© Universiti Tun Hussein Onn Malaysia Publisher's Office

JSCDM

http://penerbit.uthm.edu.my/ojs/index.php/jscdm

e-ISSN : 2716-621X

Journal of Soft Computing and Data Mining

Comparative Analysis for Test Case Prioritization Using Particle Swarm Optimization and Firefly Algorithm

Lee Zhiang Yue[1], Rosziati Ibrahim[1]*

[1]Faculty of Computer Science and Information Technology,
Universiti Tun Hussein Onn Malaysia, Batu Pahat, 86400, MALAYSIA

*Corresponding Author

DOI: https://doi.org/10.30880/jscdm.2023.04.02.007
Received 14 February 2023; Accepted 21 September 2023; Available online 04 October 2023

Abstract: Software testing is the most importance phase for software development life cycle. However, it is always time consuming and costly. In order to solve this problem, regression testing is required to be conducted since it can verify the software modifications with zero effect to the software actual features. Test Case Prioritization (TCP) is one type of regression testing techniques. It can reduce the cost and time taken. In the area of TCP, there are several algorithms. This study will focus on the comparison analysis of prioritization of test case by using Particle Swarm Optimization (PSO) and Firefly Algorithm (FA). In order to choose an algorithm with better performance between PSO and FA, they are converted into Python code. Then, PSO and FA are implemented into Case Study A and Case Study B. Their result will be compared and analyzed based on the execution time, Big-O, and APFD. The comparison showed that FA outperforms than PSO since FA has the least execution time (0.001 second), less complexity (O(N)) than PSO (O(N^3)), and similar APFD values (0.520 and 0.600). Thus, FA has better prioritization performance compared to PSO.

Keywords: Software testing, Test Case Prioritization (TCP), Particle Swarm Optimization (PSO), Firefly Algorithm (FA)

1. Introduction

Nowadays, there is a high demand for software used towards most of the field in the world such as education, business, and financial around the world. After the software has been deployed to the customer, it will soon become invaluable for the user without periodic updates. In order to reduce errors that occur after updating the software, it should be tested before deployed to the user. But the process of software testing is always time consuming and costly. There is one type of software testing called regression testing which can verify the software modifications with zero effect to the software actual features [1]. Test Case Prioritization (TCP) is one of the regression testing techniques which can be applied in software testing in order to reduce the time and cost. It can find the priorities for the test cases that can reveal system defects [2]. In the prioritization of test cases, only the most important test cases will be chosen to execute. Those test cases with the highest priority are executed earlier. TCP can enhance the software qualities within a limited time and cost [3]. Due to the high cost and time consuming of software testing, TCP is one of the methods that can reduce the time and cost. In this study, there are two types of algorithms that will be applied in test case prioritization which are Particle Swarm Optimization (PSO) and Firefly Algorithm (FA).

This study will focus on two objectives. Firstly, the objective is to develop PSO and FA in test case prioritization by using Python. Secondly, the study wants to compare and analyze the performance of PSO and FA for TCP in Case Study A and Case Study B based on the execution time, Big-O, and Average Percentage of Faults Detected (APFD).

*Corresponding author: rosziati@uthm.edu.my

This article is structured as follows: The review of associated works is covered in Section 2. In Section 3, the research technique is covered. The experimental findings, discussion, and comparison with current techniques are presented in Section 4. Finally, Section 5 offers findings and ideas for future research.

2. Literature Review

This section contains the overview for regression testing, TCP, PSO, and FA. The related work done by other researchers has been reviewed to identify suitable algorithm and evaluation parameter for conducting this study.

2.1 Regression Testing

Regression testing ensures that the latest version of the software do not affect its existing function. However, executing all test cases is costly and time consuming. Hence, researchers will find solution to minimize the time taken and the costs of regression testing. There are four types of methods that can be used to reduce the time and costs for the regression testing process which are Test Case Prioritization, Test Suite Minimization, Test Case Selection, and Retest All [4].

2.2 Test Case Prioritization

In order to reduce the time and cost spent in software testing, TCP is implemented to sort the sequence of test case execution by using different techniques. These techniques include Particle Swarm Optimization (PSO) [5], Firefly Algorithm (FA) [6], Ant Colony Optimization (ACO) [7], Black Hole Algorithm (BHA) [8], Genetic Algorithm (GA) [9], and Cuckoo Search Algorithm (CSA) [10] which are powerful nature-inspired algorithm. TCP allows testers to focus more on filtering and labeling samples with high priorities [2]. It can help to determine a better sequence for the execution of test cases. Apart from this, TCP can reduce the testing time and cost [11].

2.3 Particles Swarm Optimization

PSO is an optimization algorithm which was inspired from the activities of flocks of birds in food searching [12]. In the process of finding food, each bird will always observe its neighbor. Subsequently, it will update its position and velocity when it senses that find the neighbors are closer to the food. Consequently, the birds will return to the nest after finding the food. In PSO, the best path is the path with the least distance between the nest and food [13]. The goal of this algorithm is to shorten the path and decrease the execution time to solve the problem. Fig. 1 shows the flowchart PSO for TCP. First, the initial population of the birds is generated. Then, their position in the test case list is determined randomly and the birds will start random movement. At the same time, they will observe the position and velocity of the other birds. They will keep updating their position and velocity based on the global best position. The bird creates a test case order list for each iteration. When the termination criteria which is the maximum iteration number are satisfied, the travelled path of the birds will be recorded and the distance of the path will be calculated. The process ends here. However, if the termination criteria are not satisfied, it will return back to the step of observing the position and velocity of its neighbor and repeat the steps.

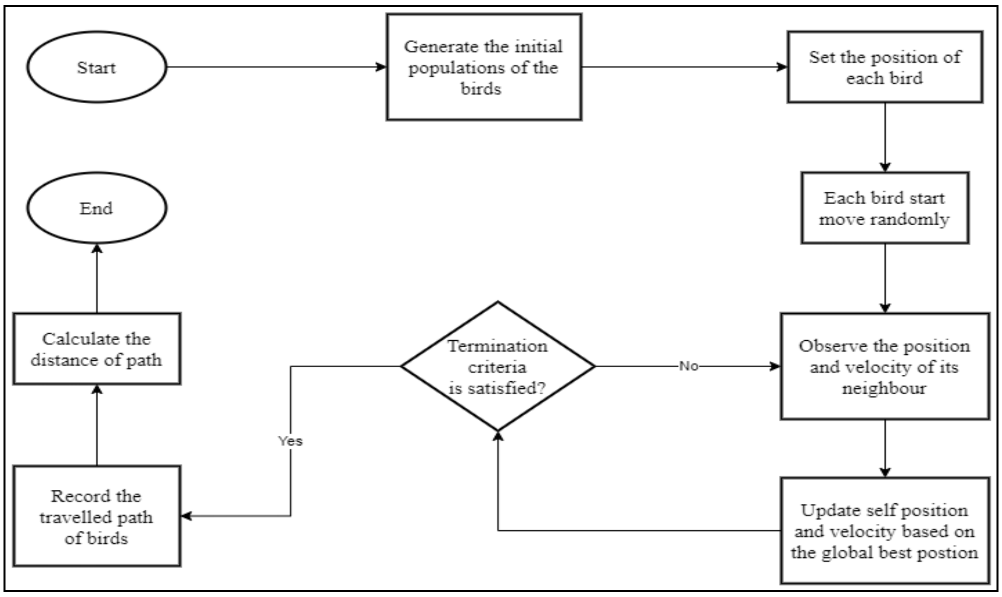

Start
Generate the initial populations of the birds
Set the position of each bird
Each bird start move randomly
Observe the position and velocity of its neighbour
Update self position and velocity based on the global best postion
Termination criteria is satisfied?
No
Yes
Record the travelled path of birds
Calculate the distance of path
End

Fig. 1 - Flowchart for Particle Swarm Optimization (PSO)

2.4 Firefly Algorithm

FA is an algorithm which was developed based on the natural behavior of fireflies. The position of fireflies is the solution. The attractiveness of a firefly depends on its brightness. A firefly will always be attracted to the brighter fireflies. The distance between the fireflies will be calculated. After visiting all fireflies, they will stop moving. The visited path of the fireflies is recorded for the prioritization of test case [6]. Fig. 2 shows the flowchart of FA in TCP. At the beginning, FA generates the population of fireflies. Next, FA creates the light intensity of fireflies. After that, the distance between the fireflies and its brightness is calculated to determine the attractiveness of each firefly. The fireflies then move to the brighter non-visited fireflies. After visiting all fireflies, the movement path is recorded and the distance path will be calculated. In FA, the shortest distance is the best path for the prioritization of test case.

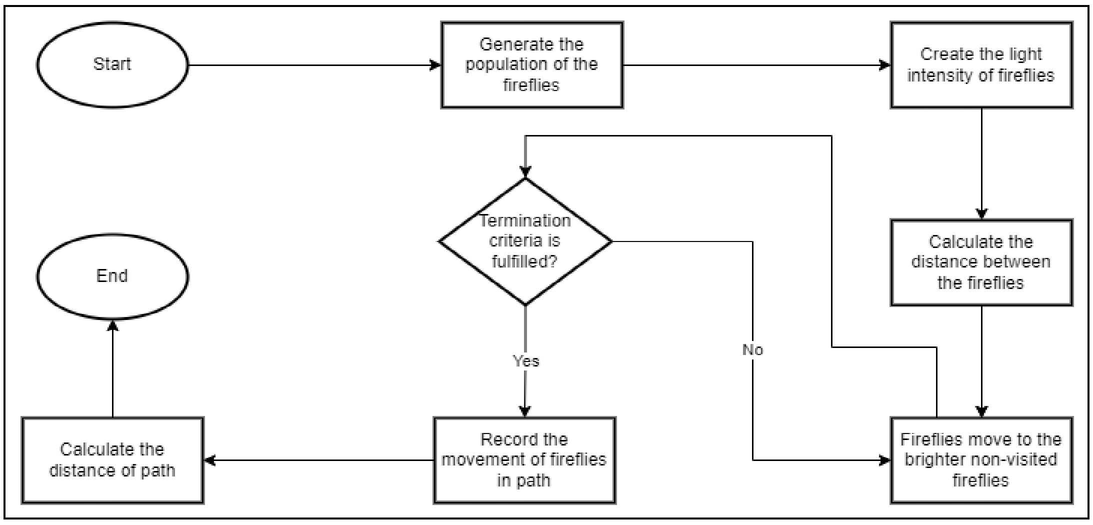

Start
Generate the population of the fireflies
Create the light intensity of fireflies
Calculate the distance between the fireflies
Fireflies move to the brighter non-visited fireflies
Termination criteria is fulfilled?
No
Yes
Record the movement of fireflies in path
Calculate the distance of path
End

Fig. 2 - Flowchart of Firefly Algorithm (FA)

2.5 Performance Evaluation Parameters

In order to identify the performance of PSO and FA for test case prioritization, there are three parameters involved in this study, which are the execution time, Big-O and also APFD. The parameters used are described in the following details. The execution time is measured in seconds. The execution time is acquired by deducting the end time from the start time. The shorter the execution time, the better the performance. Equation (1) shows the formula of the execution time.

$$Execution\ Time = End\ Time - Start\ Time \tag{1}$$

Big-O is used to represent the time and space complexity of an algorithm. The time complexity for constant value, linear statement, quadratic values, and cubic values are O(1), O(N^2), and O(N^3) respectively [14]. In order to identify the required storage and time of an algorithm, the complexity of space and time are calculated. By calculating the number of methods, the time complexity can be estimated. Also, the complexity of space is calculated with the input size. Both time and space complexity are expressed with Big-O [15]. Equation (2) until (4) shows the calculation of Big-O.

$$O(N) = for\ loop \tag{2}$$

$$O(N^2) = \begin{array}{l} for\ loop \\ \quad for\ loop \end{array} \tag{3}$$

$$O(N^3) = \begin{array}{l} for\ loop \\ \quad for\ loop \\ \qquad for\ loop \end{array} \tag{4}$$

APFD is applied to measure the weighted average percentage of faults detected over the list of a test case. It represents the fault detection rates with a value from 0 to 100. The greater the value, the faster the fault detection rate [11]. Equation (5) shows the formula of APFD.

$$APFD = 1 - \frac{TF_1 \ + TF_2 \ + TF_m \ \dots}{nm} + \frac{1}{2n} \tag{5}$$

where
TF - test suite
m - faults number
n - test cases number
TFi - the position of the first test in test suite that exposes fault *i*

2.6 Related Work

This subsection discussed the related works with this study. Table 1 shows the summary of related works.

Table 1 - Summary of related work

Ref No.	Performance
[6]	FA has better performance than Ant Colony Optimization in test case prioritization. Parameters: Fault coverage, execution time, APFD
[16]	Artificial Fish School Algorithm could better solve the TCP problems.
[2]	TCP with neuron valuation-based pattern perform better than most of the algorithms. Parameters: APFD
[17]	The combination of PSO and Crow Search Algorithm (CSA) will produce highly optimized and prioritized the test cases. Note: PSO is commonly used in TCP
[18]	The proposed technique performance better than other techniques in APFD and execution time. Parameters: execution time, APFD
[19]	Analyzed a few clustering techniques that applied to test case prioritization. Parameters: execution time, APFD
[11]	Improved Quantum-behaved particle swarm PSO has better performance compared to all other algorithms in the reduction of test case and cost. Parameters: APFD
[20]	A method to generate optimal test paths by using FA. Parameters: Percentage Redundancy in path coverage
[21]	Use FA to analyze, generate, and optimize random test cases. Parameters: Test case generation
[22]	TCP by using FA. APFD results show that FA is a strong competitor in TCP. Parameters: APFD, execution time
[23]	Generate Big-O Analysis of algorithms automatically. Note: Big-O

[24]	Applied multi-objective FA to solve design optimization benchmarks and multi-objective optimization problems. Note: Using FA in optimization

According to Xing et al. [16], TCP uses swarm intelligence algorithms to preprocess the test suite. The study showed that the Artificial Fish School Algorithm can solve the problems of TCP and improve the efficiency of software testing in terms of both single objective and multiple objectives. Ariffin et al. [6] presented TCP by using ACO and FA and evaluated the performance based on fault coverage, execution time, and APFD. Results showed that FA has better performance since it has the highest Big-O and the lowest execution time.

Yan et al. [2] presented TCP with neuron valuation-based pattern. The results showed that the prioritization method performed better than other existing techniques. The parameters used is APFD. Senthil et al. [17] have proposed a hybrid method in TCP by using PSO. He argued that the combination of PSO and CSA can improve the quality of the software by optimized and prioritized test cases. Su et al. [18] applied a meta-heuristic TCP method in hybrid model. Results showed that the proposed method performed better than other techniques in terms of the execution time and APFD.

Bajaj et al. [11] proposed that Improved Quantum-behaved particle swarm optimization has better performance compared to all other algorithms in the reduction of test case and cost. The parameters used are APFD. Srivatsava et al. [20] proposed a method to generate optimal test paths by using FA. It has minimized the efforts and gives the best test paths. The parameters used was percentage redundancy in path coverage. Mohapatra et al. [21] has proposed to use FA to analyze, generate, and optimize random test cases in his study. This study also briefed about the firefly method which is used for the optimization and generation of test case. Khatibsyarbini et al. [22] has conducted a study about the TCP by using FA. The APFD results showed that FA is a strong competitor in TCP.

Vaz et al. [23] has presented a way to generate Big-O Analysis of algorithms automatically. This study was conducted due to the manual calculation for Big-O is time consuming. By applying this automatic method, the researcher can get the Big-O analysis result quickly and easily. This study is mainly focused on the automated big o analysis for algorithm. Yang [24] applied multi-objective FA to solve design optimization benchmarks and multi-objective optimization problems. The results showed that multi-objective FA is more efficient than previous.

Based on Table 1, the commonly used algorithms to be implemented into test case prioritization are PSO and FA while the common parameters are the execution time, Big-O, and APFD. Hence, in this study, the algorithms used are PSO and FA while the evaluation parameters used will be the execution time, Big-O, and APFD.

3. Research Methodology

Research methodology describes the whole process and the method used to complete this study. It contains the research process, case study, and the implementation of the algorithm.

3.1 Case Study

PSO and FA will be implemented into two case studies which are Case Study A and Case Study B. They are two different graph which consists of cost, destinations, and paths. Case Study A has five destinations while Case Study B has four destinations. Every destination is linked with a path and each path has a cost which is called as distance. The original ordering of the graph for Case Study A and B are {1, 2, 3, 4, 5} and {1, 2, 3, 4} respectively. The graph data of Case Study A and Case Study B was turned into adjacency matrix in Equations (6) and (7) to be executed by FA and PSO for prioritizing the destination.

$$Case\ Study\ A = \begin{bmatrix} 0 & 14 & 8 & 12 & 14 \\ 14 & 0 & 5 & 7 & 15 \\ 8 & 5 & 0 & 18 & 11 \\ 12 & 7 & 18 & 0 & 9 \\ 14 & 15 & 11 & 9 & 0 \end{bmatrix} \tag{6}$$

$$Case\ Study\ B = \begin{bmatrix} 0 & 6 & 11 & 8 \\ 6 & 0 & 9 & 7 \\ 11 & 9 & 0 & 7 \\ 8 & 7 & 7 & 0 \end{bmatrix} \tag{7}$$

3.2 Implementation of Algorithms

The dataset in Case Study A and Case Study B will be applied in PSO and FA. In this study, Python is the programming language that is used to apply FA and PSO on the case studies. The tools used to run the algorithm is PyCharm. The steps to implement these two algorithms are described further.

For PSO, initializing the number of artificial birds and iterations will go through. After that, release all the artificial birds into the path and they will travel randomly. Each travelled path and its time taken will be recorded. They will compare the global best position and velocity with its neighbors in order to change their location and velocity based on it. Each bird will become closer towards the best position once the velocity is changed. All these steps have been transformed into programming code. The shortest path will be found after sorting all the travelled path. After the artificial birds finished all the iterations, the execution time will be stopped and calculated. The shortest travelled path, total costs of the path, and the execution time will be printed out. The same process has been implemented to Case Study A and B. The number of iterations for Case Study A and B are 100. The iteration number used 100 because it is an ideal iteration in TCP. Meanwhile, the number of test cases for Case Study A and B are 5 and 4 respectively.

FA needs to use light intensity to prioritize the test case instead of use the distance between the fireflies only. Thus, FA has two graphs which are cost graph (also called as distance) and the light intensity graph. The number of artificial fireflies is based on the matrix of the light intensity graphs. First, all the related library is imported and the time from the beginning is recorded. Second, all the artificial fireflies are released into the light intensity graphs and they will move to the solution. The artificial fireflies are released randomly. They will be attracted to the higher light intensity. The position of the artificial fireflies is the solution to the problem. In order to identify the shortest path, all of the paths travelled by the artificial fireflies are recorded and sorted. The time measurement will be stopped and calculated after all artificial fireflies have completed the iterations and the path has been recorded. Lastly, the output of the shortest path, total cost of the path, and the execution time will be printed out. The number of iterations for Case Study A and B are 100. The iteration number used 100 because it is an ideal iteration in TCP. Meanwhile, the number of test cases for Case Study A and B are 5 and 4 respectively. Equations (8) and (9) show the data of light intensity graphs for Case Study A and B respectively. The data are randomly put follow the size of the cost graph which are 5X5 and 4X4 for Case Study A and Case Study B respectively. The negative data indicates that the light cannot be seen, which is considered as weak brightness while the positive data indicates that the light can be seen, which is considered as strong brightness. The fireflies will always be attracted to the brighter non-visited fireflies. Otherwise, the fireflies will move randomly.

$$\text{Light Intensity A} = \begin{bmatrix} -0.004 & -0.303 & -0.126 & -0.327 & -0.232 \\ -0.124 & -0.129 & -0.043 & -0.232 & -0.291 \\ 0.102 & 0.383 & 0.015 & 0.007 & 0.271 \\ 0.064 & 0.108 & 0.201 & 0.184 & 0.261 \\ 0.112 & 0.162 & 0.173 & 0.181 & 0.124 \end{bmatrix} \tag{8}$$

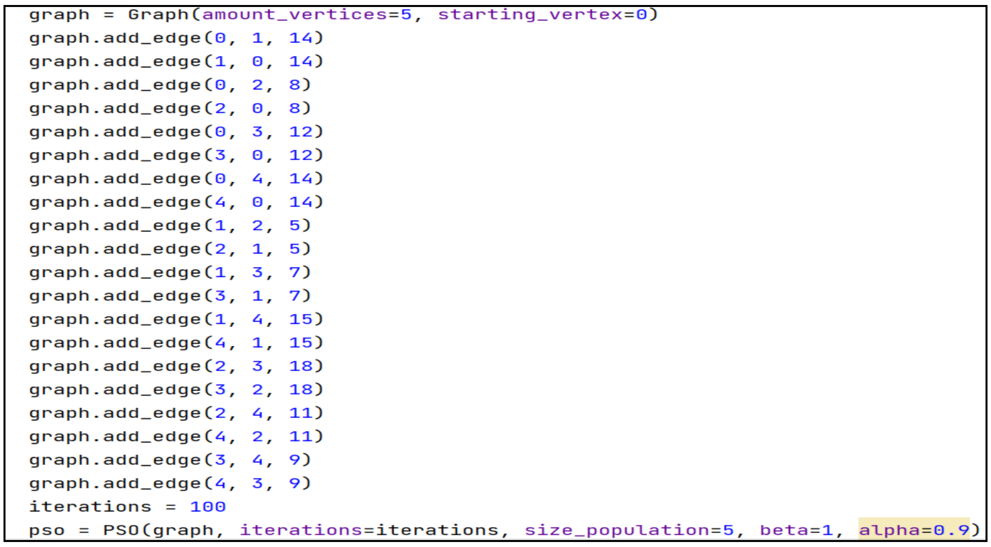
$$\text{Light Intensity B} = \begin{bmatrix} -0.004 & -0.313 & -0.326 & -0.027 \\ -0.124 & -0.229 & -0.343 & -0.312 \\ 0.002 & 0.283 & 0.275 & 0.127 \\ 0.264 & 0.278 & 0.211 & 0.264 \end{bmatrix} \tag{9}$$

4. Results and Analysis

After PSO and FA are executed into Case Study A and B, their performances are compared and analyzed based on the execution time, Big-O, and APFD.

4.1 Particle Swarm Optimization

For PSO in Case Study A and B, the path consists of five artificial birds and five destinations with different paths and costs while the path of PSO in Case Study B consists of four artificial birds and four destinations with different paths and costs. After releasing the artificial birds into the path, they will go through the path with 100 iterations. After all paths have been travelled, the best paths of PSO in Case Study A for these test cases are {1, 5, 4, 2, 3, 1} with 43 costs while the best paths of PSO in Case Study B for these test cases are {1, 4, 3, 2, 1} with 30 costs. The execution time for PSO in Case Study A and B are 0.005 seconds and 0.004 seconds respectively. Based on the source code PSO, the Big-O is O(N^3). APFD value for PSO in Case Study A and B are 0.520 and 0.600 respectively. Fig. 3 shows the segmentation source code of PSO in Case Study A.

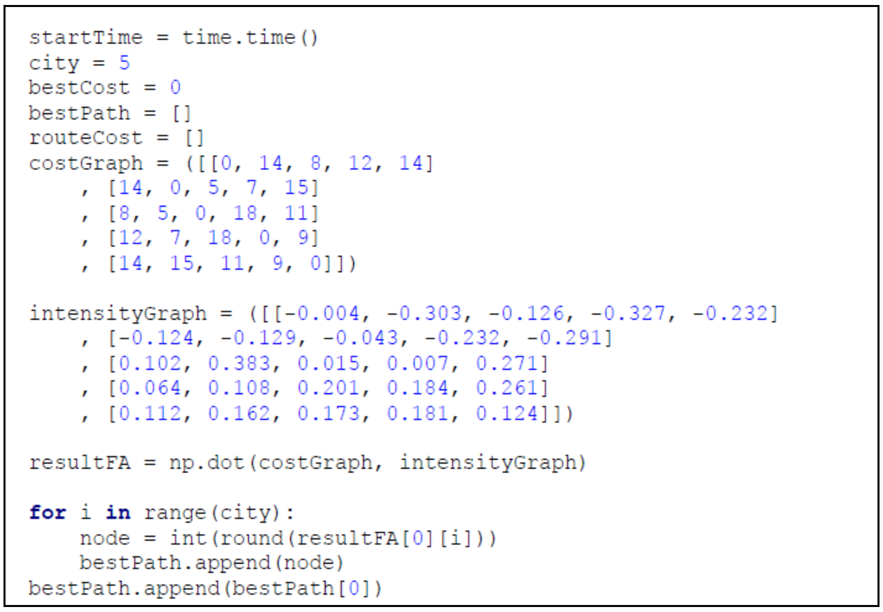


```
graph = Graph(amount_vertices=5, starting_vertex=0)
graph.add_edge(0, 1, 14)
graph.add_edge(1, 0, 14)
graph.add_edge(0, 2, 8)
graph.add_edge(2, 0, 8)
graph.add_edge(0, 3, 12)
graph.add_edge(3, 0, 12)
graph.add_edge(0, 4, 14)
graph.add_edge(4, 0, 14)
graph.add_edge(1, 2, 5)
graph.add_edge(2, 1, 5)
graph.add_edge(1, 3, 7)
graph.add_edge(3, 1, 7)
graph.add_edge(1, 4, 15)
graph.add_edge(4, 1, 15)
graph.add_edge(2, 3, 18)
graph.add_edge(3, 2, 18)
graph.add_edge(2, 4, 11)
graph.add_edge(4, 2, 11)
graph.add_edge(3, 4, 9)
graph.add_edge(4, 3, 9)
iterations = 100
pso = PSO(graph, iterations=iterations, size_population=5, beta=1, alpha=0.9)
```

Fig. 3 - Segmentation source code for PSO

4.2 Firefly Algorithm

For FA in Case Study A and B, it must have the light intensity graph and cost graph in order to solve the prioritization of test case. The matrices *C.I* = *P* is applied in the source code where *C* is the cost of the path, *I* is the light intensity graph, and *P* is the solution. The artificial fireflies are released randomly and are attracted to the highest light intensity to find the solution from the intensity graph. The position of the artificial fireflies is the optimization solution. After all paths have been travelled, the best paths for these test cases for Case Study A and B are {1, 5, 4, 2, 3, 1} and {1, 4, 3, 2, 1} respectively with the execution time of 0.001 seconds. The APFD value is 0.520 and 0.600 respectively. Based on the source code FA, the Big-O is O(N). After completing all iterations, the artificial fireflies obtained 43 costs and 30 costs from the travelled path in Case Study A and B respectively. Fig. 4 shows the segmentation source code for FA.



```
startTime = time.time()
city = 5
bestCost = 0
bestPath = []
routeCost = []
costGraph = ([[0, 14, 8, 12, 14]
    , [14, 0, 5, 7, 15]
    , [8, 5, 0, 18, 11]
    , [12, 7, 18, 0, 9]
    , [14, 15, 11, 9, 0]])

intensityGraph = ([[-0.004, -0.303, -0.126, -0.327, -0.232]
    , [-0.124, -0.129, -0.043, -0.232, -0.291]
    , [0.102, 0.383, 0.015, 0.007, 0.271]
    , [0.064, 0.108, 0.201, 0.184, 0.261]
    , [0.112, 0.162, 0.173, 0.181, 0.124]])

resultFA = np.dot(costGraph, intensityGraph)

for i in range(city):
    node = int(round(resultFA[0][i]))
    bestPath.append(node)
bestPath.append(bestPath[0])
```

Fig. 4 - Segmentation source code for FA

4.3 Comparison of Execution Time

The execution time is vital for any software. Based on the Equation (1), the execution time is calculated by using the program source code. Table 2 shows the average execution time for each algorithm used in this study. Based on Table 2, the execution time for both algorithms is different. The execution time of FA for Case Study A and B are 0.001 second while the execution time of PSO for Case Study A and B is 0.005 seconds and 0.004 seconds respectively. FA uses less execution time for both case studies in TCP compared to PSO. Fig. 5 shows the graph for the comparison of execution time.

Table 2 - Results of average execution time

Algorithm	Execution Time (Seconds)
PSO (Case Study A)	0.005
PSO (Case Study B)	0.004
FA (Case Study A)	0.001
FA (Case Study B)	0.001

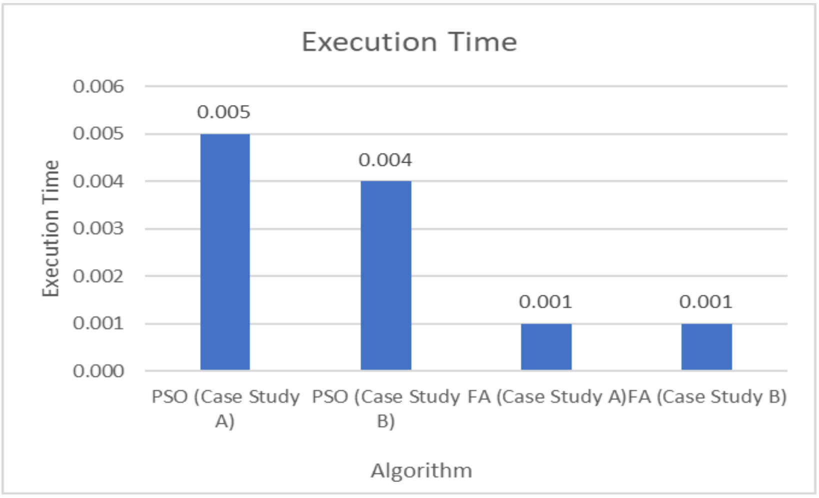


Fig. 5 - Comparison of execution time

4.4 Comparison of Big-O

The results for the Big-O are shown in Table 3. Based on Table 3, the Big-O of PSO is $O(N^3)$ while FA is O(N). The results show that the complexity of PSO is higher than FA.

Table 3 - Results of Big-O

Algorithm	Big-O
PSO	$O(N^3)$
FA	O(N)

4.5 Comparison of APFD

APFD is applied to find faults in test cases. For PSO, the APFD value is 0.520 and 0.600 for Case Study A and B respectively. For FA, the APFD value is 0.520 and 0.600 for Case Study A and B respectively. Table 4 shows the APFD results for each algorithm and case studies.

Table 4 - Results of APFD values

Algorithm	APFD Values
PSO (Case Study A)	0.520
PSO (Case Study B)	0.600
FA (Case Study A)	0.520
FA (Case Study B)	0.600

Based on Table 4, PSO and FA have same APFD value which are 0.520 for Case Study A and 0.600 for Case Study B. Fig. 6 shows the graph for the comparison of APFD values.

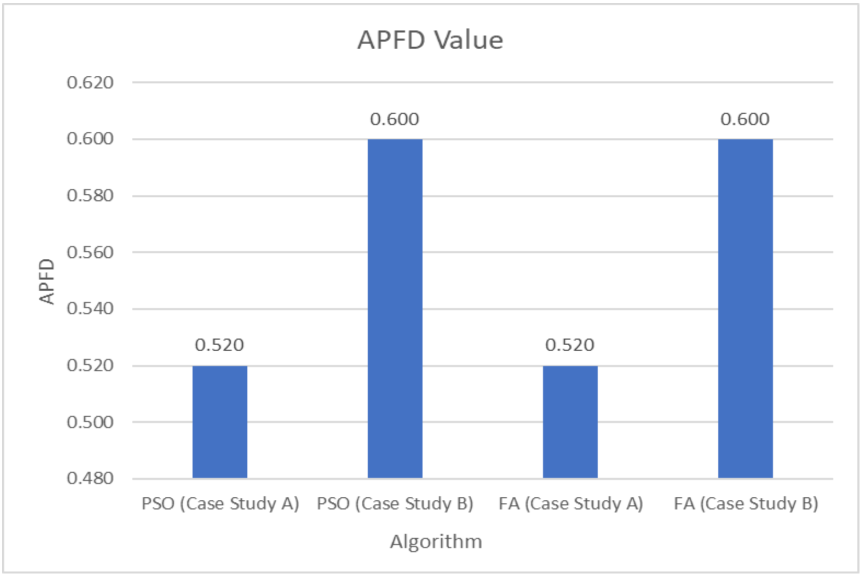


Fig. 6 - Comparison of execution time

4.6 Results and Discussion

The performance of PSO and FA in Case Study A and B are being compared and analyzed. They are evaluated based on the execution Time, Big-O, and APFD in order to identify a better algorithm for TCP. The algorithm with a higher APFD value means it has better performance. For the execution time, the algorithm with minimum time taken means it has better performance. For Big-O, the algorithm with lower complexity has good performance. Table 5 shows the summarization of results for PSO and FA.

Table 5 – Summar ization of results for PSO and FA

Algorithm	PSO	PSO	FA	FA
Measurement	**(Case Study A)**	**(Case Study B)**	**(Case Study A)**	**(Case Study B)**
Execution Time	0.005	0.004	0.001	0.001
Big-O	$O(N^3)$		$O(N)$	
APFD	0.520		0.600	

Based on Table 5, FA and PSO have similar APFD value for Case Study A and B which are 0.520 and 0.600 respectively due to the small size of case study. For the result of execution time, FA spends 0.001 seconds for both Case Study A and B while PSO spends 0.005 seconds and 0.004 second for Case Study A and B respectively. The execution time of FA is less than PSO. The Big-O of PSO and FA are $O(N^3)$ and $O(N)$ respectively. The complexity of FA is lower than PSO. Since the execution time and Big-O of FA are lower than PSO, the overall performance of FA is considered better than PSO. Fig 7 shows the graph for summarization of results for PSO and FA.

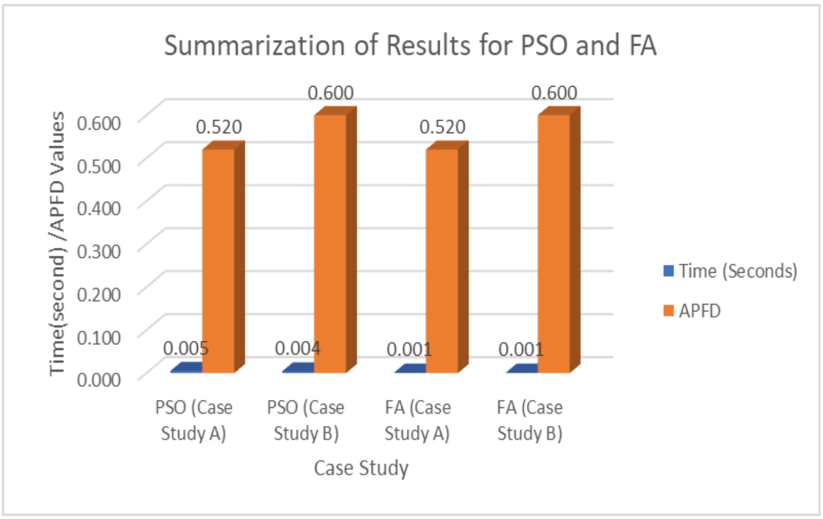


Fig. 7 - Graph for summarization of results for PSO and FA

5. Conclusion

This study had conducted a comparison analysis of prioritization of test case by using PSO and FA. In order to choose an algorithm with better performance between PSO and FA, the algorithms are turned into Python code and then implemented on Case Study A and B. Based on the result in Section 4, the performance result has been compared and analyzed based on the execution time, Big-O, and APFD. The comparison showed that FA outperforms PSO since FA has the least execution time (0.001 seconds), less complexity (O(N)) compared to PSO (O(N^3)), and same APFD values (0.520 and 0.600). The result indicates that FA has faster execution rate than PSO in the prioritization of test case since Big-O is consistent with the execution. Thus, FA has better prioritization performance compared to PSO. The contributions for this study are to find out the algorithm with better performance in test case prioritization between PSO and FA. Based on the comparative and analysis result in Section 4, in the field of the prioritization of test case, FA has better performance than PSO. FA has similar APFD value with PSO. FA is faster in the aspect of execution time. It can help to reduce the time spent on conducting software testing. At the same time, the cost can also be decreased since the time consumption is decreased. The execution rate FA also consistent to the its lower complexity of the algorithm. Hence, FA outperforms PSO. In the area of software testing, test case prioritization is important to reduce cost and time. In future, the outcome of this study can be used to compare with other latest nature inspired algorithms. For example, the comparative analysis for the other two algorithms such as Firefly Algorithm and Black Hole Algorithm or Firefly Algorithm and Cuckoo Search Algorithm. Another suggestion is to introduce hybrid algorithms to overcome the weakness of the single algorithm. The combination may use Firefly Algorithm and Artificial Fish School Algorithm or use Firefly Algorithm and Ant Colony Optimization Algorithm.

Acknowledgement

The authors would like to thank the Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia, for supporting this work.

References

[1] Sivaji, U. & Srinivasa, R. P. (2021). Test case minimization for regression testing by analyzing software performance using the novel method. Mater Today Proc. doi: 10.1016/j.matpr.2021.01.882.
[2] Yan, R., Chen, Y., Gao, H., Yan, J. (2022). Test case prioritization with neuron valuation based pattern. Sci Comput Program, 215. doi: 10.1016/j.scico.2021.102761.
[3] Qasim, M., Bibi, A., Hussain, S. J., Jhanjhi, N. Z., Humayun, M., Sama, N. U. (2021). Test case prioritization techniques in software regression testing: An overview. International Journal of Advanced and Applied Sciences, 8(5), 107–121. doi: 10.21833/ijaas.2021.05.012.
[4] Shin, M. K., Ghosh, S., Vijayasarathy, L. R. (2022). An empirical comparison of four Java-based regression test selection techniques. Journal of Systems and Software, 186. doi: 10.1016/j.jss.2021.111174.
[5] Ashraf, E., Mahmood, K., Ahmed, T., Ahmed, S. (2017). Value based PSO Test Case Prioritization Algorithm. International Journal of Advanced Computer Science and Applications, 8(1). doi: 10.14569/ijacsa.2017.080149.
[6] Ariffin, M. A., Ibrahim, R., Ibrahim, I. S., Wahab, J. A. (2022). Test Cases Prioritization Using Ant Colony Optimization and Firefly Algorithm. International Journal of Engineering Trends and Technology, 70(3), 22–28. doi: 10.14445/22315381/IJETT-V70I3P203.

[7] Zhang, W., Qi, Y., Zhang, X., Wei, B., Zhang, M., Dou, Z. (2019). On test case prioritization using ant colony optimization algorithm. in Proceedings - 21st IEEE International Conference on High Performance Computing and Communications, 17th IEEE International Conference on Smart City and 5th IEEE International Conference on Data Science and Systems, 2767–2773. doi: 10.1109/HPCC/SmartCity/DSS.2019.00388.

[8] Singh, A., Sharma, S., Singh, J. (2021). Nature-inspired algorithms for Wireless Sensor Networks: A comprehensive survey. Computer Science Review, 39. doi: 10.1016/j.cosrev.2020.100342.

[9] Mohammed, O. S., Sewisy, A. A. A. M., Taloba, A. I. (2020). Solving Optimization Problems using Hybrid Metaheuristics: Genetic Algorithm and Black Hole Algorithm, in 2020 2nd International Conference on Computer and Information Sciences, ICCIS 2020. doi: 10.1109/ICCIS49240.2020.9257717.

[10] Nagar, R., Kumar, A., Singh, G. P., Kumar, S. (2015). Test case selection and prioritization using cuckoos search algorithm, in 2015 1st International Conference on Futuristic Trends in Computational Analysis and Knowledge Management, 283–288. doi: 10.1109/ABLAZE.2015.7155012.

[11] Bajaj, A., Abraham, A., Ratnoo, S., Gabralla, L. A. (2022). Test Case Prioritization, Selection, and Reduction Using Improved Quantum-Behaved Particle Swarm Optimization, Sensors, 22(12), 4374. doi: 10.3390/s22124374.

[12] Sarvaiya, J. & Singh, D. (2022). Selection of the optimal process parameters in friction stir welding/processing using particle swarm optimization algorithm, Mater Today Proc, doi: 10.1016/j.matpr.2022.04.062.

[13] Goldbarg, E. F., Marco, C., de Souz, G. R. (2008). Particle Swarm Optimization Algorithm for the Traveling Salesman Problem. in Traveling Salesman Problem. doi: 10.5772/5580.

[14] Jan, S. U. & Khan, H. U. (2021). Identity and Aggregate Signature-Based Authentication Protocol for IoD Deployment Military Drone, IEEE Access, 9, 130247–130263. doi: 10.1109/ACCESS.2021.3110804.

[15] Patni, M., Minera, S., Groh, R. M. J., Pirrera, A., Weaver, P. M. (2018). Three-dimensional stress analysis for laminated composite and sandwich structures, Compos B Eng, 155, 299–328. doi: 10.1016/j.compositesb.2018.08.127.

[16] Xing, Y., Wang, X., Shen, Q. (2021). Test case prioritization based on Artificial Fish School Algorithm, Comput Commun, 180, 295–302. doi: 10.1016/j.comcom.2021.09.014.

[17] Kumar, S., Datta, D., Singh, S. K., Azar, A. T., Vaidyanathan, S. (2015). Black hole algorithm and its applications, Studies in Computational Intelligence, 575, 147–170. doi: 10.1007/978-3-319-11017-2_7.

[18] Su, W., Li, Z., Wang, Z., Yang, D. (2020). A Meta-Heuristic Test Case Prioritization Method Based on Hybrid Model, in Proceedings - 2020 International Conference on Computer Engineering and Application, ICCEA 2020, 430–435. doi: 10.1109/ICCEA50009.2020.00099.

[19] Chaudhary, S. & Jatain, A. (2020). Performance Evaluation of Clustering Techniques in Test Case Prioritization, International Conference on Computational Performance Evaluation (ComPE) North-Eastern Hill University, Shillong, Meghalaya, India.

[20] Srivatsava, P. R., Mallikarjun, B., Yang, X. S. (2013). Optimal test sequence generation using firefly algorithm, Swarm Evol Comput, 8, 44–53. doi: 10.1016/j.swevo.2012.08.003.

[21] Mohapatra, D. P., Sahoo, R. K., Patra, M. R. (2016). A Firefly Algorithm Based Approach for Automated Generation and Optimization of Test Cases, International Journal of Computer Sciences and Engineering International Journal of Computer Sciences and Engineering, [Online]. Available: www.ijcseonline.org.

[22] Khatibsyarbini, M., Isa, M. A., Jawawi, D. M. A., Hamed, H. N. A., Mohamed Suffian, M. D. (2019). Test Case Prioritization Using Firefly Algorithm for Software Testing, IEEE Access, 7, 132360–132373, doi: 10.1109/ACCESS.2019.2940620.

[23] Vaz, R., Shah, V., Sawhney, A., Deolekar, R. (2017). Automated Big-O analysis of algorithms, in 2017 International Conference on Nascent Technologies in Engineering. doi: 10.1109/ICNTE.2017.7947882.

[24] Yang, Z. S. (2013). Multiobjective firefly algorithm for continuous optimization, Eng Comput, 29(2), 175–184. doi: 10.1007/s00366-012-0254-1.