Learning Cuckoo Search Strategy for t-way Test Generation

Abdullah B. Nasser, Abdulrahman Alsewari, and Kamal Z. Zamli(✉)

Faculty of Computer Systems and Software Engineering,
Universiti Malaysia Pahang, 26300 Kuantan, Pahang, Malaysia
Abdullahnasser83@gmail.com, alsewari@gmail.com,
kamalz@ump.edu.my

Abstract. The performance of meta-heuristic algorithms highly depends on their exploitation and exploration techniques. In the past 30 years, many meta-heuristic algorithms have been developed which adopts different exploitation and exploration techniques. Several studies reported that the hybrid of meta-heuristics algorithms often perform better than its corresponding original algorithm. This paper presents a new hybrid algorithm; called Learning Cuckoo Search (LCS) strategy based on the integration student phase from Teaching Learning based Optimization (TLBO) Algorithm. To evaluate the developed algorithm, we use the problem of t-way test generation as our case study. The experiment results show that LCS has better performance as compared as to the original Cuckoo Search as well many other existing strategies.

1 Introduction

The performance of meta-heuristic algorithms highly depends on their search technique capabilities. Often, the performance of meta-heuristic algorithms depends on their exploitation and exploration strategy. Exploitation explores the promising regions in the hope to find better solutions while the exploration ensures that all regions of the search space have been visited. Usually, a good balance between intensive and efficient exploration plays an important part in the performance of meta-heuristic algorithms [1].

Many meta-heuristic algorithms have been developed in the past 30 years such as Tabu search (TS) [2], Simulated Annealing (SA) [3], Genetic Algorithm (GA) [4], Particle Swarm Optimization (PSO) [5], Ant Colony Optimization (ACO) [6], Differential Evolution (DE) [7], Harmony Search HS [8], Flower Pollination Algorithm (FPA) [9], Sine Cosine Algorithm (SCA) [10], Bat Algorithm (BA) [11], Cuckoo Search (CS) [12] Teaching–Learning-Based Optimization Algorithm (TLBO) and Firefly Algorithm (FA) [13]. Meta-heuristic algorithms have been successfully used for solving a wide range of software engineering problems on software engineering management, requirements engineering, design, testing, and refactoring.

Concerning exploitation and exploration techniques, GA, for instance, adopts selection, and crossover, mutation operations. PSO adopts as two search approach: search around overall best (gbest) and search around personal best (pbest). CS depends on Levy flight and elitism technique. TLBO divides the search into teacher and learner phases.

R. Sharma et al. (Eds.): ICAN 2017, CCIS 805, pp. 97–110, 2018.
https://doi.org/10.1007/978-981-13-0755-3_8

Each algorithm has its strengths and limitations as there is no single algorithm that is superior for all optimization problems. For these reasons, the search for a new algorithm is justified by developing a new meta-heuristics algorithm. Experiments studies reported that the hybrid of meta-heuristics algorithms often perform better than its corresponding origin algorithm [14]. For example, He et al. demonstrated a hybridized Variable Neighbourhood search with TS to minimize the discrete time/cost trade-off problem [15]. Yildiz presented a new hybrid algorithm based on Hill Climbing local search and Artificial Immune Algorithm for solving general optimization problem [16]. Wang et al. adopted SA and GA algorithms to optimize the cutting conditions in plain milling [17].

Building from the aforementioned prospect, the main focus of this work to present a new hybrid algorithm, called Learning Cuckoo Search strategy (LCS), based on the integration of Cuckoo search with the student phase from Teaching Learning based Optimization algorithm (TLBO). Our hybridization approach is unique from existing hybridizations of CS as we use the peer learning phase operator during the elitism phase of the Cuckoo search algorithm. As a case study, we adopt LCS for the t-way test generation problem. In a nutshell, t-way test generation problem is a sampling technique to select a sub-set of test cases that can be used to test overall system such that every t combination of input values (where t refer to interaction strength) is covered at least one time [18].

To this end, much recent works on t-way strategies are focusing using a single meta-heuristic algorithm (such as TS, SA, GA, ACA, PSO, and HS, to name a few [18–21]). Although useful, existing work has not sufficiently dealt with hybrid meta-heuristic algorithm (i.e. combinations of two or more meta-heuristic algorithms) as the backbone for t-way strategies. Taking this challenge has led us toward the current work.

The organization of this paper is as follows. In Sect. 2, an overview on how t-way testing works is provided. Section 3 reviews existing t-way strategies. Section 4 describes the proposed strategy, while Sect. 5 highlights and discusses the results. Lastly, Sect. 6 gives the conclusion and future work.

2 Problem Definition of t-way Testing

To illustrate the concept of t-way testing, consider a Car Ordering System (COS) example. The system allows for buying and selling cars in Malaysia. The system allows two types of booking; online booking or in store during Opening hours or closing hours. Here, this system consists of five inputs (i.e. Order category, Location, Car brand, Order type and Order time), three parameters with two values, one parameter with nine values, and one parameter with one value. The system can be summarized in Table 1.

In COS, there are 72 combinations need to be tested for ideal testing (i.e. exhaustively testing which considers all interaction strength, $t = 5$). By considering two-way interaction, the reduction of test suite size can be achieved 72 to 9 test cases as shown in Table 2.

Table 1. Car ordering system parameters

Order category	Location	Car brand	Order type	Order time
Buy	Kuala Lumpur	Perodua	E-Booking	Opening hours
Selling	Penang	Toyota	In store	Closing hours
	Johor Bahru	Proton		

Table 2. Two-way test suite for car ordering system

No	Order category	Location	Car brand	Order type	Order time
1	Buy	Kuala Lumpur	Perodua	E-Booking	Opening hours
2	Selling	Penang	Perodua	In store	Closing hours
3	Buy	Johor Bahru	Toyota	In store	Opening hours
4	Selling	Johor Bahru	Proton	E-Booking	Closing hours
5	Buy	Penang	Toyota	E-Booking	Closing hours
6	Selling	Kuala Lumpur	Proton	In store	Opening hours
7	Buy	Penang	Proton	E-Booking	Opening hours
8	Selling	Kuala Lumpur	Toyota	In store	Closing hours
9	Selling	Johor Bahru	Perodua	E-Booking	Closing hours

The same generalization can be done for 3-way and so forth. It is up to the creativity and the knowledge of the test engineers to decide on the right *t* value based on the testing requirement at hand. Researchers often advocate the best range of values for *t* is from 2 to 6.

3 Related Work

Generally speaking, there are two main domains of the existing works on *t*-way testing: algebraic and computational methods [22]. Algebraic methods construct based on lightweight mathematical functions without enumerate any combinations. Strategies of this approach (e.g. Combinatorial Test Services (CTS) strategy and TConfig [23]) are restricted for small configurations $(t \leq 3)$.

Computational methods offer flexibility to address large configuration. As the name suggests, computational approaches use pure computation strategies or meta-heuristic algorithms to construct the test cases. Strategies adopting computational methods can be categorized into two categories: one-test-at-a-time (OTAT) and one-parameter-at-a-time (OPAT) strategies. Computational methods mainly based on generating all possible combinations (i.e. interaction elements) according to system's configurations.

OPAT strategies start by constructing a completed test suite for the smallest interaction parameters, then in every iteration one parameter (i.e. one column) are added until all the parameters are covered. In-parameter-order (IPO) [24] is the pioneer work in this respect. Many improvements to the basis of IPO strategy have been developed such as that of IPOG-D [25], IPOG [26], IPOF and IPAD2 [27] to obtain the smallest test sizes and fast execution times.

Concerning OTAT strategies, a complete test case is constructed per iteration that covers the maximum number of uncovered interaction elements. The same procedure is repeated until all interaction elements are covered. In the literature, numerous tools and strategies have developed based on OTAT approach such as Automatic Efficient Test Generator (AETG) [28], Classification-Tree Editor eXtended Logics (CTE-XL) [29], Pairwise Independent Combinatorial Testing (PICT) [30], Deterministic Density Algorithm (DDA) [31, 32], Test Vector Generator (TVG) [33], GTWay [34], Jenny [35], and WHITCH [36].

Recently, meta-heuristic algorithms have been adopted as the backbone for t-way test suite generation. In general, meta-heuristic-based strategies start with a random set of solutions. These solutions are subjected to a series of search operation in an attempt to improve them. During each iteration, the best candidate solution is selected and added to the final test suite. In the literature, many meta-heuristic algorithms have been successfully applied for t-way testing such as TS [37], SA [38], GA [39], ACA [39], PSO [22, 40], HS [18], and CS [41].

Stardom [20] presented a description of adopting SA, GA and TS algorithms for two-way testing. SA [38] is a single-based physical Algorithm, inspired from the physical annealing process. The algorithm starts searching from one position and then employ neighborhood search in a local region in attempt to find better solution. SA allows moving to poor solution with acceptance probability to avoid stuck in a local minimum solution. GA is an early algorithm for adopting a population-based algorithm in t-way testing. it starts finding optimal test case from many positions and then repeated apply selection, crossover, and mutation operations in order to mimic natural selection of biological evolution. Similar to SA, TS accept a worse move if no improving move is available. TS utilizes memory structures (termed Tabu list) for guiding the search process (i.e. to remember the visited solutions).

Later on, Cohen extended SA to support 3-way interaction testing [42], and Shiba et al. extended GA and ACO to support 3-way interaction testing [43]. ACO mimics the behavior of colonies of ants for finding food paths. Here, each test case represents the quality of the paths to the food and the food represents the value of the parameter. ACA use trails of a chemical substance, called as pheromone which reinforce over time. Pheromone enables other ants to find short paths of the food source. Comparative experiments between SA, TS and GA conducted by Colbourn et al. demonstrate that SA performs better than TS and GA [38].

Chen et al. [44] implemented PSO algorithms for 2-way testing and [21] for t-way testing. The algorithm mimics the swarm behavior of bird and fish swarm in searching food. Based on simple formulae, the population (i.e. called a swarm) moves in the search space, guided by global best and personal best in attempt to find better solution. Recently, [18] adopted HS for design and implementation a new t-way strategy called Harmony Search Strategy (HSS). HS is inspired by the behavior of musicians, to produce a new musical tone. The strategy support high interaction strengths (i.e. $t \geq 15$).

Nasser et al. [45] have implemented the Flower pollination based strategy (FPA) for generating t-way test generation. FPA is also used for generating sequence t-way test suite [46]. Inspired by the pollination behavior of flowering plants, FPA can be represented as two steps (i.e. Global Pollination and Local Pollination), controlled by probability parameter. Global Pollination step exploits lévy flight to transfer the pollen to another flower while local pollination transfers the pollen to female part within the same flower. In similar work, Alsariera adopted Bat Algorithm for *t*-way testing [47, 48]. The algorithm is inspired by the hunting behavior of Microbats which are able to find its prey in complete darkness.

Recently, Zamli et al. [49] proposed 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); Particle Swarm Optimization, Teaching Learning based Optimization, Cuckoo Search Algorithm and Global Neighborhood 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 [50] with new acceptance mechanism based on fuzzy inference system and new LLH operations (i.e. GA's crossover search operator of, TLBO's learning search operator, FPA's global Pollination, and Jaya algorithm's search operator).

Building from earlier approach, Zamli et al. presented [51] a new strategy, called Adaptive Teaching Learning-Based Optimization (ALTBO). ATLBO improves the performance of standard TLBO resulting from a good balance between intensification and diversification through the adoption of fuzzy inference rules.

4 Proposed Strategy

This section describes the Learning Cuckoo Search Strategy (LCS) strategy based on the hybridization of Cuckoo Search with the student phase of Teaching Learning based Optimization algorithm.

4.1 Cuckoo Search

CS is one of the latest nature inspired algorithms inspired from brood parasitic behavior of Ani and Guira cuckoos [12]. Cuckoo has an aggressive reproduction strategy in that they lay their eggs in the nests of other host birds. In order to increase the hatching probability of their own eggs, they often remove the eggs of the host bird.

Concerning its corresponding algorithm, CS initially generates a population of nests randomly. The nest is updated using Levy Flight, based on the following equation:

$$x_i^{(t+1)} = x_i^{(t)} + \alpha \oplus L\acute{e}vy(\lambda) \tag{1}$$

To mimic the removal of host eggs, CS proposes elitism technique which is controlled by the probability (*pa*). Figure 1 summarizes the complete CS algorithm.

Algorithm 1: Cuckoo Search Algorithm

```
1:  Objective function f(x), x = (x1, ..., xd) ;
2:  Initial a population of n host nests $x_i$ $(i = 1, 2, ..., n)$;
3:  While (t <MaxGeneration) or (stop criterion)
4:      Get a cuckoo (say i) randomly by Lévyflights;
5:      Evaluate its quality/fitness Fi;
6:      Choose a nest among n (say j) randomly;
7:      IF (Fi > Fj)
8:          Replace j by the new solution;
9:      End if
10:     Abandon a fraction (pa) of worse nests and build
11:     new ones.
12:     Keep the best solutions (or nests with quality solutions);
13:     Rank the solutions and find the current best;
14: End while
15: Postprocess results and visualization;
16: End-Procedure
```

Fig. 1. Cuckoo search algorithm [12]

4.2 Teaching-Learning Based Optimization

Teaching-Learning Based Optimization (TLBO) algorithm is proposed by Rao, et al. [52]. The algorithm inspired from the classroom setting between a teacher and learners. In general, TLBO can view as two phases: Teacher Phase and Learner Phase. Here, the student can learn from the teacher or from his/her partner. In TLBO, the population consider as two groups teachers and learners. At each iteration, TLBO undergoes the two phases sequentially.

Teacher Phase attempts to improve individual solution x_i, by moving their position toward their teacher (Eq. 2):

$$x_i^{(t+1)} = x_i^{(t)} + r\left(x_{teacher} - T_f * x_{mean}\right) \tag{2}$$

Learner Phase represents local search's part in TLBO. This phase attempts to explore the solution around each individual. Each individual x_i improves its knowledge by interacting with its random peer x_i then move its position to new learner's position. The individual moves to learner, if only there is improvement (Eq. 3), otherwise move toward Eq. 4:

$$x_i^{(t+1)} = x_i^{(t)} + r\left(x_j - x_i\right) \tag{3}$$

$$x_i^{(t+1)} = x_i^{(t)} + r\left(x_i - x_j\right) \tag{4}$$

4.3 Learning Cuckoo Search

LCS is a composition of two main steps: generating interaction elements which represent the search space and find optimal t-way test suite using meta-heuristic algorithm.

As mentioned previously, the core parts of the CS algorithm are generating new solutions using Levy Flight and replacing the worst nests by new nest using elitism technique. Although CS has good capability for exploration the search space efficiently, many studies have shown CS has local conversion issues as the algorithm can easily be trapped in local minima [53–55]. Addressing this issue, the proposed strategy integrates CS with learner phase from TLBO. Original CS replaces the worst by new nest generated randomly. Our proposed strategy generates the new nests as part of elitism based on learner phase of TLBO. Figure 2 illustrates the proposed Learning Cuckoo Search Algorithm (LCS).

```
Algorithm 2: Learning Cuckoo Search
Input: Parameter number p, and
       set of values for each parameter $V = [v0..vj]$;
Output: Final test suite List FTS;
Beging
1:  Let FTS be a set of candidate tests;
2:  Generate interactions elements EL based on P and V
3:  Generate initial population of nests randomly
4:  while EL is not empty do
5:  while t <MaxGeneration  or  stop criterion do
6:      Generate a cuckoo (say i) randomly by Lévyflights;
7:       Choose a nest among n (say j) randomly;
8:        IF (Fi > Fj)
9:               Replace j by the new solution;
10:       End if
11:       Find a fraction (pa) of worse nests and replace them
12:       by new nest generated based on learner phase.
13:       Evaluate new solutions
14:       Find the current best solution gbest
15: End while
16: Add the best test case, gbest, into FTS.
17: Remove covered interactions elements from IEL
18: End while
End-Procedure
```

Fig. 2. Learning cuckoo search strategy

5 Results

In order to evaluate the performance of LCS, first, we compare the convergence rate of LCS with its counterparts CS, and then LCS is compared against other existing strategies including CS. The parameters of LCS are set at $Pa = 0.8$, population size = 30 and the maximum number of improvements = 300.

5.1 Convergence Rate Analysis

In order to evaluate the performance of the proposed strategy against the original CS, the convergence rate for different systems is studied. Here, convergence rate is used to measure how fast meta-heuristic algorithms converge (i.e. covering all the required interactions) per generation. We apply CS and LCS on two systems as shown in Table 3. System column refers to system configuration in that y^x means that this system has x parameters, each parameter with y values. In this experiment, both of CS and LCS are implemented and executed using NetBeans 8.0.1. Different values of iteration (i.e. 5, 10, 20, 30, 40, 50, 100, 200, 300, 500, and 1000) are used to measure the convergence rate.

Table 3. Description of three problems

No.	Systems	t	Description
1	10^5	2	System with 5 parameters, each have 10 values
3	7^3, 5^2, 3^1	3	System with 6 parameters, 3 parameters have 2 values, 2 parameters have 5 values, and 1 parameter has 3 values

Figures 3 and 4 show a comparison of convergence rate between CS and modified LCS for two problems For quickly identify the different between the results of CS and LCS, we add test suite size labels (Y points) to the figures, while X points take the same values of X aixs. As the figures show, LCS outperforms CS in the two problems. In this case, by introducing elitism-learning technique component into origin CS, we observe that the quality of solutions, in LCS strategy, is improved and convergence rate becomes faster.

5.2 Performance Evaluation

This section compares LSC's results with published results in [18, 21, 41]. Different systems configurations have been used have been adopted in this comparison (see Tables 4 and 5).

The cells marked as NA indicates "Not Available results", while NS indicates "Not Support". Tables 4 and 5 present a comparison of LCS against the existing strategies. The cells marked with bold font present the optimal value achieved by the strategy on the corresponding column. Most of existing strategies, in Table 3, support only t ≤ 3

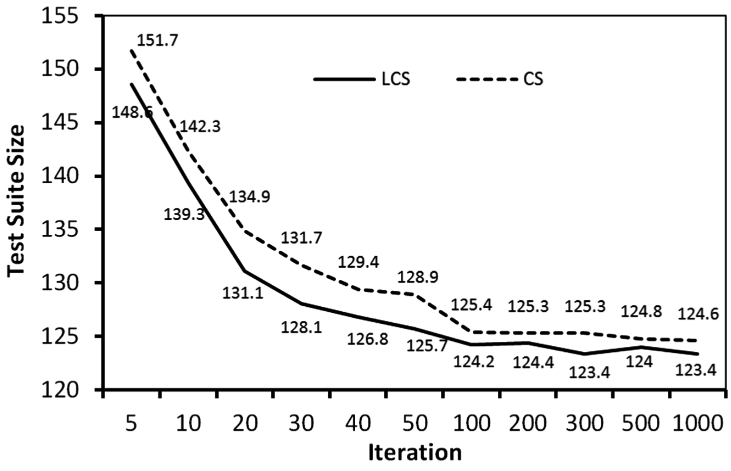

Fig. 3. Convergence rate of CS and LCS for 10^5 with $t = 2$

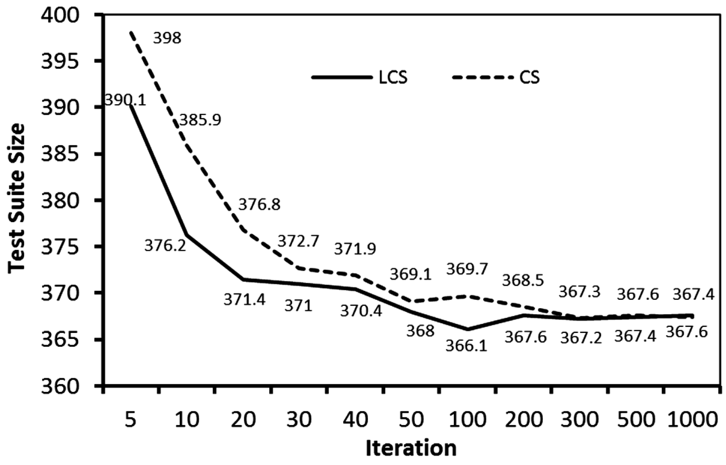

Fig. 4. Convergence rate of CS and LCS for $7^3, 5^2, 3^1$ with t $= 3$

such as GA, SA, and ACO, while Table 4 includes strategies support higher interaction strength $(t > 3)$ such as PSO, HS, and CS. Results in Table 4 shows that LCS performs better than other strategies (i.e. 6 out of 14 cases), while the worst results have been obtained by TVG followed by AETG and IPOG.

Table 5 show that LCS obtains the smallest test suite in almost all cases, only HS has managed to outperform LCS in one case (i.e. case No. 13). Both of LCS and HS produce equal results in 4 cases as marked with bold font. Comparing CS with LCS only, the results in Table 4 show that the results of LCS are much better than standard CS.

Table 4. Comparison with existing strategies $(t \leq 3)$

No.	Systems	t	AETG	IPOG	Jenny	TVG	SA	ACO	GA	LCS
1	3^4	2	**9**	**9**	10	11	**9**	**9**	**9**	**9**
2	3^{13}	2	**15**	20	20	19	16	17	17	18
3	10^{10}	2	NA	176	**157**	208	NA	159	157	**157**
4	5^{10}	2	NA	50	45	51	NA	NA	NA	**42**
5	8^{10}	2	NA	117	104	124	NA	NA	NA	**102**
6	15^{10}	2	NA	373	**336**	473	NA	NA	NA	352
7	3^{10}	2	NA	20	19	18	NA	NA	NA	**16**
8	3^6	3	47	53	51	49	**33**	**33**	**33**	39
9	4^6	3	105	**64**	112	123	**64**	**64**	**64**	101
10	5^6	3	NA	216	215	234	152	**125**	**125**	196
11	6^6	3	343	382	373	407	**300**	330	331	335
12	7^{10}	2	NA	90	83	98	NA	NA	NA	**79**
13	$7^1\ 6^1\ 5^1\ 4^6\ 3^8\ 2^3$	3	45	43	50	51	**42**	**42**	**42**	50
14	$5^2\ 4^2\ 3^2$	3	NA	111	131	136	**100**	106	108	118

Table 5. Comparison with existing strategies

No.	Systems	t	PSO	HS	CS	LCS
1	2^{10}	2	8	**7**	8	8
2	3^{10}	2	17	NA	17	**16**
3	2^{10}	3	17	**16**	**16**	**16**
4	6^6	3	42	**39**	43	**39**
5	5^7	4	1209	1186	1200	**1177**
6	5^8	4	1417	1358	1415	**1329**
7	2^{10}	5	82	81	79	**75**
8	3^7	5	441	NA	439	**437**
9	2^{10}	6	158	158	157	**154**
10	3^7	6	977	NA	973	971
11	2^{10}	7	NS	298	NS	**292**
12	2^{10}	8	NS	498	NS	**501**
13	2^{10}	9	NS	**512**	NS	587
14	2^{10}	10	NS	**1024**	NS	**1024**

Figure 5 shows the comparison results between CS and LCS. LCS strategy appears to generate the optimum results in most cases owing to enhancement of its elitism (i.e. based on learning from other solution instead of purely on random basis).

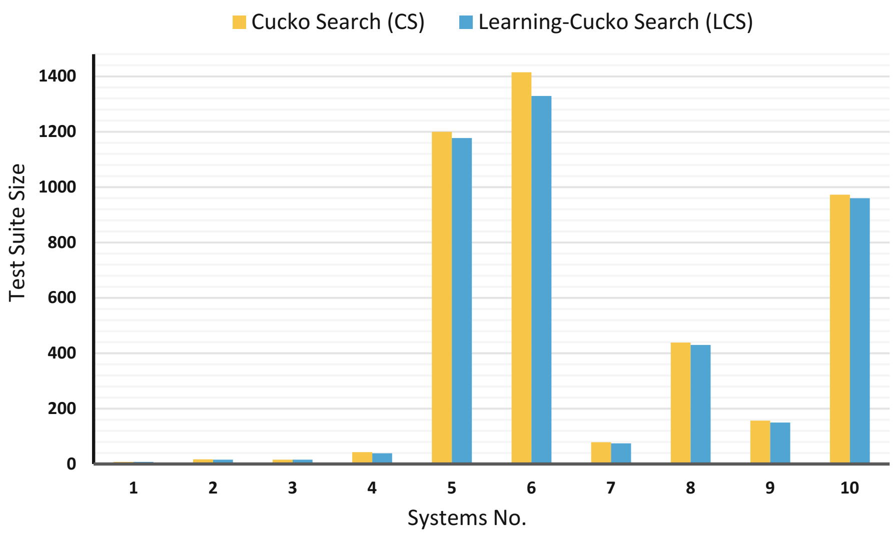

Fig. 5. CS and LCS test suite size comparison

6 Conclusion and Future Work

In this paper, LCS is adopted for design and implementation a new *t*-way strategy for test generation. The strategy utilizes the TLBO's learner phase as part of the Cuckoo elitism process. Initial results show that LCS is able to outperform some of existing meta-heuristic based strategies including origin CS.

Owing to its promising results, we intend to extend this work to apply LCS on different *t*-way interaction possibilities such as cumulative strength interaction, variable strength interaction, and input output based relation. As part of the future work, we plan to improve the design of the LCS by hybridize LCS with another meta-heuristic to improve its overall search capabilities.

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

References

1. Yang, X.S., Deb, S., Fong, S.: Metaheuristic algorithms: optimal balance of intensification and diversification. Appl. Math. Inf. Sci. **8**, 977–983 (2013)
2. Glover, F., Laguna, M.: Tabu Search. Springer, New York (1999)
3. Kirkpatrick, S.: Optimization by simulated annealing: quantitative studies. J. Stat. Phys. **34**, 975–986 (1984)

4. Holland, J.H.: Genetic algorithms. Sci. Am. **267**, 66–72 (1992)
5. Kennedy, J.: Particle swarm optimization. In: Sammut, C., Webb, G.I. (eds.) Encyclopedia of Machine Learning, pp. 760–766. Springer (2011)
6. Dorigo, M., Birattari, M., Blum, C., Clerc, M., Stützle, T., Winfield, A.: Ant colony optimization and swarm intelligence. In: 6th International Conference, Ants 2008, Brussels, Belgium. Springer, Heidelberg (2008). https://doi.org/10.1007/11839088
7. Feoktistov, V.: Differential Evolution. Springer, New York (2006)
8. Lee, K.S., Geem, Z.W.: A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice. Comput. Methods Appl. Mech. Eng. **194**, 3902–3933 (2005)
9. Yang, X.S.: Flower pollination algorithm for global optimization. In: Durand-Lose, J., Jonoska, N. (eds.) Unconventional Computation and Natural Computation. UCNC 2012. Lecture Notes in Computer Science, vol. 7445, pp. 240–249. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32894-7_27
10. Mirjalili, S.: SCA: A sine cosine algorithm for solving optimization problems. Knowl.-Based Syst. **96**, 120–133 (2016)
11. Pham, D., Ghanbarzadeh, A., Koc, E., Otri, S., Rahim, S., Zaidi, M.: The bees algorithm–a novel tool for complex optimisation. In: Intelligent Production Machines and Systems-2nd I* PROMS Virtual International Conference, 3–14 July 2006 (2011)
12. Yang, X.-S., Deb, S.: Cuckoo search via lévy flights. In: 2009 World Congress on Nature and Biologically Inspired Computing, NaBIC 2009, pp. 210–214. IEEE (2009)
13. Yang, X.S.: Firefly algorithm, lévy flights and global optimization. In: Bramer, M., Ellis, R., Petridis, M. (eds.) Research and Development in Intelligent Systems XXVI, pp. 209–218. Springer, London (2010). https://doi.org/10.1007/978-1-84882-983-1_15
14. Blum, C., Roli, A.: Hybrid metaheuristics: an introduction. In: Blum, C., Aguilera, M.J.B., Roli, A., Sampels, M. (eds.) Hybrid Metaheuristics. Studies in Computational Intelligence, vol. 114, pp. 1–30. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78295-7_1
15. He, Z., He, H., Liu, R., Wang, N.: Variable neighbourhood search and tabu search for a discrete time/cost trade-off problem to minimize the maximal cash flow gap. Comput. Oper. Res. **78**, 564–577 (2017)
16. Yıldız, A.R.: A novel hybrid immune algorithm for global optimization in design and manufacturing. Robot. Comput.-Integr. Manufact. **25**, 261–270 (2009)
17. Wang, Z., Rahman, M., Wong, Y., Sun, J.: Optimization of multi-pass milling using parallel genetic algorithm and parallel genetic simulated annealing. Int. J. Mach. Tools Manufact. **45**, 1726–1734 (2005)
18. 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)
19. Nurmela, K.J.: Upper bounds for covering arrays by tabu search. Discrete Appl. Math. **138**, 143–152 (2004)
20. Stardom, J.: Metaheuristics and the Search for Covering and Packing Arrays. Simon Fraser University, Burnaby (2001)
21. Ahmed, B.S., Zamli, K.Z., Lim, C.P.: Constructing a t-way interaction test suite using the particle swarm optimization approach. Int. J. Innovative Computing, Inf. Control **8**, 431–452 (2012)
22. Chen, X., Gu, Q., Qi, J., Chen, D.: Applying particle swarm optimization to pairwise testing. In: IEEE 34th Annual on Computer Software and Applications Conference (COMPSAC), pp. 107–116. IEEE (2010)
23. Williams, A.: TConfig download page (2008)

24. Lei, Y., Tai, K.-C.: In-parameter-order: a test generation strategy for pairwise testing. In: Proceedings of the 3rd IEEE International Symposium on High Assurance Systems Engineering, pp. 254–261. IEEE Computer Society (1998)
25. Lei, Y., Kacker, R., Kuhn, D.R., Okun, V., Lawrence, J.: IPOG/IPOG-D: efficient test generation for multi-way combinatorial testing. Softw. Test. Verification Reliab. **18**, 125–148 (2008)
26. Lei, Y., Kacker, R., Kuhn, D.R., Okun, V., Lawrence, J.: IPOG: a general strategy for t-way software testing. In: 14th Annual IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS 2007), pp. 549–556. IEEE (2007)
27. Forbes, M., Lawrence, J., Lei, Y., Kacker, R.N., Kuhn, D.R.: Refining the in-parameter-order strategy for constructing covering arrays. J. Res. Nat. Inst. Stand. Technol. **113**, 287 (2008)
28. Cohen, D.M., Dalal, S.R., Fredman, M.L., Patton, G.C.: The AETG system: an approach to testing based on combinatorial design. IEEE Trans. Softw. Eng. **23**, 437–444 (1997)
29. Lehmann, E., Wegener, J.: Test case design by means of the CTE XL. In: Proceedings of the 8th European International Conference on Software Testing, Analysis and Review (EuroSTAR 2000), Kopenhagen, Denmark (2000)
30. Czerwonka, J.: Pairwise testing in the real world: practical extensions to test-case scenarios. In: Proceedings of 24th Pacific Northwest Software Quality Conference, pp. 419–430. Citeseer (2006)
31. Colbourn, C.J., Cohen, M.B., Turban, R.: A deterministic density algorithm for pairwise interaction coverage. In: The International Association of Science and Technology for Development (IASTED) Conference on Software Engineering, pp. 345–352 (2004)
32. Bryce, R.C., Colbourn, C.J.: The density algorithm for pairwise interaction testing. Softw. Test. Verification Reliab. **17**, 159–182 (2007)
33. https://sourceforge.net/projects/tvg/
34. Zamli, K.Z., Klaib, M.F., Younis, M.I., Isa, N.A.M., Abdullah, R.: Design and implementation of a t-way test data generation strategy with automated execution tool support. Inf. Sci. **181**, 1741–1758 (2011)
35. http://www.burtleburtle.net/bob/math
36. Hartman, A., Klinger, T., Raskin, L.: IBM intelligent test case handler. Discrete Math. **284**, 149–156 (2010)
37. Nie, C., Leung, H.: A survey of combinatorial testing. ACM Comput. Surv. (CSUR) **43**, 11 (2011)
38. Colbourn, C.J., Cohen, M.B., Turban, R.: A deterministic density algorithm for pairwise interaction coverage. In: IASTED Conference on Software Engineering, pp. 345–352. Citeseer (2004)
39. Shiba, T., Tsuchiya, T., Kikuno, T.: Using artificial life techniques to generate test cases for combinatorial testing. In: The 28th Annual International Computer Software and Applications Conference, pp. 72–77 (2004)
40. Ahmed, B.S., Zamli, K.Z.: A variable strength interaction test suites generation strategy using particle swarm optimization. J. Syst. Softw. **84**, 2171–2185 (2011)
41. 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)
42. Cohen, M.B.: Designing Test Suites for Software Interaction Testing. University of Auckland (2004)
43. 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. IEEE (2004)

44. Chen, X., Gu, Q., Qi, J., Chen, D.: Applying particle swarm optimization to pairwise testing. In: IEEE 34th Annual Computer Software and Applications Conference, pp. 107–116. IEEE (2010)
45. Nasser, A.B., Alsariera, Y.A., Alsewari, A.R.A., Zamli, K.Z.: Assessing optimization based strategies for t-way test suite generation: the case for flower-based strategy. In: 5th IEEE International Conference on Control Systems, Computing and Engineering, Pinang, Malaysia (2015)
46. Nasser, A.B., Hujainah, F., Alsewari, A.A., Zamli, K.Z.: Sequence and sequence-less t-way test suite generation strategy based on flower pollination algorithm. In: 2015 IEEE Student Conference on Research and Development (SCOReD), pp. 676–680. IEEE (2015)
47. Alsariera, Y.A., Nasser, A.B., Zamli, K.Z.: Benchmarking of Bat-inspired Interaction Testing Strategy
48. Alsariera, Y.A., Zamli, K.Z.: A bat-inspired strategy for t-way interaction testing. Adv. Sci. Lett. **21**, 2281–2284 (2015)
49. Zamli, K.Z., Alkazemi, B.Y., Kendall, G.: A tabu search hyper-heuristic strategy for t-way test suite generation. Appl. Soft Comput. **44**, 57–74 (2016)
50. Zamli, K.Z., Din, F., Kendall, G., Ahmed, B.S.: An experimental study of hyper-heuristic selection and acceptance mechanism for combinatorial t-way test suite generation. Inf. Sci. **399**, 121–153 (2017)
51. Zamli, K.Z., Din, F., Baharom, S., Ahmed, B.S.: Fuzzy adaptive teaching learning-based optimization strategy for the problem of generating mixed strength t-way test suites. Eng. Appl. Artif. Intell. **59**, 35–50 (2017)
52. Rao, R.V., Savsani, V.J., Vakharia, D.: Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems. Comput. Aided Des. **43**, 303–315 (2011)
53. Liu, X., Fu, M.: Cuckoo search algorithm based on frog leaping local search and chaos theory. Appl. Math. Comput. **266**, 1083–1092 (2015)
54. Zhou, Y., Zheng, H.: A novel complex valued cuckoo search algorithm. Sci. World J. **2013** (2013)
55. Li, X., Yin, M.: A hybrid cuckoo search via Lévy flights for the permutation flow shop scheduling problem. Int. J. Prod. Res. **51**, 4732–4754 (2013)