JOURNAL OF INFORMATION AND COMMUNICATION TECHNOLOGY
http://jict.uum.edu.my

How to cite this article:
Zamli, K. Z., Din, F., Nasser, A., Ramli, N., & Mohamed, N. (2021). Dynamic probability selection for flower pollination algorithm based on metropolis-hastings like criteria. *Journal of Information and Communication Technology, 20*(1), 41-56.

DYNAMIC PROBABILITY SELECTION FOR FLOWER POLLINATION ALGORITHM BASED ON METROPOLIS-HASTINGS CRITERIA

[1]Kamal Zuhairi Zamli, [1,2]Fakhrud Din, [1]Abdullah Nasser, [3]Nazirah Ramli & [3]Noraini Mohamed
[1]Faculty of Computer Systems & Software Engineering, Universiti Malaysia Pahang, Malaysia
[2]Department of Computer Science & IT, University of Malakand, Pakistan
[3]Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA Pahang, Malaysia

Corresponding author: kamalz@ump.edu.my
fakhruddin@uom.edu.pk, abdullahnasser83@gmail.com.my
nazirahr, noraini_mohamed@uitm.edu.my

Received: 30/9/2019 Revised: 6/11/2020 Accepted: 4/6/2020 Published: 4/11/2020

ABSTRACT

Flower Pollination Algorithm (FPA) is a relatively new meta-heuristic algorithm that adopts its metaphor from the proliferation role of flowers in plants. Having only one parameter control (i.e. the switch probability, p_a) to choose from the global search (i.e. exploration) and local search (i.e. exploitation) is the main strength of FPA as compared to other meta-heuristic algorithms. However, FPA still suffers from variability of its performance as there is no one size that fits all values for p_a, depending on the characteristics of the optimisation function. This paper proposed flower pollination algorithm

metropolis-hastings (FPA-MH) based on the adoption of Metropolis-Hastings criteria adopted from the Simulated Annealing (SA) algorithm to enable dynamic selection of the p_a probability. Adopting the problem of t-way test suite generation as the case study and with the comparative evaluation with the original FPA, FPA-MH gave promising results owing to its dynamic and adaptive selection of search operators based on the need of the current search.

Keywords: Dynamic probability selection, flower pollination algorithm, optimisation, t-way testing, data mining.

INTRODUCTION

Nowadays, the need to deal with optimisation problems is commonplace mainly to ensure effective return on investments (Odili, 2018). In a nutshell, optimisation problems relate to the problem of finding the best solution from all the available feasible solutions. In some scenarios, the focus is to maximise profits, whilst in other scenarios, the stress is on minimising costs (Basir, Yusof, & Hussin, 2020; Clarke et al., 2003; Harman, 2007; Rao & Pawar, 2020; Swesi & Bakar, 2020). Meta-heuristic-based algorithms have been known to be effective to deal with optimisation problems (Hairuddin, Yusuf, & Othman, 2020; Rao, 2019; Shehab, Khader, & Laouchedi, 2020). Many successful meta-heuristic algorithms have been designed to-date utilising and mimicking many natural phenomena from genetic evolution, swarming of birds, teaching learning scenario as well as pollination of flowers, to name a few (Zou, Chen, & Xu, 2019). Flower pollination algorithm (FPA) is one of the efficient contemporary meta-heuristic algorithms. The metaphor taken by FPA comes from flowers proliferation role in plants (Yang, 2010). Having only one parameter control (i.e. probability, p_a) to choose from the global search or exploration and local search or exploitation is the main strength of FPA as compared to other meta-heuristic algorithms.

On one hand, the fact that FPA only has one parameter control (i.e. probability selection operators, p_a) facilitate its tuning process (unlike other meta-heuristics that adopt more than one parameter control such as genetic algorithm (GA), ant colony optimisation (ACO), and particle swarm optimisation (PSO). On the other hand, FPA still suffers from variability of its performance as there is no one size that fits all values for p_a, depending on the features of the optimisation function.

It is known that search spaces vary from problem to problem, i.e. dynamic; therefore, any fixed and pre-set p_a can be counterproductive for considering a sufficient amount of exploration and exploitation. For most of the optimisation

problems, more exploration is preferred at the beginning of the search process. Exploitation is performed when the search process is about to end. To achieve such behaviour, FPA needs to replace its static p_a probability with a dynamic one. This paper proposes Flower Pollination Algorithm Metropolis-Hastings based on the adoption of Metropolis-Hastings criteria from the Simulated Annealing algorithm to enable dynamic selection of the p_a probability. Adopting the benchmark t-way test suite construction problem as the case study and with the relative evaluation of the results obtained with the original FPA, FPA-MH gives promising performances owing to its dynamic and adaptive mechanisms for selecting appropriate search operators based on current search requirements.

OVERVIEW OF FLOWER POLLINATION ALGORITHM

FPA optimises solutions by utilising the flowering plants' pollination process where pollens are transferred by pollinators for reproduction. In the pollination process, pollen grains produced by the flower's male component travel to the flower's female component (i.e. ovules borne) through insects, birds, animals or other sources such as wind or water. The former is termed as biotic pollination, whereas the latter is termed as abiotic pollination. Transmission of pollens from male to female parts within the same flower or within flowers of the same plant is known as self-pollination. On the other hand, cross-pollination is the process where pollens are transferred within flowers of different plants (see Figure 1). These various pollination strategies are embedded in FPA for solving optimisation problems (Azad, Bozorg-Haddad, & Chu, 2018). FPA consists of two key operations, namely global pollination and local pollination. These operations are inspired by the pollination processes mentioned above. Global pollination operation simulates the transfer of pollens within flowers over long distances, where insects or other animals act as pollinators. Equation 1 describes the global pollination operation mathematically.

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

where $x_i^{(t)}$ is the i^{th} pollen or a solution at a particular run t, $gbest$ is best solution in the population at the current run, γ is the step size and is greater than 0. $L\acute{e}vy(\lambda)$ represents lévy flight. This efficiently mimics the behaviour of long-distance movement of pollinators such as insects, and denotes a random walk interspersed by long steps distributed on different parts of the search landscape based on a power law.

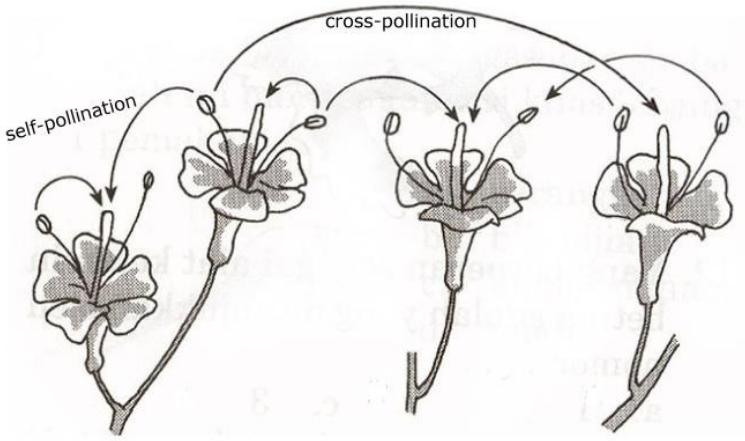

Figure 1. *Pollination process within flowering plants.*

The local pollination step mimics the short distance transfer of pollens (i.e. the transfer of flower pollens from one flower to itself or to close-by neighbours). Equation 2 gives the formulation for local pollination.

$$x_i^{(t+1)} = x_i^{(t)} + \varepsilon\left(x_j^{(t)} - x_k^{(t)}\right) \tag{2}$$

where $x_j^{(t)}$ and $x_k^{(t)}$ represent two solutions chosen randomly such that $i \neq j$ from separate flowers, ε is a random number from interval [0, 1].

Generally, FPA starts by randomly initialising the population of flower pollens (i.e. solutions). During each generation, FPA generates a new solution by employing either global pollination or local pollination, which is selected by the value assigned to the switch probability p_a from the interval [0, 1]. Algorithm 1 summarises the steps of basic FPA.

Algorithm 1: Flower Pollination Algorithm (Yang, 2010)	
INPUT	Population of solutions
OUTPUT	Updated population and best solution gbest
while $t <$ *Maximum no. of generations* **or** stop criteria not met **do**	
for $i = 1$: n (every pollen in the population)	
if (random no. $r < p_a$)	

(continued)

Algorithm 1: Flower Pollination Algorithm (Yang, 2010)

L (a step vector) is generated based on the problem dimension that follows a lévy distribution
Global pollination operation: $x_i^{(t+1)} = x_i^{(t)} + \gamma Lévy(\lambda)(X_t - gbest)$
else
Generate ε value randomly that falls in [0,1] following uniform distribution
Randomly select *j* and *k* such that both are not same
Local pollination operation: $x_i^{(t+1)} = x_i^{(t)} + \varepsilon(x_j^{(t)} - x_k^{(t)})$
end if
Evaluate fitness for all new solutions updated in the current run
If fitness for the new updated solutions are better, replace the old ones in the population with them
end for
Obtain the current best pollen *gbest*
end while

RELATED WORKS

After its entry in the optimisation literature, many variants of FPA have been introduced. Several research endeavours consider the hybridisation of FPA with existing meta-heuristic algorithms. In this connection, the work by Abdel-Raouf, El-Henawy, and Abdel-Baset (2014) presented the FPA and Harmony Search (HS) algorithm hybridisation for solving Sudoku puzzles. In the work by Abdel-Baset and Hezam (2015), they proposed a hybrid approach based on FPA and GA to address problems with constraints. Similar efforts (hybridisation with different meta-heuristic algorithms) can be seen in other instances of optimisation problems such as satellite image classification (Johal, Singh, & Kundra, 2010), wind-thermal dynamic multi-objective dispatch problems (Dubey, Pandit, & Panigrahi, 2015), and route planning for unmanned undersea vehicles (Zhou & Wang, 2016), to name a few, which have been successfully addressed by FPA. Recently, Wang and Zhou (2014) utilised local neighbourhood search and dimension-wise evaluation to improve the convergence speed of FPA. In 2016, a study (Zhou, Wang, & Luo, 2016) employed the elite opposition approach to find the best possible result. Zhou et al. (2016) modified FPA by introducing three new operators: elite-based mutation, discard pollen, and crossover. Rao et al. (2018) introduced a two-stage initialisation process, namely creation of a sub population vector and application of evolutionary operators, to improve the performance of basic FPA for detecting more optimal solutions. In a more recent work (Kopciewicz & Łukasik, 2019), the researchers proposed Biotic FPA (BFPA), which is based on the natural phenomenon called flower constancy, i.e. insects' ability to remember locations of quality pollen sources. Although useful, most of

the improvements of FPA involve hybridisation with other meta-heuristic algorithms. To the very least, some variants also incorporate search operators from different meta-heuristic algorithm into FPA. The effect of adding new search operators can be twofold. Firstly, there are potentially new additions of control parameters that may introduce difficulty of tuning. Secondly, the addition of new search operators tends to alter the original structure of FPA, thus, demoting its learnability. In this research study, a minimalist approach is adopted so as to maintain the original structure of FPA.

As far as the problem of test suite construction is concerned, the literature includes many works that adopted different meta-heuristic algorithms such as simulated annealing (SA), PSO, GA, artificial bee colony (ABC), etc. SA (Cohen, Gibbons, Mugridge, & Colbourn, 2003) was one of the first meta-heuristic algorithms that constructed optimal test suites. The algorithm outperforms greedy-based strategies; whereas for higher strength arrays, it exhibits acceptable performance against computational strategies. PSO (Ahmed & Zamli, 2011) offers better results for uniform as well as variable interaction strength test suites. For many of the adopted cases, PSO results have been found effective against a range of strategies based on meta-heuristic and greedy algorithms. Recently, GA (Esfandyari & Rafe, 2018) has been adopted for the problem of test suite construction. The strategy based on GA offers the highest strength test suites to date. Artificial bee colony strategy (ABCS) (Alazzawi, Rais, & Basri, 2018) is based on the ABC algorithm that can generate test suites. The strategy produces good results as compared to other strategies based on artificial intelligence (AI) and computational methods. Hyper-heuristic strategies (Din & Zamli, 2018) based on Monte Carlo hyper-heuristic can generate good quality pairwise test suites. This strategy employs more than one low-level heuristic that generate test suites under the control of a high-level heuristic. Though very effective, strategies proposed using these algorithms require extensive parameter tuning. On the contrary, FPA has only one parameter that requires tuning as compared to tuning of more parameters in case of the mentioned algorithms.

FLOWER POLLINATION ALGORITHM WITH METROPOLIS-HASTINGS CRITERIA

The Metropolis-Hastings criteria are well-known within the SA algorithm's probability density function. This probability density function decreases in nature, allowing SA to perform exploration (i.e. high probability of accepting poor solution) early in the iteration and converge to specific value(s) towards the end of the search process (i.e. with low probability of accepting poor solution). The Metropolis-Hastings criteria work well with SA as the search operator is purely based on the neighbourhood search (i.e. new solution is

merely a perturbation of the old solution), allowing an elegant way of getting out of local optima. In a nutshell, the Metropolis-Hastings criteria exploit (minus) exponential difference between (the fitness function values) of the best solution and the current solution as well as the current iteration value to permit an exponentially decreasing function.

Revisiting Algorithm 1, the present study opted to make the static p_a probability dynamic using a similar probability density function as Metropolis-Hastings criteria. Unlike the Metropolis-Hastings criteria where the probability density function was utilised for selecting the candidate solution for next iteration, the present study applied the dynamic probability density function for selecting the actual search operators for use in the next iteration. Additionally, unlike the Metropolis-Hastings criteria, the current work did not exploit the differences (between the fitness function values) of the best solution and the current solution. To be specific, the present study proposed the probability density function as defined in Equation 3 as follows:

$$p_a = e^{-\left(1+get\_weight(gbest)*\frac{iteration}{MaxGeneration}\right)} \quad (3)$$

where *gbest* is the value of the last best solution, iteration is the current i^{th} iteration, *MaxGeneration* is the total number of improvement generations.

The net effect of this dynamic p_a was that during the early part in the quest for searching optimal solutions, there were more chances to adopt global search via the global pollination operator (as the probability density function gave a large value in an exponentially decreasing manner). In this manner, the search process correctly explored all the search space for the best candidate solution. Towards the end, the probability density function gave a (decreasingly) low value, thus, favouring the local search via the local pollination operator. Here, the movement of pollen was restricted to small steps, allowing the search to go around the current best solution.

CASE STUDY: THE T-WAY TEST GENERATION PROBLEM

For the illustration of the t-way test generation problem, a hypothetical web-based application was considered. This application encompasses four parameters, each containing four values as shown in Table 1. Testing all possible combinations of these parameters is desirable in an ideal situation. In this example, there were exhaustively $3^4 = 81$ combinations. However, all exhaustive combinations could be enormously large in real-life testing, thus, performing exhaustive testing was prohibitively impossible considering the

limited resources and time constraints. Therefore, the researchers formulated a sampling method that used t-way interaction between parameter values (here *t* in t-way is the interaction strength).

The choice of t-way interaction strength depended on the test requirements (Alazzawi, Rais, & Basri, 2019). Typically, based on the empirical results within the literature, *t* might be selected from $t = 2$ until $t = 6$ depending on the parameters of choice. However, for the current problem, the maximum *t* was just four (as the exhaustive tests). To show how t-way test suite was generated, let $t = 2$ (i.e. pairwise interaction).

Table 1

Hypothetical Web-based Application

P1	P2	P3	P4
Firefox	Linux	ISDN	Screen
IE	Macintosh	PPP	Networked
Netscape	Windows	LAN	Local

Here, the set of interactions (*P1*, *P2*), (*P1*, *P3*), (*P1*, *P4*), (*P2*, *P3*), (*P2*, *P4*), and (*P3*, *P4*) denoted the pairwise interactions among the four parameters of the web-based application. For the running example, there would be 54 total pairs of interactions for the nine 2-way interactions among six parameters. These pairs of interactions are also known as interaction tuples. It could be easily deduced that all the 2-way interactions could be covered by a total of 10 test cases as given in Table 2.

Table 2

Suggested 2-way Test Set

Test ID	P1	P2	P3	P4
1	Netscape	Windows	LAN	Local
2	IE	Windows	PPP	Networked
3	Firefox	Windows	ISDN	Screen
4	Netscape	Macintosh	PPP	Screen
5	IE	Macintosh	LAN	Local
6	Firefox	Macintosh	LAN	Networked
7	Netscape	Linux	ISDN	Networked
8	IE	Linux	LAN	Screen
9	Firefox	Linux	PPP	Local
10	IE	Macintosh	ISDN	Local

To be specific, the t-way test suite generation involved the following fitness function in Equation 4 (Ali, Othman, Yacob, & Alkanaani, 2019; Huang et al., 2020):

$$\text{Maximise } f(x) = \sum_1^N x_i$$
$$\text{Subject to } x \in x_i \ , i = 1, 2, \ldots, N \tag{4}$$

where $f(x)$ represents the fitness function that computes the weight of the individual test case in terms of the maximum number of coverages of the interaction tuples, x represents the set of each decision variable x_i; x_i, which consists of a series of values, that is, $xi = \{x_i(1), x_i(2),\ldots, x_i(K)\}$, such that $(x_i(1) < x_i(2) <\ldots < x_i(K))$, N represents total parameters of the system, K represents the total number of values that each discrete variable holds.

Test suites with t-way interactions could be described mathematically with covering arrays notation represented as CA (N;t,v^k) (Al-Sammarraie & Jawawi, 2020; Esfandyari & Rafe, 2020; Younis, 2020). For example, a test suite with nine test cases (i.e. size nine) that covered all pairwise or 2-way interaction tuples of a system consisting of four 3-value parameters could be represented as CA (9; 2, 3^4). In this paper, the same covering array notation would be used.

T-WAY TEST SUITE GENERATION BASED ON THE PROPOSED STRATEGY

In this section, the design of FPA-MH-based strategy is described for addressing the construction problem of t-way test suites. Algorithm 2 presents the complete FPA-MH strategy. The dotted line box shows the proposed modification in basic FPA.

Referring to Algorithm 2 and based on the receiving configuration (i.e. interaction strength t, number of parameter and their values), FPA-MH begins generating all possible combinations of parameters-values with size t. Then, FPA-MH starts generating population of pollen randomly in the Candidate Solution list, CS. Here, each flower pollen represents one test case and fitness function is the number of interactions covered by the test case. Then, CS is subjected to updating process using FPA-MH operations (local and global pollinations). Unlike standard FPA, the switch between local and global pollinations is dynamically based on current best solution and iteration. The process updating CS continues until the stopping criteria is met, and the best test case that covers the maximum number of interactions, among CS, is

selected and added into final test suite. This process is repeated until all the interactions are covered.

Algorithm 2: Strategy based on FPA-MH for solving the problem of t-way test suite generation

INPUT :(*P*, *v*, *t*): Set of parameters *P*, values *v* of *P* and *t* as interaction strength
OUTPUT :*TS* final test suite
Based on *P*, *v*, *t,* populate interaction element list (*IEL*) by generating all possible interaction elements
Let *TS* be the final test suite;
Generate initial population of pollen randomly in the Candidate Solution list, *CS*
//Perform search using FPA
while *IEL* size is not zero (i.e. not empty) **do**
while *iteration* < *MaxGeneration* or stop criteria not met **do**
gbest = get_best_pollen();
for *i* = 1: n (all *n* solutions in *CS*)
if (rand < $p_a = e^{-\left(1+get\_weight(gbest)*\frac{iteration}{MaxGeneration}\right)}$)
Generate a step vector L based on the problem dimension that follows a lévy distribution
Global pollination operation: $x_i^{(t+1)} = x_i^{(t)} + \gamma Lévy(\lambda)(X_t - gbest)$
else
Generate e using a uniform distribution from the interval [0,1]
Randomly select *j* and *k* such that both are not same
Local pollination operation: $x_i^{(t+1)} = x_i^{(t)} + \varepsilon(x_j^{(t)} - x_k^{(t)})$
end if
Evaluate fitness of the newly generated solutions
If newly generated solutions are better, replace the old ones in the population with them
end for
Obtain the current best pollen *gbest*
end while
Add the best test case denoted by the best pollen *gbest* into the test suite *TS*.
Remove the interaction tuples covered by *gbest* from *IEL*.
end while
Display and save *TS*

RESULTS AND DISCUSSION

The present study's main goal in terms of evaluation focuses on ascertaining whether or not the introduction of dynamic p_a improves the performance of the original FPA. For a fair comparison (i.e. both FPA and FPA-MH are non-deterministic strategies), both of them were run 20 times for every configuration and the best obtained test suite sizes and execution time were reported. For comparison purpose, FPA parameters such as switch probability,

pollen population size, and maximum number of improvements were set at 0.8, 50, and 600, respectively, as suggested in a related study (Nasser, Zamli, Alsewari, & Ahmed, 2018). Based on the same FPA parameter settings, the researchers ensured that the number of fitness function evaluation for both testing strategies was also the same as only one operator (i.e. either local or global pollination) was selected per iteration. As far as ensuring fair time comparison, the study also adopted the same data structure and language implementation for FPA and FPA-MH running on the same laptop with Core i7, 3.60 GHz CPU, 16 GB RAM and Windows 10.

Table 3

Performance Comparison of FPA-MH with original FPA, PSO, and CS based on Generated Test Suite Sizes

	FPA-MH	FPA	PSO	CS
	Best Size/Time(sec)	Best Size/ Time(sec)	Best Size	Best Size
CA (N; 2, 3^7)	15/1.36	15/1.53	15	**14**
CA (N; 3, 3^7)	**48**/9.33	49/9.32	50	**48**
CA (N; 4, 3^7)	**150**/35.82	**150**/36.70	155	154
CA (N; 5, 3^7)	**432**/86.42	**432**/87.18	441	434
CA (N; 6, 3^7)	**886**/104.61	920/105.10	977	963
CA (N; 3, 3^4)	**27**/0.93	**27**/0.97	30	28
CA (N; 3, 3^5)	**38**/2.68	39/2.76	39	**38**
CA (N; 3, 3^6)	**33**/4.75	**33**/4.70	45	43
CA (N; 3, 3^8)	**51**/24.25	**51**/24.31	54	53
CA (N; 3, 3^9)	**56**/41.67	**56**/42.21	58	58
CA (N; 3, 3^{10})	**59**/67.71	**59**/67.56	62	62
CA (N; 3, 2^7)	15/4.09	**15**/3.78	13	**12**
CA (N; 3, 4^7)	**112**/31.93	**112**/32.75	116	117
CA (N; 3, 5^7)	**216**/63.90	217/64.08	225	223
CA (N; 3, 6^7)	370/121.06	**368**/121.27	-	-

Entries with bold font indicate best test suite sizes. Entries with '-' indicate result not available

Referring to Table 3, FPA-MH gave very good overall performance. To be specific, FPA-MH obtained the most optimal size in many cases (i.e. 12 out of 15 cases). Comparing with the standard FPA, FPA-MH outperformed FPA in five cases, while FPA outperformed FPA-MH in only one case. The FPA-MH

strategy appeared to produce better results owing to its ability for exploring all the search space for the best candidate solution and subsequently decreasing the exploration in exchange for increased exploitation.

As far as the comparison in terms of test suite sizes of FPA-MH with other strategies based on PSO and CS algorithms is concerned, FPA-MH mostly outperformed both in terms of the generated best test suite sizes. FPA-MH outperformed the PSO-based test suite generation strategy in 12 out of the total 14 instances. PSO matched FPA-MH's result for CA (N; 2, 3^7), whereas it obtained a better result than it did for CA (N; 3, 2^7). Similarly, FPA-MH outperformed the CS-based strategy in 10 out of the total 14 cases. CS obtained better t-way test suite sizes for two cases (i.e. CA (N; 2, 3^7) and CA (N; 3, 2^7)), whereas it twice matched the best test sizes of FPA-MH.

Concerning execution time, it was observed that both FPA-MH and FPA took the similar execution time and the new improvement of FPA-MH did not take any extra overhead. These results suggested that the modification was robust and easy to embed with the basic FPA.

Referring to the boxplot analysis in Figure 2, FPA-MH outperformed the basic FPA by producing the best mean results in case of both CA (N, 2 3^7) and CA (N; 2, 3^4). Similarly, the interquartile ranges of FPA-MH were better than that of FPA, showing its consistency in obtaining optimal results. This analysis justified the effectiveness of the proposed FAP-MH over FPA.

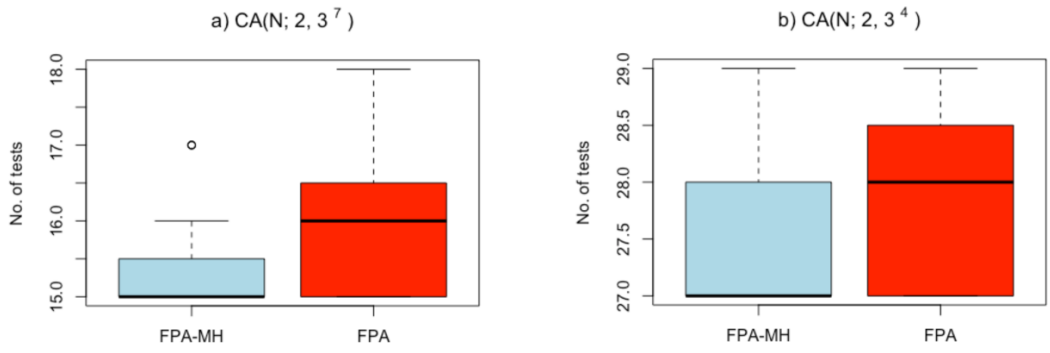

Figure 2. *Boxplot analysis for FPA-MH Vs FPA.*

CONCLUSION

This paper proposed a new strategy based on FPA and FPA-MH for solving the t-way test suite generation problem. FPA-MH adopted the Metropolis-

Hastings criteria from the Simulated Annealing algorithm to enable dynamic selection of global and local pollinations. The experimental results showed that FPA-MH has very good overall performance as compared to the standard FPA, owning to the dynamic and adaptive selection of search operators based on the need of the current search.

As part of future research work, the application of FPA-MH will be examined for other optimisation problems such as scheduling, clustering, and constrained optimisation.

ACKNOWLEDGEMENT

The FRGS Grant for the project under the title "A Reinforcement Learning Sine Cosine based Strategy for Combinatorial Test Suite Generation (Grant no.: RDU170103)" has funded the work presented in this paper. The authors are thankful to the Ministry of Higher Education Malaysia (MOHE) for the grant.

REFERENCES

Abdel-Baset, M., & Hezam, I. M. (2015). An effective hybrid flower pollination and genetic algorithm for constrained optimization problems. *Advanced Engineering Technology and Application An International Journal*, *4*, 27–27. https://doi.org/10.12785/aeta/040203

Abdel-Raouf, O., El-Henawy, I., & Abdel-Baset, M. (2014). A novel hybrid flower pollination algorithm with chaotic harmony search for solving sudoku puzzles. *International Journal of Modern Education and Computer Science*, *6*(3), 38. https://doi.org/10.5815/ijmecs.2014.03.05

Ahmed, B. S., Abdulsamad, T. S., & Potrus, M. Y. (2015). Achievement of minimized combinatorial test suite for configuration-aware software functional testing using the cuckoo search algorithm. *Information and Software Technology*, *66*, 13–29. https://doi.org/10.1016/j.infsof.2015.05.005

Ahmed, B. S., & Zamli, K. Z. (2011). A variable-strength interaction test suites generation strategy using particle swarm optimization. *Journal of Systems and Software*, *84*(12), 2171–2185. https://doi.org/10.1016/j.jss.2011.06.004

Al-Sammarraie, H. N. N., & Jawawi, D. N. A. (2020). Multiple black hole inspired meta-heuristic searching optimization for combinatorial testing. *IEEE Access*, *8*, 33406–33418. https://doi.org/10.1109/ACCESS.2020.2973696.

Alazzawi, A. K., Rais, H. M., & Basri, S. (2018). Artificial bee colony algorithm for t-way test suite generation. Paper presented at *the 4th International Conference on Computer and Information Sciences*. https://doi.org/10.1109/ICCOINS.2018.8510601

Alazzawi, A. K., Rais, H. M., & Basri, S. (2019). Parameters tuning of hybrid artificial bee colony search based strategy for t-way testing. *International Journal of Innovative Technology and Exploring Engineering*, *8*(5S). https://doi.org/10.1109/ICCOINS.2018.8510601

Ali, A. S. M., Othman, R. R., Yacob, Y. M., & Alkanaani, J. M. (2019). Parameters tuning of adaptive firefly algorithm based strategy for t-way testing. *International Journal of Innovative Technology and Exploring Engineering*, *19*(1). https://doi.org/10.35940/ijitee.A6111.119119

Azad, M., Bozorg-Haddad, O., & Chu, X. (2018). Flower pollination algorithm (FPA). In *Advanced optimization by nature-inspired algorithms* (pp. 59–67) Singapore: Springer.

Basir, M. A., Yusof, Y., & Hussin, M. S. (2020). Optimization of attribute selection model using bio-inspired algorithms. *Journal of Information and Communication Technology*, *18*(1), 35–55. https://doi.org/10.32890/jict2019.18.1.8280.

Clarke, J., Dolado, J. J., Harman, M., Hierons, R., Jones, B., Lumkin, M., . . . Roper, M. (2003). Reformulating software engineering as a search problem. *IEEE Proceedings-software*, *150*(3), 161–175. https://doi.org/10.1049/ip-sen:20030559

Cohen, M. B., Gibbons, P. B., Mugridge, W. B., & Colbourn, C. J. (2003). *Constructing test suites for interaction testing.* Paper presented at *the 25th International Conference on Software Engineering, Portland, USA*. https://doi.org/10.1109/ICSE.2003.1201186

Din, F., & Zamli, K. Z. (2018). Hyper-heuristic based strategy for pairwise test case generation. *Advanced Science Letters*, *24*(10), 7333–7338. https://doi.org/10.1166/asl.2018.12938

Dubey, H. M., Pandit, M., & Panigrahi, B. (2015). Hybrid flower pollination algorithm with time-varying fuzzy selection mechanism for wind integrated multi-objective dynamic economic dispatch. *Renewable Energy*, *83*, 188–202. https://doi.org/10.1016/j.renene.2015.04.034

Esfandyari, S., & Rafe, V. (2018). A tuned version of genetic algorithm for efficient test suite generation in interactive t-way testing strategy. *Information and Software Technology*, *94*, 165–185. https://doi.org/10.1016/j.infsof.2017.10.007

Esfandyari, S., & Rafe, V. (2020). Extracting combinatorial test parameters and their values using model checking and evolutionary algorithms. *Applied Soft Computing*, *91*, 106219. https://doi.org/10.1016/j.asoc.2020.106219

Hairuddin, N. L., Yusuf, L. M., & Othman, M. S. (2020). Gender classification on skeletal remains: Efficiency of metaheuristic algorithm method and optimized back propagation neural network. *Journal of Information and Communication Technology*, *19*(2), 251–277. https://doi.org/10.32890/jict2020.19.2.4950.

Harman, M. (2007). *The current state and future of search based software engineering.* Paper presented at *the Future of Software Engineering Conference (FOSE)*. https://doi.org/10.1109/FOSE.2007.29

Huang, R., Chen, H., Zhou, Y., Yueh Chen, T., Towey, D., Fai Lau, M., . . . Chen, J. (2020). Covering array constructors: An experimental analysis of their interaction coverage and fault detection. *The Computer Journal*. https://doi.org/10.1093/comjnl/bxaa020

Johal, N. K., Singh, S., & Kundra, H. (2010). A hybrid fpab/bbo algorithm for satellite image classification. *International Journal of Computer Applications (0975–8887)*, *6*(5). https://doi.org/10.5120/1074-1403

Kopciewicz, P., & Łukasik, S. (2019). Exploiting flower constancy in flower pollination algorithm: Improved biotic flower pollination algorithm and its experimental evaluation. *Neural Computing and Applications*, 1–12. https://doi.org/10.1007/s00521-019-04179-9

Nasser, A. B., Zamli, K. Z., Alsewari, A. A., & Ahmed, B. S. (2018). Hybrid flower pollination algorithm strategies for t-way test suite generation. *PloS one*, *13*(5), e0195187–e0195187. https://doi.org/10.1371/journal.pone.0195187

Odili, J. B. (2018). Implementation analysis of cuckoo search for the benchmark rosenbrock and levy test functions. *Journal of Information and Communication Technology*, *17*(1), 17–32. https://doi.org/10.32890/jict2018.17.1.8243

Rao, R. V. (2019). *Jaya: An advanced optimization algorithm and its engineering applications*. Cham: Springer International Publishing.

Rao, R. V., & Pawar, R. B. (2020). *Optimal weight design of a spur gear train using rao algorithms*. Cham: Springer. https://doi.org/10.1007/978-3-030-44758-8_33

Shehab, M., Khader, A. T., & Laouchedi, M. (2020). A hybrid method based on cuckoo search algorithm for global optimization problems. *Journal of Information and Communication Technology*, *17*(3), 469–491. https://doi.org/10.32890/jict2018.17.3.8261

Swesi, I. M. A. O., & Bakar, A. A. (2020). Feature clustering for PSO-based feature construction on high-dimensional data. *Journal of Information and Communication Technology*, *18*(4), 439–472. https://doi.org/10.32890/jict2019.18.4.8297

Wang, R., & Zhou, Y. (2014). Flower pollination algorithm with dimension by dimension improvement. *Mathematical Problems in Engineering*, *2014*. https://doi.org/10.1155/2014/481791

Yang, X.-S. (2010). *Nature-inspired metaheuristic algorithms*. United Kingdom: Luniver Press.

Younis, M. (2020). Deo: A dynamic event order strategy for t-way sequence covering array test data generation. *Baghdad Science Journal*, *17*(2), 0575. https://doi.org/10.21123/bsj.2020.17.2.0575

Zhou, Y., & Wang, R. (2016). An improved flower pollination algorithm for optimal unmanned undersea vehicle path planning problem. *International Journal of Pattern Recognition and Artificial Intelligence*, *30*(04), 1659010. https://doi.org/10.1142/S0218001416590102

Zhou, Y., Wang, R., & Luo, Q. (2016). Elite opposition-based flower pollination algorithm. *Neurocomputing*, *188*, 294–310. https://doi.org/10.1016/j.neucom.2015.01.110

Zou, F., Chen, D., & Xu, Q. (2019). A survey of teaching–learning-based optimization. *Neurocomputing*, *335*, 366–383. https://doi.org/10.1016/j.neucom.2018.06.076