VOL. 10, NO. 19, OCTOBER 2015 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

ON ADOPTING PARAMETER FREE OPTIMIZATION ALGORITHMS FOR COMBINATORIAL INTERACTION TESTING

Kamal Z. Zamli, Yazan A. Alsariera, Abdullah B Nasser and Abdulrahman Alsewari
Faculty of Computer Systems and Software Engineering, Universiti Malaysia Pahang, Lebuhraya Tun Razak, Pahang, Malaysia
E-Mail: kamalz@ump.edu.my

ABSTRACT
Combinatorial interaction testing is a practical approach aims to detect defects due to unwanted and faulty interactions. Here, a set of sampled test cases is generated based on t-way covering problem (where t indicates the interaction strength). Often, the generation process is based on a particular t-way strategy ensuring that each t-way interaction is covered at least once. Much useful progress has been achieved as plethora of t-way strategies have been developed in the literature in the last 30 years. Recently, in line with the upcoming field called Search based Software Engineering (SBSE), many newly strategies have been developed adopting specific optimization algorithm (e.g. Genetic Algorithm (GA), Ant Colony (AC), Simulated Annealling (SA), Particle Swarm Optimization, and Harmony Search Algorithm (HS) as their basis in an effort to generate the most optimal solution. Although useful, strategies based on the aforementioned optimization algorithms are not without limitation. Specifically, these algorithms require extensive tuning before optimal solution can be obtained. In many cases, improper tuning of specific parameters undesirably yields sub-optimal solution. Addressing this issue, this paper proposes the adoption of parameter free optimization algorithms as the basis of future t-way strategies. In doing so, this paper reviews two existing parameter free optimization algorithms involving Teaching Learning Based Optimization (TLBO) and Fruitfly Optimization Algorithm (FOA) in an effort to promote their adoption for CIT.

Keywords: optimization algorithms, teaching learning based optimization, fruitfly algorithm.

INTRODUCTION
Combinatorial optimization problem involves searching for the most optimal set of objects from a large pools of potential solution. As exhaustive search is not feasible, researchers often settle for approximate solution through the adoption of optimization algorithms (termed metaheuristics algorithms). In the effort to get the best solution (i.e. as close to the optimal solution as possible and with less computational efforts), continuous endeavors for new breed of optimization algorithms are still desirable and relevant.

Within the context of combinatorial interaction testing (CIT), many efforts are being carried out to adopt optimization algorithms as the backbone of the search strategies for generating the optimal t-way set of test cases (where t indicates the interaction strength. Complementing the upcoming field called Search based Software Engineering (SBSE), many newly strategies have been developed adopting specific optimization algorithms including Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Ant Colony (AC), Simulated Annealing (SA), and Harmony Search Algorithm (HS).

At a glance, the adoption of the aforementioned algorithms has been effective for obtaining optimal solution. A closer look reveals otherwise. Specifically, these algorithms require extensive tuning before optimal solution can be obtained. In many cases, improper tuning of specific parameters undesirably yields sub-optimal solution. Addressing this issue, this paper advocates the adoption of parameter free optimization algorithms as the basis for t-way strategies in an effort to promote their adoption for CIT.

PROBLEM DEFINITION MODEL
A configurable Fire Alarm System is used here (refer to Figure-1) to illustrate the combinatorial optimization problem involving CIT (Zamli & Alkazemi, 2015). Here, the Fire Alarm system has 4 main features: Power Supply, Initiating Device, Notification Appliance and Control Panel. Each of the features takes at most two possibilities. As for constraints (or forbidden combinations), Initiating Device must either be Digital Sensor or Analog Sensor. Additionally, Power Supply must either be Primary (AC Source) or Secondary (Battery). Finally, Fire Bell requires Keypad. Using a feature model diagram as described by Kang *et al.* (Kang, S., Hess, Nowak, & Peterson, 1990), Figure-2 captures the required parameters, values and constraints for the Fire Alarm System.

The feature model is often adopted to express different configuration of a software product line. Here, a tree structure is used to capture the relationship among different features. Such relationship must hold "true" in order to create a valid product configuration. As depicted in Figure-2, there are four types of relationship, namely, optional, compulsory, alternative and or, as well as two composition rules called requires and excludes. Furthermore, a feature may include cross-tree constraints that are explicitly expressed by the user.

Referring to the tree structure, the semantic of optional implies that the given feature is optional whilst the semantic of compulsory dictates the necessary presence of the given feature. Meanwhile, the semantic of *optional* is such that at least one or all combinations of the given features can be selected. As for the semantic of *alternative* (i.e. *XOR*), only one feature must be selected from a combination of features. Finally, *requires* dictates the need of a particular feature to co-exist with the given feature of interest and *excludes* prescribes elimination of the combination of the given features.

Going back to the Feature Model for the Fire Alarm System shown in Figure-2, Table-1 highlights an alternative view of the representation for the Fire Alarm system as the base configuration value subjected to a list of constraints that must be observed. Here, constraints can be thought of as forbidden combinations.

Exhaustive test selection for the Fire Alarm System requires 256 test cases (i.e. 1 × 2 × 2 × 2 × 2 × 2 × 2 × 2 × 2). As exhaustive testing is practically impossible in many real systems with large parameters and value, it is often desirable to focus only on specific interactions.

Here, the grand challenge is to find the most optimal subset of test cases from a large pools of potential values (based on the defined interaction) and to strictly observe the given constraints accordingly. One of the potential solutions for 2-way testing is depicted in Table 2. It can be observed all the required interactions are covered at least once and all constraints lists are observed accordingly.

When the number of parameters is small and with small constraints, the test generation process based on interaction can be done manually. However, as the parameters increase along with large constraints, manual process is impossible.

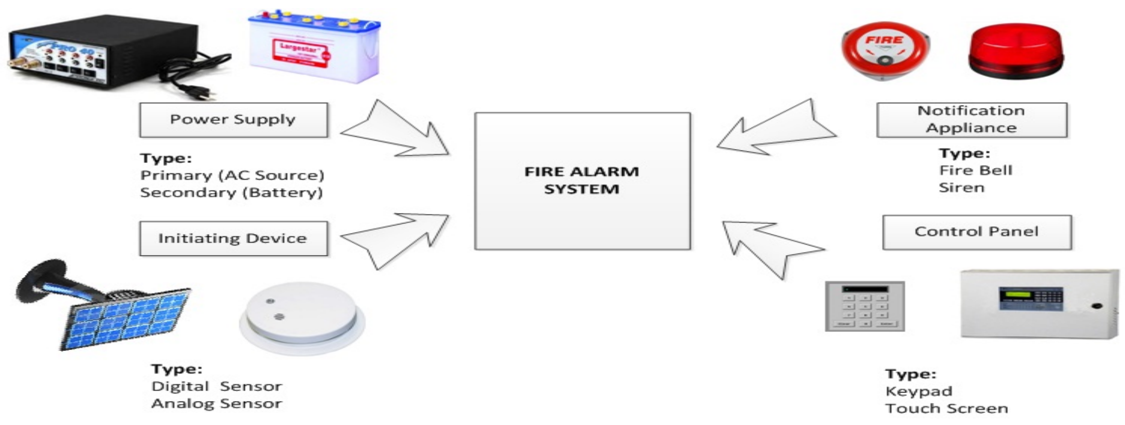

Figure-1. Fire alarm system.

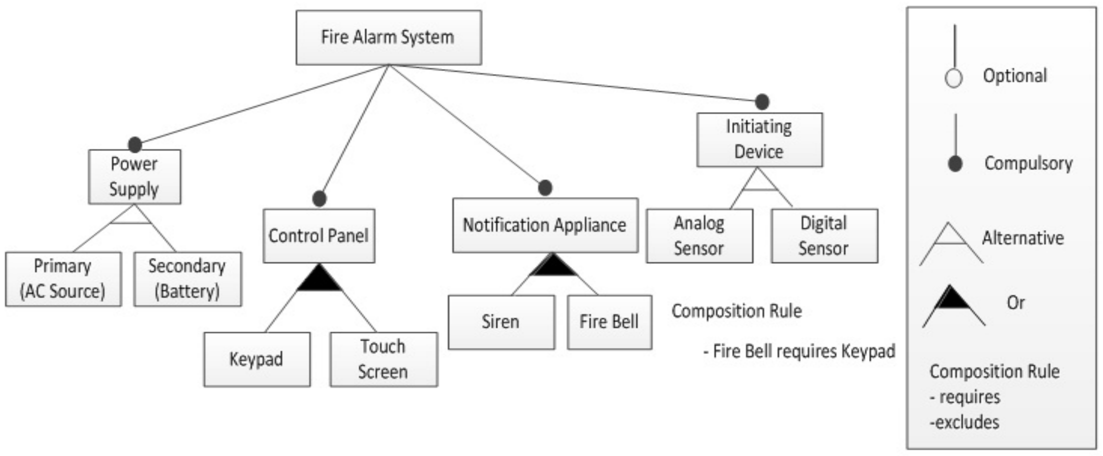

Figure-2. Fire alarm system model.

RELATED WORK

As highlighted in earlier sections, the generation of interaction test suite with optimal test size can be regarded many real systems with large parameters and value, it is often desirable to focus only on specific interactions.

Here, the grand challenge is to find the most optimal subset of test cases from a large pools of potential values (based on the defined interaction) and to strictly observe the given constraints accordingly. One of the potential solutions for 2-way testing is depicted in Table-2. It can be observed all the required interactions are covered at least once and all constraints lists are observed accordingly.

When the number of parameters is small and with small constraints, the test generation process based on

interaction can be done manually. However, as the parameters increase along with large constraints, manual process is impossible.

RELATED WORK

As highlighted in earlier sections, the generation of interaction test suite with optimal test size can be regarded as combinatorial optimization problem (Floudas *et al.* 1999). Naturally, optimization algorithm based strategies excel in this respect.

Genetic Algorithm (GA) (Afzal, Torkar, & Feldt, 2009; Bryce & Colbourn, 2007; Chen, Gu, Li, & Chen, 2009; McCaffrey, 2010; Shiba, Tsuchiya, & Kikuno, 2004; Sthamer, 1995) and Ant Colony Algorithm (ACA) (Afzal *et al.* 2009; Chen *et al.* 2009; Harman & Jones, 2001; Shiba *et al.* 2004; Wang, Xu, & Nie, 2008) represent early works in adopting optimization algorithms for t-way test generation. The GA strategy mimics the natural selection process. GA begins with randomly created test cases, which are referred to as chromosomes. These chromosomes undergo a cycle of crossover and mutation processes until the predefined fitness function is met. In each cycle, the best chromosomes are selected and added to the final test suite. Up till now, existing strategies based on GA provide no support for constraints.

Table-1. Base values and constraints for fire alarm system.

All 1-value parameter (Fire Alarm System, Power Supply, Control Panel, Notification Appliance, Initiating Device)	Primary (AC Source)	Secondary (Battery)	Keypad	Touch Screen	Siren	Fire Bell	Analog Sensor	Digital Sensor
True	True	True	True	True	True	True	True	True
False	False	False	False	False	False	False	False	False

Subjected to:

Constraints List
(x, Primary (AC Source) = true, Secondary (Battery) = true),
(x, Primary (AC Source) = false, Secondary (Battery) = false)
(x, Analog Sensor = true, Digital Sensor= true)
(x, Analog Sensor = false, Digital Sensor= false)
(x, Keypad = false, Fire Bell = true)

where 'x' represents "don't care" value

Table-2. 2-way test selection for the fire alarm system with observed constraints.

S.#	All 1-value parameter (Fire Alarm System, Power Supply, Control Panel, Notification Appliance, Initiating Device)	Primary (AC Source)	Secondary (Battery)	Keypad	Touch Screen	Siren	Fire Bell	Analog Sensor	Digital Sensor
1	True	True	False	True	True	False	False	False	True
2	True	False	True	False	False	True	False	True	False
3	True	True	False	True	False	False	True	True	False
4	True	False	True	True	True	True	True	False	True
5	True	True	False	False	False	True	False	False	True
6	True	False	True	False	True	False	False	True	False

Unlike GA, ACA (Afzal *et al.* 2009; Chen *et al.* 2009; Harman & Jones, 2001; Shiba *et al.* 2004; Wang *et al.* 2008) mimics the behaviour of colonies of ants in search for food. Because colonies of ants travel from place to place (representing the parameter) to find food (representing the value selection of each parameter), the quality of the paths taken (representing the test case) is evaluated based on the generation. The GA strategy mimics the natural selection process. GA begins with randomly created test cases, which are referred to as chromosomes. These chromosomes undergo a cycle of crossover and mutation processes until the predefined fitness function is met. In each cycle, the best chromosomes are selected and added to the final test suite. Up till now, existing strategies based on GA provide no support for constraints.

Unlike GA, ACA (Afzal *et al.* 2009; Chen *et al.* 2009; Harman & Jones, 2001; Shiba *et al.* 2004; Wang *et al.* 2008) mimics the behaviour of colonies of ants in search for food. Because colonies of ants travel from place to place (representing the parameter) to find food (representing the value selection of each parameter), the quality of the paths taken (representing the test case) is evaluated based on the amount of pheromones left behind (representing interaction coverage). The best path represents the best value of a test case to be added to the final test suite. Currently, no support is provided for constraints.

Simulated Annealing (SA) (Cohen, Colbourn, & Ling, 2008; Stardom, 2001) relies on a large random search space for generating the interaction test suite. Using probability-based transformation equations, SA adopts binary search algorithm to find the best test case per iteration to be added to the final test suite. A variant of SA has currently been developed, called CASA (Garvin, Cohen, & Dwyer, 2011), that addresses the support for constraints. CASA has been successfully adopted for software product lines testing.

PSTG (Ahmed & Zamli, 2010a, 2010b, 2011; Ahmed, Zamli, & Lim, 2012) is a strategy based on Particle Swarm Optimization which mimics the swarm behaviour of birds. Internally, PSTG iteratively performs local and global searches to find the candidate solution to be added to the final suite until all the interaction tuples are covered. No support is provided for constraints.

Complementary to PSTG, HSS (Alsewari & Zamli, 2012) is a novel strategy based on the Harmony Search Algorithm. Intuitively, HSS mimics the musician trying to compose good music from improvisation form the best tune from his memory or from random. In doing so, HSS iteratively exploits the Harmony memory to store the best found solution through a number of defined improvisations within its local and global search process. In each improvisation, one test case will be selected to the final test suite until all the required interaction tuples are covered. HSS supports the implementation of constraints.

Although much useful progress has been achieved, a subtle limitation still exists, that is, in terms of the need for extensive calibration and tuning for each algorithm parameters. Genetic Algorithm requires substantial tuning for population size, mutation and cross over rate. The improper tuning of algorithm specific parameters either increases the computational efforts or yields the local optimal solution. The same is the case of Particle Swarm Optimization algorithm which relies on population size, repetition, inertia weight, social and cognitive parameters as parameters. In similar manner, the Ant Colony algorithm requires tuning of population size, max iteration, exponent parameters, pheromone evaporation rate, and reward factor. As far as Simulated Annealing is concerned, the tuning focuses on iteration and annealing schedule. Finally, Harmony Search dictates the tuning its Harmony Memory size, max iteration as well as the two probabilistic variables Harmony Memory Considering Rate and Pitch Adjustment.

In the absence of proper tuning of the algorithms' specific parameters, the computational efforts may be wasted and the obtained solution may not be optimal. In line with such a concern, this paper reviews two existing parameter free optimization algorithms involving Teaching Learning Based Optimization (TLBO and Fruitfly Optimization Algorithm (FOA) in an effort to promote their adoption as well highlights their strengths and limitations.

ON PARAMETER FREE OPTIMIZATION ALGORITHMS

Parameter free optimization algorithms refer to the algorithms that depend only on population size and iterations for addressing the optimization problem at hand. As such, the tuning process of these algorithms may be downgraded to straightforward calibration of values (i.e. between population size and iteration). In this manner, the results obtained reflect the actual algorithm's optimal performance without the necessity of painstakingly difficult tuning process.

Owing to such attractive feature, a number of recent optimization algorithms have been developed advocating parameter free options. These algorithms include Teaching Learning based Optimization (TLBO) and Fruitfly Optimization Algorithm (FOA). The next section reviews these algorithms in details.

Teaching learning based optimization (TLBO)

In a nut shell, Teaching Learning based Optimization (TLBO) (Rao, Savsani, & Vakharia, 2011) takes an analogy from teaching and learning process between teachers and students (see Figure-3).

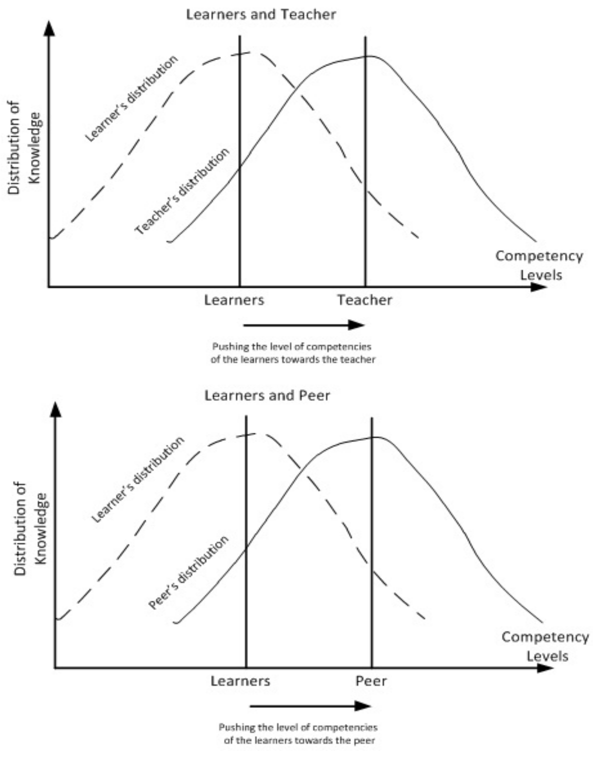
Learners and Teacher
Distribution of Knowledge
Learner's distribution
Teacher's distribution
Competency Levels
Learners
Teacher
Pushing the level of competencies of the learners towards the teacher
Learners and Peer
Distribution of Knowledge
Learner's distribution
Peer's distribution
Competency Levels
Learners
Peer
Pushing the level of competencies of the learners towards the peer

Figure-3. Improving the competencies of the learners.

Basically, teachers are trying to impart knowledge in a way that would enhance the knowledge of the students. Ideally, with knowledge gained from a particular teacher, the level of competencies of the students in a particular topic would be enhanced (i.e. resulting in improvement of the students' respective marks in that topic). As teachers also have different competency levels, there could also be potential improvements if the students learned from other teachers either on the same or different topics. At the same time, students can also learn from other students yielding similar effects (i.e. improving their level of competencies).

Specifically, TLBO divides the searching process into two main phases. The first phase, called the teacher phase, involves improving the mean differences between updated value of the potential solution (i.e. the learners) against the best solution (i.e. teacher). Here, any value of the solution is represented in a vector $X_{j,k,i}$, where j means the jth design variable (i.e. subject taken by learners), k=1,2,…m; k represents the kth population member (i.e. learner), i=1,2…..n; and i represents the ith iteration, i=1,2,….. G_{max}, where G_{max} is the maximum generations. Let $X_{k,j}$ be the best solution at any iteration i for which the value of $f(X_{k,i})$ is optimal (either minimum or maximum), the next step is to calculate the mean result $M_{j,i}$ of the learners in a particular subject j. As highlighted earlier, a teacher tries to increase the overall mean results of the class. The increase in the existing mean result of each subject by the teacher for each subject is given by:

$$\text{Difference\_Mean}_{j,k,i} = r_{j,i}(X_{j,kbest,i} - T_F M_{j,i})$$

where $X_{j,kbest,i}$ is the result of the best learner (i.e. as teacher) in the subject j, T_F is the teaching factor which decides the value of mean to be changed (capability of a teacher) and $r_{j,i}$ is the random number from [0,1]. Here, the value T_F can either be 1 or 2 decided randomly with equal probability as follows:

$$T_F = \text{round}\,[1+\text{rand}(0,1)\{2-1\}]$$

Based on the $Difference\_Mean_{j,k,i}$,the existing solution is updated in the teacher phase according to the following expression:

$$X'_{j,k,i} = X_{j,k,i} + Difference\_Mean_{j,k,i}$$

where $X'_{j,k,i}$ is the update value of $X_{j,k,i}$, $X'_{j,k,i}$ is accepted if it gives better values than $X_{j,k,i}$ and these values become input to the second phase, the learner's phase.

In the second phase, learners increase their knowledge by interaction among their peers. A learner learns iff and on if the other learners have more knowledge than he or she does. Considering a population size of n, the learning process of this phase is expressed as follows:

At any iteration i, each learner is compared with other learners randomly. Here, two learners P and Q are selected such that $X'_{P,i} \neq X'_{Q,i}$ where ($X'_{P,i}$ and $X'_{Q,i}$ are the updated values at the end of the teacher phase).

$$X''_{j,P,i} = X'_{j,P,i} + r_{j,i}(X'_{j,P,i} - X'_{j,Q,i}), \text{ if } f(X'_{P,i}) < f(X'_{Q,i})$$
$$X''_{j,P,i} = X'_{j,P,i} + r_{j,i}(X'_{j,Q,i} - X'_{j,P,i}), \text{ if } f(X'_{Q,i}) < f(X'_{P,i})$$

In this case, accept $X''_{j,P,i}$ when the objective function $f(X''_{j,P,i})$ is better than that of $f(X'_{j,P,i})$.

Fruitfly optimization algorithm

Fruit Fly Optomization algorithm (FOA) (Pan, 2012) is modelled based on the foraging behavior of the fruit fly. Fruit flies are known to have strong sense of smell and perception. As for sense of smell, fruit flies rely on its osphresis organs that can sense food sources as far as 40 km away (Pan, 2012). Using its sensitive vision, fruit flies can also find food based on the collective intelligence of individual fruit flies. Here, the value of a food source depends on its proximity, concentration, and ease of extracting. Upon finding a good lead, the population of fruit flies will flock towards the source.

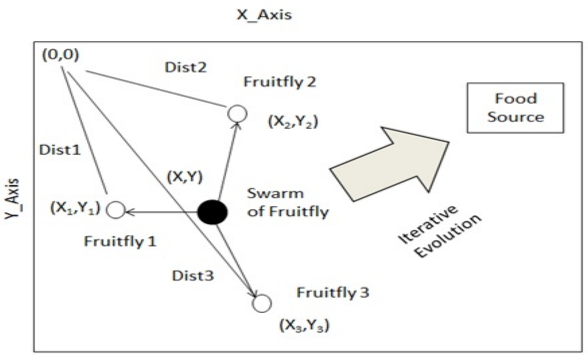

Figure-4. Fruitfly foraging behavior model.

Similar to TLBO, FOA is a population based algorithm. Here, the solutions can be represented as 2 vectors X_i and Y_i where i means the ith design variables (see Figure 4). Initially, all fruit flies search is based on specific X_axis and Y_axis at some defined origin and in random direction and distance based on osphresis sensing organs.

$X_i = X\_axis + random\_value$
$Y_i = Y\_axis + random\_value$

Since the actual food location is unknown, the distance to the origin is estimated first (*Dist*) as follows:

$Dist_i = \sqrt{(X_i^2 + Y_i^2)}$

Upon finding a particular food source, each fruit fly evaluates its potential value. Here, the smell concentration judgement value (*S*) is calculated as the reciprocal of the distance. The actual smell judgment function (or the fitness function) $Smell_i$ of the individual location of the fruitfly is modeled as the function of the concentration judgment value (*S*).

$S_i = 1/Dist_i$
$Smell_i = Function\ (S_i)$

With $Smell_i$, each fruit fly continuously compares with their peers on the best food source.

[best smell index=i] = max (Smell)

The fruit fly swarm fly toward *the best smell* and stays at the positions of the overall best food source while sending and receiving messages on other profitable location from their counter parts.

Smellbest = bestSmell
X_axis = X(bestindex)
Y_axis = Y(bestindex)

The same cycle continues for finding new food source until maximum number of generation.

Reflection on TLBO and FOA

Within any optimization algorithms, there are number of features in common including the sense of purpose, the memory, tuning issue, the diversity consideration, and the intensification process. The following paragraphs debate the usefulness of these features as far as its applicability for CIT.

The sense of purpose defines what the algorithm of interests is looking for. This consideration normally translates on how the searching process is done Analogically, the sense of purpose(s) (in term of the objective function) for CIT is to get the most optimal solution. Collectively, this so-called sense of purpose can be converted to computer algorithms. It must be noted that for some applications, there may be multiple sense of purpose (i.e. multi-objectives). In the case of TLBO, the algorithm is to interact and impart knowledge both from teacher-learners as well as learners-peers. For FOA, the sense of purpose is to search for food.

Apart from the sense of purpose(s), the memory consideration is also vital as buffers for best selection. Both TLBO and Fruitfly exploit their populations (i.e. population based) as a form of memory. It must be noted that both memory (i.e. population based) or memory-less (i.e. single solution based) are merely designed choices. There appears to be inconclusive evidence on the effectiveness of either approach.

In many optimization algorithms, tuning can be a difficult endeavor. Without proper tuning, the performance of any given optimization algorithm can be relatively poor. Unlike many existing algorithms, TLBO and FOA are two distinct algorithm that rely solely on randomization and do not rely of any specific parameter values for controlling their search process. For these reason, the obtained results from TLBO and FOA represent the actual performance of the algorithm with minimal calibration of iteration and population size.

Other than tuning, other important considerations for any optimization algorithm are on the diversification and intensification. Diversification (also termed exploration) relates to the mechanism embedded in the optimization to ensure sufficient exploration of the search space ensuring diverse solution. In the case of TLBO, the algorithm randomly selects peers for teaching and interaction and adjust the current values against some mean values of the overall population. As for fruitfly, the algorithm randomly does the random search and inform their peers of the most profitable food source. Both mechanisms appear to be helpful for ensuring diverse solution and hence, avoid false solution (i.e. local maxima minima).

In many applications, lack of diversity is often counter-productive. Researchers in software testing coin the term pesticide paradox – implying that running the same tests on the same product for some time would not yield any new bugs. Here, borrowing terms from agriculture – bugs tend to build up tolerance to pesticides. With diversity, newly created tests can be appropriately selected so as to facilitate the process of locating bugs.

Intensification (also termed exploitation) involved exploiting or pertubating few selected variables during the search process and generate variants solutions from potentially good solutions (Yang, 2010), (Yang and Karamanoglu, 2013). In TLBO, the algorithm restricts interaction exchange for better amongst peers with the same knowledge level. FOA exchanges messages and share their profitable food location. In this manner, all fruit flies can focus their search within the perimeter of the profitable location.

Both components diversification and intensification are equally important. Diversification ensures high probability to gain optimal solution in the expense of slow convergence rate. Intensification ensures quick convergence but in the expense of optimal solution. It is the matter of balancing between these two components in order to ensure higher chance of diversified solutions and within acceptable time.

CONCLUSIONS

This paper has discussed the optimization problem poses by CIT. Much work has been done in the literature as part of Search based Software Engineering (SBSE) research to address the problem adopting a number of optimization algorithms including Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Ant Colony (AC), Simulated Annealing (SA), and Harmony Search Algorithm (HS).

Although useful, the aforementioned algorithms requires extensive tuning of its parameters values in order to obtain optimal solution. Enhancing these work, this paper advocates the adoption of newly developed parameter free algorithms. In fact, as part of our future work, we are proposing two strategies for CIT, called t-FOA (t-way FOA) and t-TLBO (t-way TLBO). Our initial evaluation of both strategies have been promising as we are able to generate optimal results in many of the configurations considered.

ACKNOWLEDGEMENT

The work reported here results from many generous grants from MOSTI, Malaysia particularly involving the grant entitled: “Constraints T-Way Testing Strategy with Modified Condition Decision Coverage”. We thank MOSTI for the support.

REFERENCES

[1] Afzal W., Torkar R. and Feldt R. 2009. A systematic review of search-based testing for non-functional system properties. Information and Software Technology, Vol. 51, No. 6, pp. 957-976. doi: 10.1016/j.infsof.2008.12.005

[2] Ahmed B. S. and Zamli K. Z. 2010a. Pstg: A t-way strategy adopting particle swarm optimization. Proceedings of the 4th Asia International Conference on Mathematical /Analytical Modelling and Computer Simulation.

[3] Ahmed B. SS. and Zamli K. Z. 2010b. T-way test data generation strategy based on particle swarm optimization. The Proceedings of the 2nd International Conference on Computer Research and Development.

[4] Ahmed B. S. and Zamli K. Z. 2011. The development of a particle swarm based optimization strategy for pairwise testing. Journal of Artificial Intelligence, Vol. 4, No. 2, pp. 156-165.

[5] Ahmed B. S., Zamli K. Z. and Lim C. P. 2012. Constructing a t-way interaction test suite using the particle swarm optimization approach. International Journal of Innovative Computing, Information and Control, Vol. 8, No. 1, pp. 431-452.

[6] 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, pp. 553-568.

[7] Bryce R. and Colbourn C. 2007. One-test-at-a-time heuristic search for interaction test suites. Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation, London, England.

[8] Chen X., Gu Q., Li A. and Chen D. 2009. Variable strength interaction testing with an ant colony system approach. Proceedings of the 16th Asia-Pacific Software Engineering Conference.

[9] Cohen M. B., Colbourn C. J. and Ling A. C. H. 2008. Constructing strength three covering arrays with augmented annealing. Discrete Mathematics, Vol.. 308, No. 13, pp. 2709-2722.

[10] Floudas C. A., Pardalos P. M., Adjiman C. S., Esposito, W. R. Gümüs Z. H., Harding S. T., Schweiger C. A. 1999. Handbook of test problems in local and global optimization, Vol. 33, Kluwer Academic Publishers Dordrecht.

[11] Garvin B. J., Cohen M. B. and Dwyer M. B. 2011. Evaluating improvements to a meta-heuristic search for constrained interaction testing. Empirical Software Engineering, No. 16, pp. 61-102.

[12] Harman M. and Jones B. F. 2001. Search-based software engineering. Information and Software Technology, Vol. 43, No. 14, pp. 833-839.

[13] Kang K. S. C., Hess J., Nowak W. and Peterson S. 1990. feature-oriented domain analysis (foda) feasibility study: Software Engineering Institute, Carnegie Mellon University.

[14] McCaffrey J. 2010. An empirical study of pairwise test set generation using a genetic algorithm. Proceedings of the 7th International Conference on Information Technology.

[15] Pan W.-T. 2012. A new fruit fly optimization algorithm: Taking the financial distress model as an example. Knowledge Based Systems, Vol. 26, pp. 69-74.

[16] Rao R. V., Savsani V. J. and Vakharia D. P. 2011. Teaching-learning-based optimization: A novel method for constrained mechanical design optimization problems. Computer Aided Design, Vol. 43, pp. 303-313.

[17] Shiba T., Tsuchiya T. and Kikuno T. 2004. Using artificial life techniques to generate test cases for combinatorial testing. Proceedings of the 28th Annual

International Computer Software and Applications Conference.

[18] Stardom J. 2001. Metaheuristics and the search for covering and packing array (Master of Science thesis), Simon Fraser University, Canada.

[19] Sthamer H. 1995. The automatic generation of software test data using genetic algorithms. (PhD thesis), Universityof Glamorgan, Pontyprid, Wales.

[20] Wang Z. Y., Xu B. W. and Nie C. H. 2008. Greedy heuristic algorithms to generate variable strength combinatorial test suite. Proceedings of the 8th International Conference on Quality Software.

[21] Zamli K. Z. and Alkazemi B. Y. 2015. Combinatorial testing: UMP Publisher.