Flower Pollination Algorithm for Test Case Prioritization in Regression Testing

Priyanka Dhareula and Anita Ganpati

Abstract Flower Pollination Algorithm (FPA) is a significant addition made to Nature Inspired Metaheuristic Optimization Algorithms (NIMOA). It is inspired by the pollination process of flowering plants. In this research, FPA is used for Test Case Prioritization (TCP) in Regression Testing (RT). The algorithm uses code coverage of test cases as the input. The algorithm has no prior information of faults covered by the test cases. This study deals with prioritizing (ordering) the test cases in such a way that only those test cases are executed that covers maximum faults in minimum time of execution. For validation of the results Average Percentage of Fault Detected (APFD) metrics is used. APFD values for different ordering of test cases is calculated for three applications written in Java. The empirical results of APFD metrics for FPA order (TS_1) and FPA order (TS_p) are better as compared to Random Order of Original Test Suite (TS_o) and Reverse Random Order of (TS_o). Therefore, this paper states that FPA for TCP gives efficient results in RT.

Keywords Regression testing · Test case prioritization · Flower pollination algorithm · APFD

1 Introduction

Regression Testing (RT) is foremost the crucial part of software life cycle. Whenever software changes RT is done in order to ensure the proper functioning of a software [13]. With the changes in a software, the size of test suite also varies. As the software evolves, the magnitude of test suite size becomes quite large. Rationally it is not feasible to execute all the test cases due to limited time and cost constraints. Therefore, it becomes important to deduce a mechanism to perform efficient RT with maximum number of faults revealing test cases in minimum time. So, test case optimization

P. Dhareula (✉) · A. Ganpati
Computer Science Department, Himachal Pradesh University,
Shimla 171005, Himachal Pradesh, India
e-mail: priyankarana.id@gmail.com

A. Ganpati
e-mail: anitaganpati@gmail.com

S. Fong et al. (eds.), *ICT Analysis and Applications*, Lecture Notes
in Networks and Systems 93, https://doi.org/10.1007/978-981-15-0630-7_16

techniques are required to perform early fault detection in affordable time and cost [3]. Regression Test Case Optimization (RTCO) is further categorized into three main branches of test case optimization namely; Retest All [15], Test Case Selection [4] and Test Case Prioritization (TCP) [14]. In Retest All technique, all the test cases in original test suite are executed. This technique is time consuming and very expensive. Regression Test Case Selection executes subset of a test suite. Whereas, regression TCP techniques [6] deals with maximum and early fault detection by ordering the test cases based on priority given to them. Determining the faults covered by test case in advance is very difficult and time-consuming process, since each test case must be executed in advance to determine its fault coverage.

In the recent times Nature Inspired Metaheuristic Optimization Algorithms (NIMOA) have gained huge popularity [22]. Most of the NIMOA belong to the domain of Stochastic Algorithms (SA). SA are those algorithms whose outcome does not depend on its starting point as they follow a random search. Keeping in view the success of SA numerous algorithms for TCP in RT have been proposed in recent times [6, 7, 12, 21]. In 2012, a new algorithm namely, Flower Pollination Algorithm (FPA) was developed by Xin-She Yang [1]. FPA is inspired by the pollination process in flowering plants. Since, FPA is an optimization algorithm [8] therefore, in this paper TCP was done using FPA without any prior knowledge of faults covered by the test cases.

2 Related Work

Numerous studies have surfaced recently that adopted FPA in different domains like engineering, wireless sensor networking, image and signal processing, communication, structural design, computer gaming, software engineering and global function optimization [2, 11].

Alsewari et al. [1] proposed a strategy for Test Case Minimization based on FPA. Their strategy is termed as Test Generator Flower Pollination Strategy (TGFP). Their research is based on *t*-way testing. Where '*t*' represents the interaction strength between parameters. For different interaction strength their strategy performed well. Nasser et al. [10] proposed a hybridized FPA for generating *t*-way test suite. The main hybridized features used in their research are local search, mutation operator and elitism feature. It is shown in their work that elitism based FPA performed better than other variants of FPA.

Kabir et al. [5] proposed a technique to minimization test suite for *t*-way testing by using adaptive FPA. Their comparative analysis with existing techniques shows that their algorithm showed improved performance. Nasser et al. [9] proposed a Flower Strategy (FS) for generation of *t*-way test suite based on FPA. The research also critically performed the comparison of adoption Optimization Algorithm (OA). Their results show that FS outperforms OA based strategies.

Yadav and Dutta [21] used Genetic Algorithm (GA) to prioritize the test cases by using statement coverage as input to the algorithm. Average Percentage of Statements Covered (APSC) metrics is used to compute the efficiency of the algorithm. Their approach is compared with other techniques of prioritization viz., random prioritization and reverse prioritization. Their study gives optimum results.

Panwar et al. [12] proposed an improved version of Ant Colony Optimization (ACO) algorithm to prioritize the test cases. Triangle classification problem is used in this study. Their proposed technique is compared with different ordering of test cases viz., reverse ordering, random ordering, optimal ordering and original ordering. The antennas of ants are used to share the information for proposed approach. Their approach achieves better results as compared to other orderings of the test cases.

3 Flower Pollination Algorithm (FPA)

FPA was developed in the year 2012 by Xin-She Yang. There are four main rules for flower pollination process as discussed below [22].

I. Global pollination is achieved through biotic and cross-pollination process, and pollen- carrying vectors transfer pollens through Levy flight (Rule 1)
II. Local pollination is achieved through abiotic and self- pollination (Rule 2)
III. Reproduction probability is equivalent to Flower Constancy (FC) that is proportionate to the relationship between two flowers involved (Rule 3)
IV. Switch probability $p \varepsilon [0,1]$ controls the global and local pollination (Rule 4)

Figure 1 represents the FPA developed by Yang [22].

NIMOA are analyzed by the way they explore their search space [22]. In principle all algorithms have two main components i.e., exploitation and exploration which are also known as intensification and diversification respectively. Exploitation is a local search process which can leads to early convergence rate and may also get stuck at local optimum since the final solution depends on starting point. Whereas, exploration explore the global search space leading to a diverse set of solution. Exploration does not get stuck at local optimum [22]. FPA is one of the newest NIMOA [2]. Flower pollination is mainly related to the transfer of pollens, the transfer is achieved through pollinators viz., birds, insects, bats and other animals [23].

Pollination can be achieved in two ways biotic and abiotic. Nearly 90% flowering plants are a member of biotic pollination, the pollens are transferred with the help of vectors. Whereas 10% pollination belong to abiotic form where no vector is needed for transfer of pollens. The transfer of pollens take place with the help of water and wind. Various vectors develop FC [22]. FC can be defined as the affinity that is developed between the vector and the specific flower species. FC thus result in maximum transfer of pollens and reproduction of the same flower species. Biologically, the objective of flower pollination can be explained as the survival of the fittest and optimum reproduction of plants in number as well as fitness. Therefore, this natural process encourages this research to use FPA for TCP for RT.

Maximize or minimize the Fitness Function f(x) for all x_i where i= 1 to n
Create original population of n flower
Identify best flower from original population
Switch probability p ε [0,1] is defined
While (t<Maximum Generation)
For i=1 to n; where n is the number of flowers in the given population
If rand < p then perform global pollination
Perform Levy flight and draw L (step vector)
$x_i^{t+1} = x_i^t + \Upsilon L (g^* - x_i^t)$
Else
Draw ε from a uniform distribution in [0,1]
Perform local pollination as
$x_i^{t+1} = x_i^t + \varepsilon (x_j^t - x_k^t)$
End if
Calculate the fitness of new solution
If the fitness of new solution is better than previous solution then replace the previous solution
End for
Rank the solutions and keep the best solution found so far
End while

Fig. 1 Flower pollination algorithm (FPA)

4 FPA for TCP

This paper uses FPA for TCP to reduce overall time of execution in RT. FPA is used to prioritize test cases from the original test suite (TS_o) to give maximum fault coverage in minimum time of execution. The algorithm uses code coverage criteria as an input to prioritize test cases in TS_o. Original population of test cases is randomly generated and represented as $TS_o = (T_1, T_2, \ldots, T_n)$. Figure 2 shows FPA for TCP.

5 Explanation of FPA for TCP

Original population of 'n' test cases is generated and the test suite is identified as $TS_o = (T_1, T_2, \ldots\ldots, T_n)$. For each test case its code coverage is considered as the fitness function $f(x_i)$, where $x_i = (x_1, x_2, \ldots., x_n)$. The code coverage of test cases in TS_o is used to prioritize the test cases. Test case with highest coverage is maintained as g^*. If multiple test cases have same coverage, then First Come First Serve (FCFS) policy is adopted. A switch variable ($p = 0.8$) [23] is used to perform local (exploitation) or global (exploration) pollination [23]. Random number 'rand' (value ranging between 0 and 1) is used to determine whether local pollination will take place or global pollination will occur. If rand $< p$, global pollination is performed and the fittest reproduction is represented as g^* at tth iteration. In global pollination, the fittest reproduction is ensured through pollens that can travel to long distance. Therefore, for $t + 1$ iteration the pollens/test cases are picked from the tth iteration. Here the pollens are represented as test cases. In flower pollination vector has affinity

Set the Fitness Function f(x) for all x_i where i= 1 to n
Set the original population of n test case as random solutions
Set g* as test case with maximum coverage
Set a switch variable p= 0.8
While (t< stopping criteria)
For i=1: n (all test case)
If rand < p then perform global pollination
Perform Levy flight and draw L (step vector)
$x_i^{t+l} = x_i^t + \Upsilon L\ (g^* - x_i^t)$
Else
Draw ε from a uniform distribution in [0,1]
Perform local pollination as
$x_i^{t+l} = x_i^t + \varepsilon\ (x_j^t - x_k^t)$
End if
Calculate the fitness of new solution
If the fitness of new solution is better than previous solution
then replace the solution
End for
Rank the solutions and keep the best solution found so far
End while

Fig. 2 FPA for TCP

towards the pollens similarly test cases have affinity towards the fault. Those test cases with higher fault detection capability are of great value and are carried forward to the next generation of software. If we represent the fittest as g^*. Then the FC and first rule (Rule 1) can be mathematically formulated as shown in Eq. (1).

$$X_i^{t+1} = X_i^t + \Upsilon\, L\left(g * - X_i^t\right) \tag{1}$$

In Eq. (1) X_i^{t+1} is the ith pollen/test case at $t + 1$ iteration, X_i^t symbolizes the ith pollen or the ith test case at iteration t, L is the strength of the pollination and has a certain probability of distribution. In this paper all test cases/pollens are considered to have FC or affinity toward the vector carrying them for pollination. Hence giving equal chance to all the test cases/pollens to participate in pollination process (TCP). For this reason, L (i.e., step size) is equated to one. To control the step size, Υ i.e., the scaling factor is used. It is assumed that uniform scaling is performed by vector to visit pollens. Therefore, Υ is equated to one. Local pollination described in (Rule 2) and FC can be formulated as shown in Eq. 2.

$$X_i^{t+1} = X_i^t + \varepsilon\left(X_j^t - X_k^t\right) \tag{2}$$

Pollens are transferred from different flowers, but these flowers belong to a single plant species. This simulates the FC in a small neighbourhood. Here $j = i$ and $k = i$

+ 1 in the tth iteration. This emulates the FC behaviour in small area. A local random walk variable ε is derived from a uniform distribution. This paper takes the value of ε as 0 or 1 representing either the test case is selected or not selected for further prioritization; since it may be the case that in a given local area two test case cover the same code. Therefore, redundant test cases are required to be discarded. Here redundant test cases are those test case which give same code coverage. Therefore, fifty percent of the chances of test cases with same coverage being selected is reduced. This result in a reduced and prioritized set of test cases represented as test suite TS_1. TS_1 is exercised upon the application under consideration to determine the number of faults covered by prioritized test suite TS_1. The test cases now in TS_1 are prioritized again to give test suite TS_p. TS_p gives maximum fault coverage in reduced time of execution by removing those test cases that give redundant fault coverage. Effectiveness of the final prioritized test suite TS_p is determined with the help of Average Percentage of Fault Detected Metrics (APFD). Let TS be the test suite having n test cases and let F be the set of m faults identified by TS. Let F_i be first test case in original test suite TS_o which identifies fault i. Therefore, APFD for TS_i is formulated as shown in Eq. (3).

$$\text{APFD} = 1 - \frac{F_1 + F_2 + F_3 + \ldots\ldots + F_m}{nm} + \frac{1}{2n} \tag{3}$$

6 Implementation

FPA for TCP is implemented in Java using Eclipse IDE. Three applications namely, Puzzle Game [16], Area and Perimeter [18] and Tritype [17] are used, which were written in Java programming language. Area and Perimeter calculates the area and perimeter of various mathematical shapes and tritype is a classic triangle classification problem. TestNG tools is used to design the test scripts for all the applications [19]. EclEmma tool is used to get the code coverage for applications used [20]. For the validation of results five versions of Puzzle Game, eight versions of Area and Perimeter and six versions of Tritype applications were created. Each version was seeded with a unique fault. Details of the applications used for experimental evaluation are shown in Table 1.

Table 1 Details of applications used

Sr. no.	Name of the applications	Size in LOC	Number of version/Unique fault	No. of test cases used
1.	Puzzle game	246	5	33
2.	Area and perimeter	916	8	113
3.	Tritype	106	6	45

7 Results

For the explanation of empirical evaluation only one application is considered namely, Puzzle Game. The original test suite (TS_o) for Puzzle Game contains 33 test cases. Test cases are designed in no order viz., $\{T_1, T_2, T_3, T_4, \ldots, T_{25}, T_{26}, \ldots, T_{33}\}$. There are five versions of Puzzle Game application with unique faults induced in each version viz., $\{F_1, F_2, \ldots, F_5\}$. TS_o for Puzzle Game is given as input to the FPA for TCP, the following sequence of test cases is given as the output from FPA i.e., $TS_1 = \{T_3, T_5, T_8, T_{14}, T_{25}, T_{27}, T_{30}\}$. So, the output of FPA is a test suite TS_1 consisting of seven test cases. TS_1 was exercised on all the versions of Puzzle Game application to identify the faults covered by TS_1. Table 2 shows the faults identified by TS_1 for Puzzle Game.

In Table 2, circled cells represent the test case that first identifies a given fault, which is also used to compute APFD metrics. It is evident from Table 2, that test case T_3 identifies maximum faults i.e., F_3, F_4, F_5. Hence, only T_3 test case is enough to identify faults F_3, F_4 and F_5. Test case T_3 is followed by T_5 and T_8 that identifies faults F_2 and F_1 respectively. Therefore, the prioritized sequence in final test suite i.e., $TS_p = \{T_3, T_5, T_8\}$ respectively. Table 3 shows the final test suite TS_p for Puzzle Game with the faults identified by TS_p.

Table 2 Faults identified by TS_1 for puzzle game

Versions/Faults ↓ Test Cases →	T_3	T_5	T_8	T_{14}	T_{25}	T_{27}	T_{30}
F_1 (Version 1)			(1)	1	1		
F_2 (Version 2)		(1)					1
F_3 (Version 3)	(1)					1	
F_4 (Version 4)	(1)	1				1	1
F_5 (Version 5)	(1)	1	1	1	1	1	1

Table 3 Faults identified by TS_p for puzzle game

Versions/ Faults ↓ Test Cases →	T_3	T_5	T_8
F_1 (Version 1)			(1)
F_2 (Version 2)		(1)	
F_3 (Version 3)	(1)		
F_4 (Version 4)	(1)	1	
F_5 (Version 5)	(1)	1	1

TS_p gives complete fault coverage as was given by TS_o. APFD metric value for $TS_1 = \{T_3, T_5, T_8, T_{14}, T_{25}, T_{27}, T_{30}\}$ for Puzzle Game application using FPA is shown in Eq. (4).

$$\text{APFD} = 1 - \frac{8+5+3+3+3}{33 \times 5} + \frac{1}{2 \times 33} \tag{4}$$

$$\text{APFD} = 1 - 0.13333 + 0.01515$$
$$\text{APFD} = 1 - 0.14848$$
$$\text{APFD} = 0.85152 \text{ in } 0.48 \text{ s}$$

Figure 3 shows the priority order and time of execution of TS_1. As it is evident from the Fig. 4 that all the test case for TS_1 failed. Hence, TS_1 was efficient in covering all the faults. Time of execution of TS_1 is 0.48 s. Similarly, for the Prioritized final test suite $TS_p = \{T_3, T_5, T_8\}$ its APFD metric value for Puzzle Game application is shown is Eq. (5).

$$\text{APFD} = 1 - \frac{8+5+3+3+3}{33 \times 5} + \frac{1}{2 \times 33} \tag{5}$$

$$\text{APFD} = 1 - 0.13333 + 0.01515$$
$$\text{APFD} = 1 - 0.14848$$
$$\text{APFD} = 0.85152 \text{ in } 0.286 \text{ s}$$

Figure 4 shows the priority order and time of execution for TS_p. Time of execution of TS_p is 0.268 s which is comparatively lower than the execution time of TS_1.

For the validation of the results APFD metric is computer for different ordering of test cases viz., FPA order (TS_1), FPA order (TS_p), Reverse FPA Order of (TS_p), Random Order of (TS_o) and Reverse Random Order of (TS_o) for Puzzle Game application. It is also evident from Fig. 4 that the Prioritized Test Suite (TS_p) covers all faults with reduced test suite size and reduced time of execution as compared to other ordering.

In Fig. 5 the random order of execution for original Test Suite (TS_o) in 0.94 s is shown.

In Table 4 APFD results and time of execution of all the application for different ordering is shown viz., FPA order (TS_1), FPA order (TS_p), Reverse FPA Order of (TS_p), Random Order of (TS_o) and Reverse Random Order of (TS_o). It is evident from Table 4 that the APFD results for FPA order (TS_p) and FPA order (TS_1) are better as compared to other ordering for all the three applications used in this study.

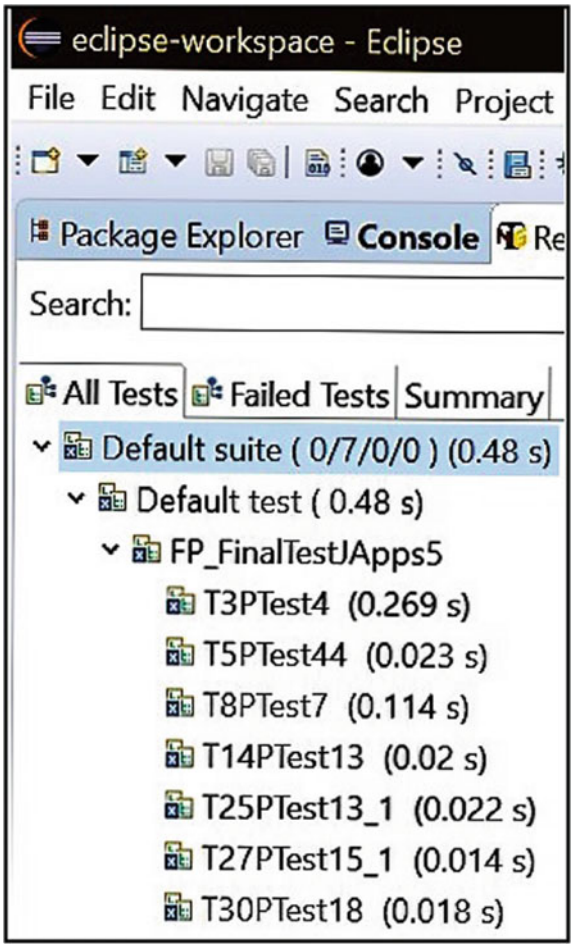

Fig. 3 Order and time of execution of TS_1

In Fig. 6 APFD results and time of execution of all the application for FPA order (TS_1), FPA order (TS_p), Reverse FPA Order of (TS_p), Random Order of (TS_o) and Reverse Random Order of (TS_o) is graphically represented.

8 Threat to Validity

The threat to validity for this study depends on various factors. Few aspects are that, FPA follows a random search mechanism which may result in an inefficient test suite giving minimal fault coverage and hence may not guarantee efficient results for TCP. In the industrial environment the size of applications and number of actual faults can

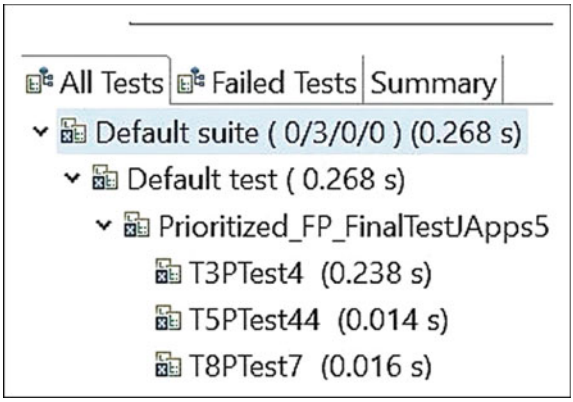

Fig. 4 Order and time of execution of TS_p

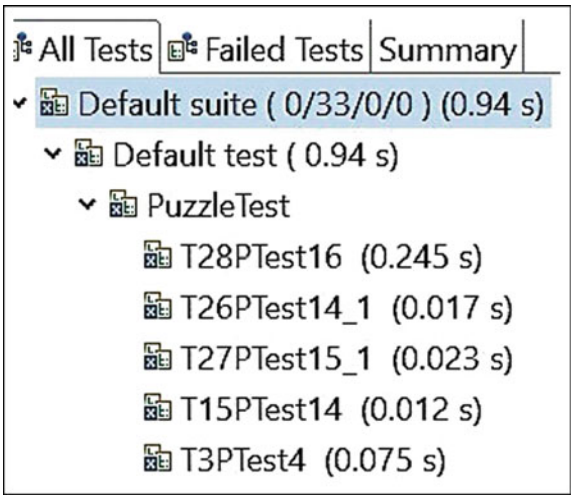

Fig. 5 Order and time of execution of random ordering of Tso

vary from large to very large including the variation in the operating environment, whereas in this study small size application with few seeded faults have been used. The value of control variables has been manipulated to suite the need of this study, which may have otherwise led to inefficient results.

Table 4 APFD results and time of execution for various ordering of test suites in all the applications

Sr. no.	Algorithm used	Puzzle game application			Area and perimeter application			Tritype application		
		Ordering	APFD results	Time (in s)	Ordering	APFD results	Time (in s)	Ordering	APFD results	Time (in s)
1.	FPA result (TS_1)	$\{T_3, T_5, T_8, T_{14}, T_{25}, T_{27}, T_{30}\}$	0.85152	0.48	$\{T_1, T_{43}, T_5, T_7, T_{11}, T_{27}, T_{35}, T_{39}, T_{40}, T_{42}, T_{44}, T_{51}, T_{59}, T_{60}, T_{68}, T_{74}, T_{77}, T_{81}, T_{88}, T_{99}, T_{106}, T_{108}\}$	0.44694	0.107	$T_1, T_{33}, T_2, T_6, T_9, T_{22}, T_{26}, T_{31}, T_{38}, T_{40}, T_{43}$	0.67039	0.05
2.	FPA result (TS_p)	$\{T_3, T_5, T_8\}$	0.85152	0.286	$\{T_1, T_{27}, T_{40}, T_{59}, T_{81}, T_{99}, T_{108}\}$	0.44698	0.088	T_1, T_2, T_6, T_{38}	0.67039	0.04
3.	Reverse order of FPA (TS_p)	$\{T_8, T_5, T_3\}$	0.8099	0.255	$\{T_{108}, T_{99}, T_{81}, T_{59}, T_{40}, T_{27}, T_1\}$	0.43696	0.078	T_{38}, T_6, T_2, T_1	03963	0.045
4.	Random ordering of original test suite (TS_0)	$\{T_{28}, T_{26}, T_{27}, T_{15}, T_3, T_2, T_{20}, T_{13}, T_{12}, T_{30}, T_{11}, T_7, T_{31}, T_8, T_5, T_{29}, T_{19}, T_{14}, T_{22}, T_{17}, T_{33}, T_1, T_{23}, T_4, T_{32}, T_{16}, T_{18}, T_{10}, T_{24}, T_{21}, T_9, T_{25}, T_6\}$	0.38485	0.94	$T_{24}, T_{79}, T_9, T_{85}, T_6, T_{90}, T_{13}, T_{101}, T_{83}, T_{77}, T_{105}, \ldots\ldots\ldots, T_{45}, T_{31}, \ldots\ldots\ldots, T_8, T_{21}, T_{37}, T_{58}, T_{14}, T_{44}, T_{23}, T_{89}$	0.42368	0.792	$T_{40}, T_{11}, T_{36}, T_{41}, T_{43}, T_{12}, T_{30}, T_{18}, T_{15}, T_{16}, T_4, T_{39}, T_{22}, T_{20}, T_{28}, T_2, T_{13}, T_{23}, \ldots\ldots\ldots, T_{44}, T_3, T_{35}, T_{34}, T_7, T_5, T_{42}, T_{29}$	0.23704	0.148
5.	Reverse random ordering test suite (TS_0)	$\{T_6, T_{25}, T_9, T_{21}, T_{24}, T_{10}, T_{18}, T_{16}, T_{32}, T_4, T_{23}, T_1, T_{33}, T_{17}, T_{22}, T_{14}, T_{19}, T_{29}, T_5, T_8, T_{31}, T_7, T_{11}, T_{30}, T_{12}, T_{13}, T_{20}, T_2, T_3, T_{15}, T_{27}, T_{26}, T_{28}\}$	0.27576	0.94	$T_{89}, T_{23}, T_{44}, T_{14}, T_{58}, T_{37}, T_{21}, T_8, \ldots\ldots, T_{31}, T_{45}, \ldots\ldots, T_{105}, T_{77}, T_{83}, T_{101}, T_{13}, T_{90}, T_6, T_{85}, T_9, T_{79}, T_{24}$	0.43363	0.793	$T_{29}, T_{42}, T_5, T_7, T_{34}, T_{35}, T_3, T_{44}, \ldots\ldots\ldots\ldots, T_{28}, T_{20}, T_{22}, T_{39}, T_4, T_{16}, T_{15}, T_{18}, T_{30}, T_{12}, T_{43}, T_{41}, T_{36}, T_{11}, T_{40}$	0.35186	0.148

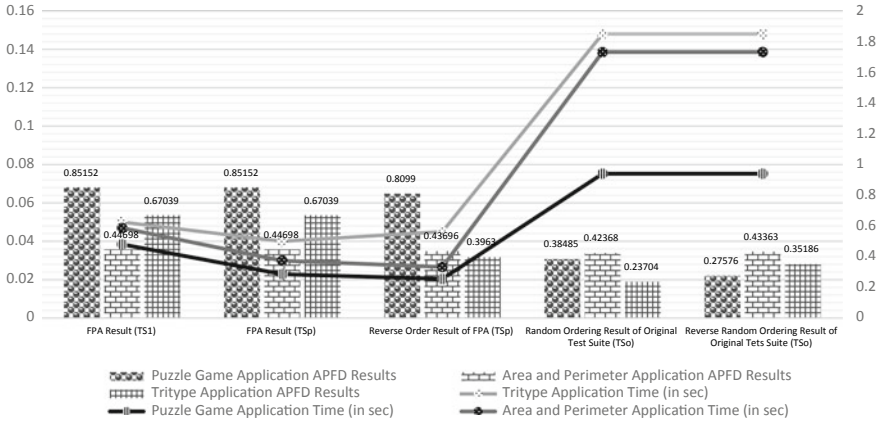

Fig. 6 APFD results for various ordering of test suites for all the applications

9 Conclusion and Future Work

In this research, FPA is used for TCP in RT. FPA uses code coverage of test cases without using any information of faults covered by the test cases. FPA is used to prioritize/order the test cases for RT to give maximum fault coverage in minimum time of execution. Three applications written in Java are used for empirical evaluation. Test scripts for all the applications were designed using TestNG tool. For all the applications three test suite ordering were considered namely, original test suite (TS_o) and two orderings generated by FPA (TS_1) and (TS_p). FPA for TCP initially generated prioritized test suite (TS_1) which was further reduced and prioritized (ordered) to get test Suite (TS_p). For validation of the results APFD metrics is used. APFD value for different orderings of test cases in three applications is calculated. The results of APFD metrics for FPA order (TS_1) and FPA order (TS_p) outperform the results given by Reverse FPA Order of (TS_p), Random Order of (TS_o) and Reverse Random Order of (TS_o). Since FPA for TCP converged early for all the applications used in this study, hence, only one iteration is used. Therefore, it can be stated that stopping criteria of FPA for TCP depends on the type and size of the application being used. For future validation of results an empirical evaluation of FPA for TCP will be performed with different NIMOA.

References

1. Alsewari AA, Har HC, Homaid AA, Nasser AB, Zamli KZ, Tairan NM (2017) Test cases minimization strategy based on flower pollination algorithm. In: International conference of reliable information and communication technology 2017 Apr 23. Springer, Cham, pp 505–512. https://doi.org/10.1007/978-3-319-59427-9_53

2. Chiroma H, Shuib NL, Muaz SA, Abubakar AI, Ila LB, Maitama JZ (2015) A review of the applications of bio-inspired flower pollination algorithm. Procedia Compu Sci 1(62):435–441. https://doi.org/10.1016/j.procs.2015.08.438
3. Harrold MJ, Gupta R, Soffa ML (1993) A methodology for controlling the size of a test suite. ACM Trans Sof Eng Method (TOSEM) 2(3):270–285. https://doi.org/10.1145/152388.152391
4. Jiang B, Wu Y, Zhang Y, Zhang Z, Chan WK (2018) ReTestDroid: towards safer regression test selection for android application. In: 2018 IEEE 42nd annual computer software and applications conference (COMPSAC) 2018 Jul 23, vol 1. IEEE, pp 235–244. https://doi.org/10.1109/compsac.2018.00037
5. Kabir MN, Ali J, Alsewari AA, Zamli KZ (2017) An adaptive flower pollination algorithm for software test suite minimization. In: 2017 3rd international conference on electrical information and communication technology (EICT) 2017 Dec 7. IEEE, pp 1–5. https://doi.org/10.1109/eict.2017.8275215
6. Khatibsyarbini M, Isa MA, Jawawi DN, Tumeng R (2018) Test case prioritization approaches in regression testing: a systematic literature review. Inf Softw Technol 1(93):74–93. https://doi.org/10.1016/j.infsof.2017.08.014
7. Luo Q, Moran K, Poshyvanyk D, Di Penta M (2018) Assessing test case prioritization on real faults and mutants. In: 2018 IEEE international conference on software maintenance and evolution (ICSME) 2018 Sep 23. IEEE, pp 240–251. https://doi.org/10.1109/icsme.2018.00033
8. Nabil E (2016) A modified flower pollination algorithm for global optimization. Expert Syst Appl 15(57):192–203. https://doi.org/10.1016/j.eswa.2016.03.047
9. Nasser AB, Sariera YA, Alsewari AA, Zamli KZ (2015) Assessing Optimization-based strategies for T-way test suite generation: the case for flower-based strategy. In: 2015 IEEE international conference on control system, computing and engineering (ICCSCE) 2015 Nov 27. IEEE, pp 150–155. https://doi.org/10.1109/iccsce.2015.7482175
10. Nasser AB, Zamli KZ, Alsewari AA, Ahmed BS (2018) Hybrid flower pollination algorithm strategies for T-way test suite generation. PLoS ONE 13(5):e0195187. https://doi.org/10.1371/journal.pone.0195187
11. Pant S, Kumar A, Ram M (2017) Flower pollination algorithm development: a state of art review. Int J Syst Assur Eng Manag 8(2):1858–1866. https://doi.org/10.1007/s13198-017-0623-7
12. Panwar D, Tomar P, Harsh H and Siddique MH (2018) Improved meta-heuristic technique for test case prioritization. In: Soft computing: theories and applications. Springer, Singapore, pp 647–664. https://doi.org/10.1007/978-981-10-5687-1_58
13. Pressman RS (2005) Software engineering: a practitioner's approach. Palgrave Macmillan
14. Rothermel G, Untch RH, Chu C, Harrold MJ (1999) Test case prioritization: an empirical study. In: Proceedings IEEE international conference on software maintenance-1999 (ICSM'99). Software Maintenance for Business Change (Cat. No. 99CB36360). IEEE, pp 179–188. https://doi.org/10.1109/icsm.1999.792604
15. Sivaji U, Shraban A, Varalaxmi V, Ashok M, Laxmi L (2019) Optimizing regression test suite reduction. In: First international conference on artificial intelligence and cognitive computing. Springer, Singapore, pp 187–192. https://doi.org/10.1007/978-981-13-1580-0_18
16. https://github.com/jkriti/JApps/blob/master/JApps.java
17. https://sir.csc.ncsu.edu/portal/bios/trityp.php
18. https://stackoverflow.com/questions/4479624/area-and-perimeter-calculation-of-various-shapes?rq=1
19. https://testng.org/doc/download.html
20. https://www.eclemma.org/download.html
21. Yadav DK and Dutta S (2017) Regression test case prioritization technique using genetic algorithm. In: Advances in computational intelligence 2017. Springer, Singapore, pp 133–140. https://doi.org/10.1007/978-981-10-2525-9_13
22. Yang XS (2014) Nature-inspired optimization algorithms. Elsevier
23. Yang XS, Karamanoglu M, He X (2014) Flower pollination algorithm: a novel approach for multiobjective optimization. Eng Optim 46(9):1222–1237. https://doi.org/10.1080/0305215X.2013.832237