number contain t ones.
   Until all t binary number generated
   Generate all possible interaction elements for B
   Add all possible interaction elements to
   IE B = Generate_next_binary_number(B).
   End loop
2. ***Generating T-way Test Suite Step:***
   While *IE* is not empty **do**
   //Perform search using ABO **while** <Max
   Generation or stop criterion **do**
   For i =0 to *buffalo_size* **loop**
   Update buffalo's fitness:
   $m^{(t+1)} = w^{(t)} + lpr1 * (gbest - w^{(t)}) + lpr2 * (bg\_best^{(t)} - w^{(t)})$
   Update buffalos location
   $w_{k'} = (w_k + m_k) / \lambda$
   Find the best buffalo
   End for
   End while
   Add the best buffalo into TS.
   Remove covered interactions elements from IE.

End while

Fig. 2. BS for T-way test suite generation

3.3 Parameter Adjustment of BS

The behavior of BS is determined by the population size *buffalo_size*, learning factors *lp1* and *lp2* as well as the iteration number *n*. As a result, these parameters need to be properly tuned. To do this, one of the well-known covering arrays CA (N; 2, 10^5) is used. To ensure systematic tuning, the values of two parameters are fixed as we investigate the best value of the other parameter. For instance, the value of buffalo size is iteration fixed (i.e. buffalo size = 10, and iteration = 30) and those of *lp1* and *lp2* (0.1, 0.2, 0.3… 0.6) are investigated (See Tables 1 and Fig. 3). Moreover, the reverse process is performed for each (See Table 3, and Table 4). To ensure reliability, BS is executed 20 times with every parameter value, and the mean is recorded (See Table 1).

Table 1. Mean Test Suite for CA (N; 2, 10^5) with varied value for Iteration and pollen size

Buffalo Size	Iteration										
	5	10	20	30	40	50	100	200	300	500	700
10	35.65	32.15	32.95	30.30	29.95	29.15	28.55	27.65	26.40	25.65	25.25
20	32.20	29.65	28.80	28.45	27.55	27.40	26.95	25.10	24.95	24.80	24.50
30	30.95	28.65	28.00	26.80	26.75	25.50	25.35	24.65	24.50	24.20	24.00
50	29.05	26.65	27.20	25.95	25.95	25.15	25.55	24.10	24.00	23.00	23.40
100	27.15	26.10	25.65	25.25	24.65	24.70	24.55	23.85	24.10	23.90	23.50
200	26.25	25.15	24.80	24.65	24.50	24.35	24.00	23.40	23.50	23.90	23.75
300	25.90	24.90	24.55	24.05	24.45	24.15	23.95	23.80	23.70	23.45	24.00
500	24.80	24.50	24.15	24.05	23.70	23.80	24.10	23.35	23.55	23.80	23.65

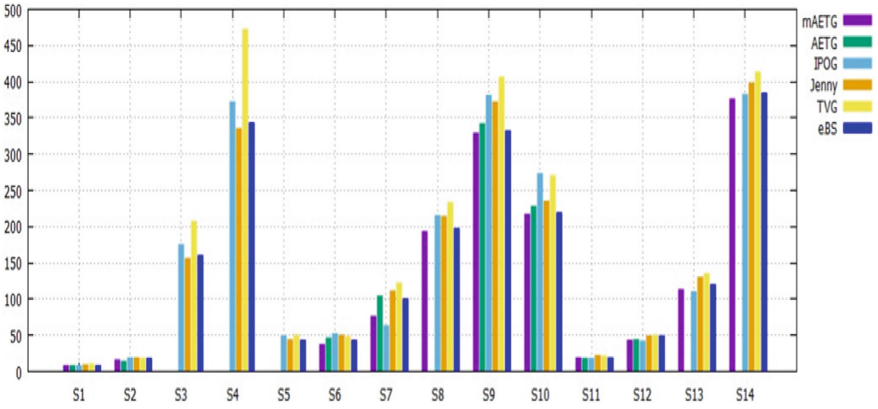

Fig. 3. Comparison of eBS with Computational-based Strategies

Table 1 clearly shows that using a large buffalo population leads to better results and vice-versa. So using a large population of up to 30 buffalos, the performance of BS improved significantly. Nonetheless, increasing the buffalo population to as many as 500 did not necessarily give better results. The best outcomes were realized when the buffalo population size ranges from 50 to 100 buffalos. Also, as the iteration value increased, the results improved (See Table 1). The best outcome was obtained when the iteration number was between 300 and 500.

With regards to the learning factors *lp1* and *lp2* (See Table 2), results show that using a higher value of *lp1* and *lp2* led to better results. Nevertheless, the best results were obtained when *lp1* and *lp2* was between 0.4 and 0.6. To summarize, BS obtained the optimal test suite when the buffalo population was between 50 and 100, iteration was between 300 and 500 and learning parameters between 0.4 and 0.6.

3.4 ABO-Based Strategies for Test Generation

From our experiments, the obtained results of BS using original ABO were very competitive when compared with those from other strategies such as: Simulated Annealing (SA); Harmony Search (HS); Particle Swarm Optimization (PSO) and Genetic Algorithm (GA) One of the observed drawbacks of ABO-based strategy is low randomization of the population.

Therefore, to enhance ABO's performance, we added a few components consistent with other efficient algorithms to the ABO algorithm. The additional components required to enhance local intensification and global diversification are discussed below.

- *Elitism Technique:* Elitism Technique is simply a way to make the randomization more efficient by replacing a part of the population to ensure that the solution quality will not be degraded in the next iteration [33];
- *Mutation operator:* This helps the algorithms to ensure the diversification of solutions of the population from one generation to the next. In mutation, one or more solution values are changed. There are different types of mutation including: Flip bit; Boundary; Bit string mutation; Gaussian mutation [34]; etc.
- *Local Search*: This is a straight-forward and highly effective technique for finding a local optimum solution. Local search deliberately moves from the current location to one of the neighboring states while seeking a local optimum. That is to say, local search emphasizes diligent exploitation of the search area that has a likelihood of good solutions [35].

The hybridization of ABO with the above components promotes greater search effectiveness (See Figs. 3 and 5). In light of the greater effectiveness, this paper proposes four different hybrid variants of BS: mutation Buffalo Strategy (mBS); local-search Buffalo Strategy (lBS); elitism Buffalo Strategy (eBS); and elitism local-search Buffalo Strategy (elBS). The eBS variant uses the elitism technique to fine-tune the buffalo population randomly replacing the unprofitable buffalos with new ones. The mBS variant, on the other hand, uses mutation operator to include diversity in the population of buffalos. On its part, the lBS uses intensive local search to improve local intensification while the elBS injects elements of both elitism technique and local search into BS.

3.5 Evaluation of Hybrid Variants of ABO

The four hybrid variants of BS were tested so as to select the best hybrid variant which can compete with existing test generation strategies techniques (See Fig. 3). Determining the optimal parameters of the various variants of BS parameters is challenging but necessary. To achieve this, a detailed parametric study was conducted for BS parameters as can be seen in Sect. 3.2. To do this, we deliberately chose the mutation rate to be 0.03 and elitism probability to be 0.25 as published in [36, 37], respectively. We applied the Hybrid Algorithms for three well-known covering array problems such as CA(N; 2, 10^5), and CA(N; 2, 4^6) (See Fig. 4). The result is presented in Table 2.

Table 2. Evolution of hybrid variants of ABO

Hybridization	CA(N; 2, 10^5)			CA(N; 2, 4^6)			CA(N; 3, 5^6)		
	Avg	Best	Time(s)	Avg	Best	Time(s)	Avg	Best	Time(s)
BS	142.0	141	9.42	25.7	25	1.03	221.4	220	108.89
eBS	122.4	120	7.89	23.5	22	1.09	198.2	197	155.35
lBS	127.10	126	35.34	23.9	23	5.01	204.8	202	11467.32
mBS	140.4	139	11.566	25.8	25	1.64	215	219.4	205.69
elBS	123.3	121	34.35	23.7	22	7.93	198.4	197	11472.83

The results in Table 2 show that all hybrid algorithm variants of BS outperformed the original BS in terms of average test suite size and best test suite size. Similarly, the results show that ABO-Elitism (eBS) produced better results than the other variants of BS with average (122.4) and Best (120). Moreover, the performance of elBS with average (123.3) and Best (121) is next to that of eBS. The next best performer is the lBS with average (127.1) and Best (126). The least efficient performer is the mBS with average (140.4) and Best (139).

In terms of time taken to achieve results, the eBS still maintained its superiority with 7.89 s, followed by BS with 9.42 s; mBS 11.566 s; elBS, 34.35 s and lBS, 35.34 s. From this analysis, it is evident that, although the BS did not produce very good results, it is an efficient algorithm since the CPU time correlates with less use of computer resources to produce results [38]. Nevertheless, kudos to eBS for producing the best results in the shortest possible time.

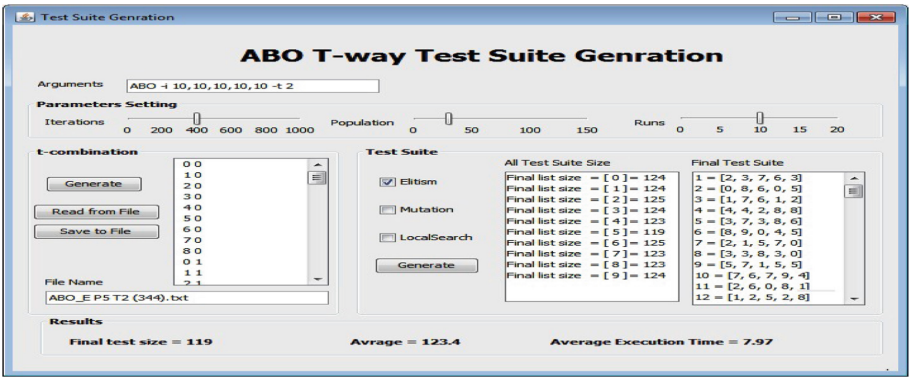

Fig. 4. Implementation of hybrid ABO-based Strategies

Next, we investigated the convergence behavior of hybrid ABO-based variants since convergence behavior is an important issue in performance evaluation of any hybridization (see Fig. 3). To evaluate the convergence rate of these hybrid variants of BS, we executed the hybrid variants of BS 20 times with different test suite sizes and iteration

limits (5, 10, 20, 30, 40, 50, 100, 200, 300, 500, and 1000). The mean value of the 20 runs for the two well-known covering arrays CA CA (N; 2, 10^5) and (N; 2, 4^6) are taken (See Fig. 5).

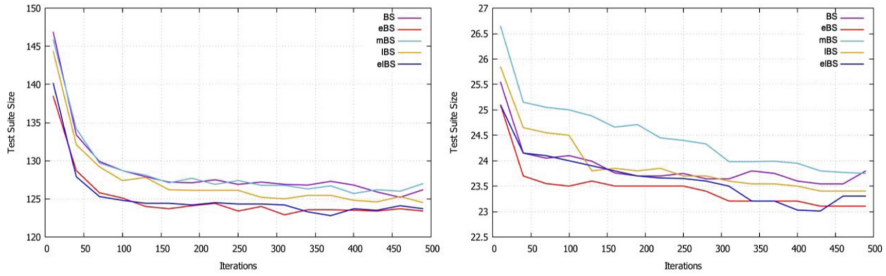

Fig. 5. The Hybrid Variants of Buffalo Strategy's Convergence Rates for CA (N; 2, 10^5) & (N; 2, 4^6)

As can be seen in Fig. 4, hybridizing ABO improved its convergence capacity; the eBS and elBS converged faster than the other variants. This result shows that the addition of an elitism component improved ABO's result, speed and convergence capacity. This is because the elitism mechanism ensures the retention of good solutions for the next generation/iteration of the algorithm. Again, the elitism component makes for efficient randomizations through the replacement of poorer quality solutions with new random solutions.

3.6 More Experimental Evaluations and Comparative Analysis of Results

To further validate our earlier findings (see Section four), we investigated the efficiency cum effectiveness of the selected strategy (the eBS algorithm) by comparing its results with the existing t-way strategies in terms of test suite size. The outcome of this investigation is displayed in the tables and graphs below. Please note that all the experiments in this study were performed on Core i7–3770 CPU@ 3.40 GHz–3.40 GHz, Windows 7 Operating System. Again, we used several benchmark problems popular with many researchers [39–41]. The eBS parameters are: lpr1 = 0.5, lpr2 = 0.5, maximum iteration = 400, buffalo size = 30 and elitism factor = 0.25. For convenience and reliability, the experiments were divided into four configurations:

1. Comparing proposed strategy with results obtained earlier strategies [13, 40, 41] for the different configuration.
2. Comparing proposed strategy with popular existing strategies for CA (N; t, 2^{10}), where *t* varies from 2 to 10.
3. Comparing proposed strategy with preexisting popular strategies for CA (N; 4, 5^P), where *p* varies from 5 to 10
4. Comparing proposed strategy with existing popular strategies for CA (N; 4, v^{10}), *where v* varies from 2 to 7 The experimental outcomes are presented in Tables 3, 4 and Fig. 5.

Table 3. Comparison with existing strategies (N; t, 2^{10}), CA (N; 4, 5^{P}), P varied from 5 to 10

p	Computational-based Strategies									Meta-heuristic-based Strategies			
	IPOG	ITCH	Jenny	PICT	TConfig	TVG	GTWay	MIPOG	CTEXL	PSO	HSS	CS	eBS
5	908	837	810	773	849	731	625	779	NS	779	751	776	782
6	1239	1074	1072	1092	1128	1027	625	1001	NS	1001	990	991	985
7	1349	1248	1279	1320	1384	1216	1125	1209	NS	1209	1186	1200	1162
8	1792	1424	1468	1532	1595	1443	1384	1417	NS	1417	1358	1415	1329*
9	1793	1578	1643	1724	1795	1579	1543	1570	NS	1570	1530	1562	1486*
10	1965	1791	1812	1878	1971	1714	1643	1716	NS	1716	1624	1731	1622*
11	2091	1839	1957	2038	2122	1852	1722	1902	NS	1902	1860	2062	1787
12	2285	1964	2103	NA	2268	2022	1837	2015	NS	2015	2022	2223	1946

Table 4. Comparison with existing strategies CA (N; t, 2^{10}), CA(N; 4, v^{10}) with v varied from 2 to 7

V	Computational-based Strategies									Meta-heuristic-based Strategies			
	IPOG	ITCH	Jenny	PICT	TConfig	TVG	GTWay	MIPOG	CTEXL	PSO	HSS	CS	eBS
2	49	58	39	43	45	40	46	43	NA	34	37	28	28*
3	241	336	221	231	235	228	224	217	NA	213	211	211	208*
4	707	704	703	742	718	782	621	637	NA	685	691	698	666
5	1965	1750	1719	1812	1878	1917	1714	1643	NA	1716	1624	1731	1622*
6	3935	NA	3519	3735	NA	4159	3514	3657	NA	3880	3475	3894	3329*
7	7061	NA	6462	NA	NA	7854	6459	5927	NA	NA	6398	NA	6261*

In Table 3, different systems configurations were used by simply varying the number of parameters (p), levels of interaction strength (t) and number of value (v) to compare the effectiveness of eBS strategy with existing strategies. We were obliged to use N/A in instances where comparative results were unavailable (such as the results for some tools like the mAETG, AETG and CS).

The results presented in Table 3 show the minimum test suite size obtained by existing strategies for different wellknown problems. All the systems addressed only small measures of interaction strength (i.e. $t = 2$ and $t = 3$) because most computational-based strategies are limited in terms of range of interaction strength. In this experiment, eBS produced the smallest test suite size in two cases, CA (N; 2, 3^4), and CA (N; 2, 5^{10}). Even though the eBS was unable to produce the smallest size for all cases (See Fig. 5), the novel strategy outperformed popular strategies such as PSO, ACO, CS and HSS in many cases. In general, it is safe to say that the eBS produced very competitive results since it matched the performance of GA, outperformed HSS, CS, ACO and PSO but was only outperformed by SA.

Furthermore, Table 3 shows that, in most cases, computational-based strategies (such as mAETG, AETG, IPOG, TVG and Jenny) performed worse (with just two optimal results) than the meta-heuristic-based strategies (with 12 optimal solutions): ACO, GA,

PSO, eBS etc. Aside from the meta-heuristic-based strategies, the mAETG strategy obtained close to optimum results in a few cases; while TVG had the worst results. The table also shows SA, ACO and GA (see Problems 81, 86, 88-s14) generated some good results in most cases whereas PSO, HSS, CS, and eBS (See Problems no 81, 83, 85) generated some good results too. In fact, eBS had the optimum result in two instances (see the shaded portions).

In addition, Tables 4 show the capacity of the eBS to address the problems of software testing with a high configuration setting. In these experiments, we increased the three parameter values of covering array v, *t,* and *p.* As can be seen in the tables, most of the existing strategies were unable to produce good results beyond $t > 6$ due to their heavy computation costs (as in the case of GA, ACO GA and PSO). Referring to the results, eBS produced the optimum results in four cases (i.e. CA (N; 10, 2^{10}), CA (N; 3, 2^{10}), CA (N; 4, 2^{10}), CA (N; 7, 2^{10})). The eBS generated the best results in three out of the eight instances. There were another three instances (see the shaded portions) where the eBS produced the best results among other comparative metaheuristics.

Finally, Table 4 reports results for CA (N; 4, 5^{P}) where P is varied from 5 to 10. The comparative results indicate that eBS outperformed the other strategies in five out of the six cases, when p is equal to 8, 9 and 10 respectively (See CA (N; 4, 5^{8}), CA (N; 4, 5^{9}) and CA (N; 4, 5^{10}). In addition, the comparative performance of the eBS with existing strategies, when *v* is varied from 2 to 7 (Table 4), shows that eBS produced satisfactory test sizes for all cases. In all, it is safe to conclude that the proposed strategy outperformed the existing metaheuristic strategies or at least produced competitive results equal to the best results generated by other strategies.

4 Conclusion

In this paper, we proposed and evaluated a new t-way test suite strategy based on African Buffalo Optimization, called the Buffalo Strategy (BS). Next, we investigated four hybridization variants of the BS. The hybridization variants were obtained by employing the concept of component-grafting such as Elitism Technique, Mutation operator and Local Search into the BS. Experimental outcomes indicated that the Elitism-BS (eBS) outperformed the other variants. The experimental output of the eBS was compared with those from the existing strategies in the context of t-way test suite generation. The results of eBS were very competitive. In some cases, the eBS outperformed the other strategies; while, in a few other cases, the proposed strategy failed to produce optimum results. Despite this, the results of eBS were still within an acceptable range. In the future, we recommend further enhancement to improve the efficiency and effectiveness of eBS not just by incorporating algorithm components but full hybridization with other algorithms, such PSO, GA, ACO etc.

Acknowledgment. The authors appreciate the Research Fund by Universiti Tun Hussein Onn, Malaysia and the UMP Short Term Grant RDU1903372 for additional supporting under Fundamental Research Grant Scheme, RDU190185 with Reference no: FRGS/1/2018/ICT03/UMP/02/3, and UMP Short Term Grant RDU1903122.

References

1. Odili, J.B., Mohmad , M.N.: Solving the traveling salesman's problem using the African buffalo optimization. Comput. Intell. Neurosci. **2016**, 1–12 (2016)
2. Odili, J.B., Mohmad Kahar, M.N.: African buffalo optimization approach to the design of PID controller in automatic voltage regulator system. In: National Conference for Postgraduate Research, Universiti Malaysia Pahang, pp. 641–648 (2016)
3. Stardom, J.: Metaheuristics and The Search For Covering and Packing Arrays. Trent University (2001)
4. Özcan, E., Mısır, M., Ochoa, G., Burke, E.K.: A Reinforcement Learning: Great-Deluge Hyper-Heuristic, Modeling, Analysis, and Applications in Metaheuristic Computing: Advancements and Trends: Advancements and Trends, p. 34 (2012)
5. Nurmela, K.J.: Upper bounds for covering arrays by Tabu search. Discrete Appl. Math. **138**, 143–152 (2004)
6. Odili, J.B., Fatokun, J.O.: The mathematical model, implementation and the parameter-tuning of the African buffalo optimization algorithm. In: 2020 International Conference in Mathematics Computer Engineering and Computer Science (ICMCECS), pp. 1–8 (2020)
7. Ahmed, B.S., Zamli, K.Z., Lim, C.P.: Constructing A T-way interaction test suite using the particle swarm optimization approach. Int. J. Innov. Comput. Inf. Control **8**, 431–452 (2012)
8. Grötschel, M., Lovász, L., Schrijver, A.: Geometric algorithms and combinatorial optimization vol. 2: Springer Science & Business Media (2012)
9. Ausiello, G., Crescenzi, P., Gambosi, G., Kann, V., A., Protasi, M.: Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties. Springer Science & Business Media, Heidelberg (2012). https://doi.org/10.1007/978-3-642-58412-1
10. Odili, J.B., Kahar, M.N.M., Anwar, S., Azrag, M.A.K.: A comparative study of African buffalo optimization and randomized insertion algorithm for asymmetric travelling salesman's problem. In: 2015 4th International Conference on Software Engineering and Computer Systems (ICSECS), pp. 90–95 (2015)
11. Odili, J.B., Kahar, M.N.M.: Numerical function optimization solutions using the African buffalo optimization algorithm (ABO). Br. J. Math. Comput. Sci. **10**, 1–12 (2015)
12. Myers, G.J., Sandler, C., Badgett, T.: The art of software testing. Wiley, New York (2011)
13. Alsewari, A.R.A., Zamli, K.Z.: Design and implementation of a harmony-search-based variable-strength t-way testing strategy with constraints support. Inf. Softw. Technol. **54**, 553–568 (2012)
14. Song, P.H., A., Robbins, J., McCullough, J.S.: Exploring the business case for ambulatory electronic health record system adoption. J. Healthcare Manage. **56**, 169 (2011)
15. Garvin, B.J., Cohen, M.B., Dwyer, M.B.: Evaluating improvements to a meta-heuristic search for constrained interaction testing. Empir. Softw. Eng. **16**, 61–102 (2011). https://doi.org/10.1007/s10664-010-9135-7
16. Sugitani, T., Hamasaki, T., Hamada, C.: Partition testing in confirmatory adaptive designs with structured objectives. Biometrical J. **55**, 341–359 (2013)
17. Arcuri, A., Iqbal, M.Z., Briand, L.: Random testing: theoretical results and practical implications. IEEE Trans. Softw. Eng. **38**, 258–277 (2012)
18. Lee, C.-C., Friedman, J.: Requirements modeling and automated requirements-based test generation. SAE Int. J. Aerosp. **6**, 607–615 (2013)
19. Petke, J., Yoo, S., Cohen, M.B., Harman, M.: Efficiency and early fault detection with lower and higher strength combinatorial interaction testing. In: Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering, pp. 26–36 (2013)

20. Gargantini, A., Vavassori, P.: Citlab: a laboratory for combinatorial interaction testing. In: 2012 IEEE Fifth International Conference on Software Testing, Verification and Validation, pp. 559–568 (2012)
21. Burr, K., Young, W.: Combinatorial test techniques: table-based automation, test generation and code coverage. In: Proceedings of the International Conference on Software Testing Analysis and Review (1998)
22. Yilmaz, C., Cohen, M.B., Porter, A.A.: Covering arrays for efficient fault characterization in complex configuration spaces. IEEE Trans. Softw. Eng. **32**, 20–34 (2006)
23. Odili, J.B.: Implementation analysis of Cuckoo search for the benchmark Rosenbrock and Levy test functions. J. Inf. Commun. Technol. **17**, 17–32 (2020)
24. Neshat, M., Sepidnam, G., Sargolzaei, M., Toosi, A.N.: Artificial fish swarm algorithm: a survey of the state-of-the-art, hybridization, combinatorial and indicative applications. Artif. Intell. Rev. **42**(4), 965–997 (2012). https://doi.org/10.1007/s10462-012-9342-2
25. Yang, X.-S., Press, L.: Nature-Inspired Metaheuristic Algorithms, 2nd edn. Luniver Press, London (2010)
26. Qiu, X., Lau, H.Y.: An AIS-based hybrid algorithm for static job shop scheduling problem. J. Intell. Manuf. **25**, 489–503 (2014). https://doi.org/10.1007/s10845-012-0701-2
27. Odili, J.B., M.N., Noraziah, A., Abiodun, O.E.: African buffalo optimization and the randomized insertion algorithm for the asymmetric travelling salesman's problems. J. Theor. Appl. Inf. Technol. **87**, 356–364 (2016)
28. Odili, J.B., M.N.: African buffalo optimization. Int. J. Softw. Eng. Comput. Syst. **2**, 28–50 (2016)
29. Odili, J.B., Kahar, M.N.M., Anwar, S.: African buffalo optimization: a swarm-intelligence technique. Procedia Comput. Sci. **76**, 443–448 (2015)
30. Odili, J.B., Noraziah, A., Ambar, R., Abd Wahab, M.H.: Implementation strategies for the Cuckoo search and the African buffalo optimization for the benchmark Rosenbrock function. In: The Eurasia Proceedings of Science Technology Engineering and Mathematics, pp. 395–402
31. Odili, J.B., Kahar, M.N.M.: African buffalo optimization. Int. J. Softw. Eng. Comput. Syst. **2**, 28–50 (2016)
32. Odili, J.B., Kahar, M.N.M.: African buffalo optimization (ABO): a new meta-heuristic algorithm. J. Adv. Appl. Sci. **3**, 101–106 (2015)
33. Malik, M.S., Wadhwa, M.S.: Preventing premature convergence in genetic algorithm using DGCA and elitist technique. Int. J. **4** (2014)
34. Gong, W., Cai, Z.: Differential evolution with ranking-based mutation operators. IEEE Trans. Cybern. **43**, 2066–2081 (2013)
35. Rodrigues, E., Gaspar, A.R., Gomes, Á.: An evolutionary strategy enhanced with a local search technique for the space allocation problem in architecture, Part 1: Methodology. Comput. Aided Des. **45**, 887–897 (2013)
36. Shiba, T., Tsuchiya, T., Kikuno, T.: Using artificial life techniques to generate test cases for combinatorial testing. In: International Computer Software and Applications Conference, pp. 72–77 (2004)
37. Nasser, A.B., Alsewari, A.R.A., Zamli, K.Z.: Tuning of cuckoo search based strategy for T-way testing. In: International Conference on Electrical and Electronic Engineering (2015)
38. Khompatraporn, C., Pintér, J.D., Zabinsky, Z.B.: Comparative assessment of algorithms and software for global optimization. J. Global Optim. **31**, 613–633 (2005). https://doi.org/10.1007/s10898-004-9971-3
39. Alsewari, A.R.A., Zamli, K.Z.: Design and implementation of a harmony-search-based variable-strength t-way testing strategy with constraints support. Inf. Softw. Technol. **54**, 553–568 (2012)

40. Ahmed, B.S., Abdulsamad, T.S., Potrus, M.Y.: Achievement of minimized combinatorial test suite for configuration aware software functional testing using the cuckoo search algorithm. Inf. Softw. Technol. **66**, 13–29 (2015)
41. Cohen, M.B.: Designing Test Suites for Software Interaction Testing. Citeseer (2004)