Efficient Test Case Prioritization Using the Rat Swarm Optimizer Algorithm

Muhammad Afiq Bin Ariffin
Department of Software Engineering, University Tun Hussein Onn Malaysia
Batu Pahat, Johor, Malaysia
gi210018@student.uthm.edu.my

Mazidah Mat Rejab
Department of Software Engineering, University Tun Hussein Onn Malaysia
Batu Pahat, Johor, Malaysia
mazidah@uthm.edu.my

Rosziati Ibrahim
Department of Software Engineering, University Tun Hussein Onn Malaysia
Batu Pahat, Johor, Malaysia
rosziati@uthm.edu.my

Shahdatunnaim Binti Azmi
Department of Software Engineering, University Tun Hussein Onn Malaysia
Batu Pahat, Johor, Malaysia
shahdatun@uthm.edu.my

Nor Amalina Mohd Sabri
Department of Software Engineering, University Tun Hussein Onn Malaysia
Batu Pahat, Johor, Malaysia
noramalina@uthm.edu.my

Nur Afiqah Sahadun
Department of Software Engineering, University Tun Hussein Onn Malaysia
Batu Pahat, Johor, Malaysia
nurafiqah@uthm.edu.my

Shelena Soosay Nathan
ICT for Technology Humanization Focus Group, Centter for Diploma Studies, University Tun Hussein Onn Malaysia
Johor, Malaysia
shelena@uthm.edu.my

2024 IEEE 22nd Student Conference on Research and Development (SCOReD) | 979-8-3315-1007-7/24/$31.00 ©2024 IEEE | DOI: 10.1109/SCOReD64708.2024.10872660

***Abstract*— Software testing is a critical phase in software development to ensure the reliability and quality of software systems. With the increasing complexity of software applications, the number of test cases can grow significantly, posing challenges in terms of execution time and resource utilization. Test Case Prioritization (TCP) has emerged as a crucial technique to optimize the order of test case execution, aiming to detect defects early in the testing process. This paper presents a novel approach to Test Case Prioritization by harnessing the power of Rat Swarm Optimization (RSO), an algorithm inspired by the collective behavior of rat colonies. Experimental results demonstrate the efficacy of our approach compared to another algorithm. This study evaluates the performance of the proposed method using the execution time taken and compared it with the others algorithm like Ant Colony Optimization (ACO), Artificial Bee Colony (ABC) algorithm, and Firefly algorithm. The results indicate that the Rat Swarm Optimizer (RSO) algorithm is notably more efficient in terms of execution time compared to the other tested algorithms. The results highlight the potential of RSO for applications requiring rapid test case prioritization, where minimizing execution time is critical. This paper underscores the potential of RSO as a viable and innovative solution for Test Case Prioritization, offering software developers and testers a valuable tool for enhancing the efficiency and effectiveness of software testing processes.**

Keywords—Rat Swarm Optimizer (RSO), Test Case Prioritization (TCP), Software Testing

I. INTRODUCTION

Software regression testing is a crucial process in ensuring the integrity of modified code, especially as software systems grow increasingly complex. Test case prioritization (TCP) is a technique used to reorder test cases such that more important test cases, typically those with higher fault detection capabilities, are executed earlier. This process aims to detect faults as early as possible, minimizing the time and resources required during the testing phase. However, the challenge remains to efficiently prioritize test cases within reasonable timeframes, especially in large software projects where the number of test cases can be overwhelming [1].

Numerous metaheuristic algorithms have been applied to tackle the TCP problem, including Genetic Algorithms (GA) [2], Particle Swarm Optimization (PSO) [3], [11] and Ant Colony Optimization (ACO) [4]. Despite their success, these algorithms often face limitations in scalability and convergence speed. Swarm intelligence-based algorithms, such as the Rat Swarm Optimizer (RSO), simulate collective animal behavior and have shown promise in solving optimization problems. By mimicking rats' foraging behavior, RSO balances exploration (searching new solutions) and exploitation (refining known solutions) [5]. However, its application in TCP remains relatively unexplored, which this study aims to address. One such algorithm, the Rat Swarm Optimizer (RSO), inspired by the foraging behaviour of rats, offers a novel approach to TCP. Its adaptive and dynamic nature allows it to search for an optimal solution in both local and global spaces, potentially improving fault detection rates while maintaining computational efficiency.

The Rat Swarm Optimizer (RSO), a swarm-based metaheuristic algorithm, has recently emerged as a promising technique for solving optimization problems. RSO mimics the collective behaviour of rats, combining elements of exploration and exploitation to search for optimal solutions in complex spaces [6]. However, its application to TCP remains relatively underexplored, and this study aims to evaluate its effectiveness in optimizing test case prioritization. The adaptive and flexible nature of RSO makes it particularly suitable for problems like TCP, where the goal is to maximize fault detection while minimizing computational overhead .

In addition to addressing execution time, this study integrates fault detection rate (FDR) and Average Percentage of Faults Detected (APFD) as metrics for

evaluating the performance of the RSO. Previous studies on TCP often neglected these crucial metrics, focusing solely on execution time, which limits the scope of their findings. By incorporating multiple metrics, this study provides a more comprehensive evaluation of the RSO algorithm's effectiveness in TCP.

This study aims to achieve the following objectives:

- Objective 1: Apply the Rat Swarm Optimizer (RSO) to the problem of Test Case Prioritization and assess its performance.
- Objective 2: Compare the RSO algorithm's effectiveness in detecting faults with other metaheuristic algorithms, such as ACO, ABC [7], and Firefly Algorithm [8].
- Objective 3: Measure multiple performance metrics, including Execution Time, Fault Detection Rate (FDR), and Average Percentage of Fault Detection.

RSO's unique capabilities make it an intriguing candidate for test case prioritization in regression testing. Compared to other metaheuristic algorithms, RSO offers an enhanced balance between exploration and exploitation, improving its potential to detect faults earlier during test case execution. The advantages of RSO, such as its flexibility in adapting to varying problem complexities and its scalability across large datasets, make it particularly suited for practical applications in software engineering. This study contributes to the growing body of research by demonstrating how RSO can be effectively used in the TCP domain, providing significant improvements over traditional methods.

This study introduces a novel application of the Rat Swarm Optimizer (RSO) algorithm to the domain of test case prioritization (TCP), marking one of the first known efforts to leverage RSO's dynamic exploration and exploitation capabilities for this purpose. The approach optimizes test case sequences to enhance fault detection and computational efficiency. A comprehensive evaluation of RSO is conducted using key performance metrics, including execution time, Fault Detection Rate (FDR), and Average Percentage of Faults Detected (APFD). These metrics provide a holistic assessment of RSO's effectiveness in addressing the challenges of TCP. Furthermore, the study benchmarks RSO against three established metaheuristic algorithms, which is Ant Colony Optimization (ACO), Artificial Bee Colony (ABC), and Firefly Algorithm, demonstrating its comparative advantages in computational efficiency and fault detection capabilities. By utilizing a dataset with 2,000 test cases, the research highlights the scalability of RSO and underscores its practical relevance for real-world software testing scenarios, offering significant insights into its application in regression testing.

Finally, the paper makes a significant contribution to the understanding of how swarm intelligence-based approaches like RSO can be tailored and enhanced for the complex requirements of software testing. The results presented here offer valuable insights for both researchers and practitioners in software engineering, particularly those involved in automated testing, test case prioritization, and optimization algorithms.

II. Literature Review

Test Case Prioritization (TCP) has attracted significant research attention, with various metaheuristic algorithms being explored to address the complexities of this optimization problem. The Rat Swarm Optimizer (RSO) algorithm [9] offers a bio-inspired approach, based on the natural behaviours of rats, particularly their foraging and hunting techniques. compared RSO with multiple well-known optimization algorithms, including Spotted Hyena Optimizer (SHO), Grey Wolf Optimization (GWO), Particle Swarm Optimization (PSO), and Genetic Algorithm (GA) across a series of real-world engineering problems such as pressure vessel design and speed reducer optimization. The results showed RSO's superior performance in solving high-dimensional and combinatorial optimization problems, outperforming other algorithms in terms of solution quality and convergence speed.

Further enhancements to RSO have been explored in recent works. In 2022 [13], a study has introduced an improved variant of RSO, termed as Multiple Adaptive Rat Swarm Optimizer (MARSO), which incorporates Adaptive Learning Exemplar (ALE) and Adaptive Population Size (APS) strategies. This modified version of RSO was tested on the IEEE CEC2017 benchmark functions and compared with several algorithms, including Differential Evolution (DE) and Lion Swarm Optimization (LSO). The experimental results demonstrated MARSO's ability to achieve higher convergence precision and improved performance in both exploration and exploitation phases, making it more suitable for complex optimization problems.

RSO has also been applied in diverse fields, such as data clustering [14]. The authors evaluated RSO’s effectiveness in clustering tasks and compared it with other algorithms like PSO, GA, and hybrid approaches such as the Multi-Verse Optimizer (MVO) and Harris Hawk Optimization (H-HHO). Their results showed that the Rat Swarm Optimizer for Clustering (RSOC) performed exceptionally well in terms of homogeneity, purity, and v-measure across several datasets. However, the study also highlighted some weaknesses of RSOC, particularly in datasets where the algorithm struggled with convergence.

These studies suggest that while RSO and its variants demonstrate significant potential in solving optimization problems, further research is needed to improve its performance, particularly in the context of test case prioritization. The inclusion of adaptive mechanisms and hybrid strategies may help enhance its capability in balancing exploration and exploitation during optimization tasks.

III. Methodology

A. Rat Swarm Optimizer (RSO)

The Rat Swarm Optimizer (RSO) is a swarm-based metaheuristic algorithm inspired by the foraging behavior of rats, emphasizing a balance between exploration and exploitation within the search space [9]. This dual capability enables RSO to navigate efficiently through complex

optimization problems like Test Case Prioritization (TCP), where test cases need to be reordered to maximize fault detection and minimize execution time.

RSO operates by simulating the collective intelligence of rats, dynamically adjusting exploration and exploitation rates. This adaptability reduces the risk of premature convergence, enhancing its potential to find optimal solutions in diverse and large-scale TCP scenarios.

RSO Algorithm Parameters

- Population Size: 10 rats, each representing a unique test case sequence.
- Maximum Iterations: 100, balancing computational cost with solution quality.
- Exploration Probability: Fixed at 0.3 to maintain simplicity and focus on RSO's original design.

RSO Workflow

The algorithm iteratively updates test case sequences by evaluating their fitness, favoring sequences that maximize fault detection early. The pseudocode in Fig. 1 below illustrates the RSO implementation:

```
1. Initialize:
   a. Define population size (num_rats) and maximum iterations (max_iterations).
   b. Generate a random initial sequence of test cases for each rat.
   c. Set exploration probability (exploration_prob).

2. Evaluate Initial Fitness:
   a. For each rat, calculate fitness using:
      Fitness = (Requirement Priority × Function Points) / (Complexity × Time × Cost).
   b. Store the best fitness value and corresponding test case sequence.

3. Main Loop (for each iteration):
   a. For each rat in the population:
      i. Update test case sequence based on exploration and exploitation.
      ii. Recalculate fitness for the new sequence.
      iii. Update the rat's position if the new sequence improves fitness.

   b. Adjust exploration probability dynamically, if applicable.

   c. Update global best fitness and sequence if a better solution is found.

4. Convergence Check:
   a. Stop if maximum iterations are reached or convergence criteria are met.

5. Output:
   a. Best prioritized test case sequence.
   b. Metrics: Fault Detection Rate (FDR), Average Percentage of Faults Detected (APFD), and Execution Time.
```

Fig 1. Pseudocode of the RSO

B. Test Case Prioritization

Test Case Prioritization (TCP) plays a pivotal role in regression testing, aiming to reorder test cases in a manner that maximizes the early detection of faults. This proactive approach ensures that faults are identified and addressed during the initial stages of testing, significantly reducing the time, effort, and costs associated with debugging and retesting. By prioritizing high-impact test cases early in the execution cycle, TCP enhances the overall efficiency of the software testing process while maintaining the quality and reliability of the system under test [12], [17].

Effective TCP is particularly vital in large-scale software projects, where the sheer number of test cases can be overwhelming. Executing test cases sequentially without a prioritization strategy can result in delays, increased resource utilization, and potentially missed faults during critical testing phases. Consequently, leveraging optimization techniques to address the TCP problem is essential for modern software development practices.

In this study, the Rat Swarm Optimizer (RSO) algorithm is applied to tackle the TCP problem, leveraging its dynamic balance between exploration (searching new test case sequences) and exploitation (refining high-quality sequences). RSO's adaptability makes it particularly suitable for prioritizing test cases in complex and diverse datasets, addressing the dual objectives of enhancing fault detection and minimizing execution time.

The algorithm evaluates test case sequences using a fitness function that integrates multiple attributes critical to effective prioritization:

- Requirement Priority (R_Priority): Represents the criticality of the business requirement associated with the test case. Higher priorities ensure that critical functionality is tested earlier.
- Function Points (FP): Quantifies the functional complexity and size of the test case, directly influencing its testing scope and impact.
- Complexity: Reflects the difficulty level of testing the associated module, with more complex test cases requiring greater effort and resources.
- Time: Indicates the estimated maximum time allocated for executing the test case, as determined by experienced QA leads or analysts.
- Cost: Represents the financial impact of testing, calculated based on the complexity and time attributes.

The fitness function, formulated as follows, ensures that test cases with higher fault detection potential are prioritized while balancing resource constraints:

$$\text{Fitness} = \frac{\text{Requirement Priority (R\_Priority)} \times \text{Function Points (FP)}}{\text{Complexity} \times \text{Time} \times \text{Cost}} \tag{1}$$

This mathematical representation provides a robust framework for evaluating the effectiveness of test case sequences. Sequences that detect faults earlier and efficiently utilize resources are assigned higher fitness values, directly contributing to an improved Fault Detection Rate (FDR) and a higher Average Percentage of Faults Detected (APFD).

By employing RSO for TCP, this study demonstrates how intelligent optimization techniques can effectively address the challenges of large-scale software testing, providing a structured approach to maximize testing efficiency and effectiveness.

C. RSO Implementation

The Rat Swarm Optimizer (RSO) algorithm was implemented to prioritize test cases in regression testing by optimizing their sequence based on fault detection and execution efficiency. A population of 10 rats was initialized, with each rat representing a unique sequence of 2,000 test cases. The algorithm iterates through 100 cycles,

dynamically updating sequences to identify the optimal prioritization.

Fitness evaluation was performed using a defined function incorporating Requirement Priority, Function Points, Complexity, Time, and Cost. The formula ensures that sequences detecting faults earlier are prioritized, contributing to a higher Fault Detection Rate (FDR) and better Average Percentage of Faults Detected (APFD).

Parameter selection for RSO, such as a fixed exploration probability of 0.3 and a population size of 10 rats was based on prior research and tuned for the current dataset's complexity. The stopping criterion was set at 100 iterations, ensuring convergence without excessive computational cost.

The dataset used in this study was gathered from a software company working on a comprehensive software package for a car financing and leasing company. This dataset was specifically designed to address the scarcity of publicly available data in the domain of software testing, particularly for research in test case prioritization (TCP). It includes detailed attributes tailored for TCP, enabling analysis and optimization through machine learning and metaheuristic techniques.

The dataset consists of 2,000 rows and six columns, formatted in a .csv file. Each row represents a test case, while the columns contain attributes critical to test case prioritization:

- B_Req (Business Requirement): Identifies the specific business requirement linked to each test case.
- R_Priority (Requirement Priority): Reflects the priority assigned to the business requirement, which plays a crucial role in determining the test case's importance.
- FP (Function Points): Represents the functional complexity of each testing task, encapsulating the scope and size of associated test cases.
- Complexity: Indicates the complexity level of the corresponding function point or related modules. This attribute impacts the testing effort required and is determined based on predefined criteria included in an attached reference document.
- Time: Represents the estimated maximum time allocated for testing each function point, as determined by experienced QA team leads or senior SQA analysts.
- Cost: Calculated based on complexity and time using the function point estimation technique, adhering to the following formula:

$$Cost = (Complexity \times Time) \times Average\ Amount\ per\ Task \tag{2}$$

Pre-processing steps ensured that the dataset was ready for use with the optimization algorithms. These steps included removing missing data, normalizing values, and encoding categorical variables where necessary. Each attribute significantly influences the algorithm's decision-making process during test case prioritization.

In this study, the dataset serves as the foundation for evaluating the performance of the RSO algorithm in prioritizing test cases, offering a practical context for assessing fault detection rate (FDR), Average Percentage of Faults Detected (APFD), and execution time. Although practical testing environments may vary in the number of test cases executed per iteration, the consistent use of a comprehensive dataset in this study ensures a standardized evaluation framework, providing a robust basis for assessing the RSO algorithm's effectiveness in test case prioritization.

D. Evaluation Metrics

The effectiveness of the RSO algorithm is evaluated using three key metrics:

I. Execution Time: The total time taken to execute the test cases in a given sequence. This metric reflects the efficiency of the algorithm in handling large datasets.

II. Fault Detection Rate (FDR): The ratio of faults detected to the total number of faults, calculated as:

$$FDR = \frac{Faults\ Detected}{Total\ Faults}\ X\ 100 \tag{3}$$

This metric is critical for assessing the quality of the prioritization.

III. Average Percentage of Faults Detected (APFD): A measure used to evaluate how quickly faults are detected within the test case sequence. APFD is calculated as:

$$\text{APFD} = 1 - \frac{TF_1 + TF_2 + \cdots + TF_m}{nm} + \frac{1}{2n} \tag{4}$$

where TF_i is the position of the i^{th} fault-detecting test case, n is the total number of test cases, and m is the total number of faults.

E. Comparison Algorithms

To benchmark the performance of the RSO algorithm, three other metaheuristic algorithms were implemented for comparison:

Ant Colony Optimization (ACO): ACO [10] simulates the pheromone-laying behaviour of ants to find optimal paths in search spaces. It has been widely applied to TCP due to its ability to handle discrete optimization problems efficiently .

Firefly Algorithm (FA): The Firefly Algorithm is a nature-inspired algorithm that simulates the attraction between fireflies to guide the search process. Its use in TCP is motivated by its balance of exploration and exploitation.

Artificial Bee Colony (ABC): ABC is based on the foraging behaviour of honeybees, where employed bees, onlooker bees, and scout bees collaborate to find optimal solutions. Its dynamic search mechanism is useful in prioritizing test cases with varying fault detection capabilities.

The RSO algorithm is compared against these algorithms in terms of execution time, FDR, and APFD to assess its overall performance in test case prioritization.

IV. Results

The results of this study are evaluated using three performance metrics: Execution Time, Fault Detection Rate (FDR), and Average Percentage of Faults Detected (APFD). These metrics were chosen to comprehensively assess the effectiveness of the Rat Swarm Optimizer (RSO) algorithm in solving the Test Case Prioritization (TCP) problem. For comparison, the Ant Colony Optimization (ACO), Artificial Bee Colony (ABC), and Firefly Algorithm were implemented under the same experimental conditions.

A. Execution Time

Execution time is a critical factor in regression testing, where efficiency determines the feasibility of an algorithm in real-world applications. Table 1 and Fig. 2 presents the execution time for all algorithms. Among them, RSO demonstrated the fastest execution time at 4033.57 seconds, followed closely by ACO at 4254.37 seconds. ABC and Firefly were slower, with execution times of 5601.51 seconds and 5964.31 seconds, respectively.

This performance demonstrates RSO's computational efficiency, which can be attributed to its adaptive exploration-exploitation mechanism. Unlike ABC and Firefly, which rely on more exploratory search processes, RSO effectively balances exploration and exploitation to reduce computational overhead.

TABLE I. EXECUTION TIME RESULTS

Algorithm	Execution Time (s)
RSO	4033.57
ACO	4254.37
ABC	5601.51
FA	5964.31

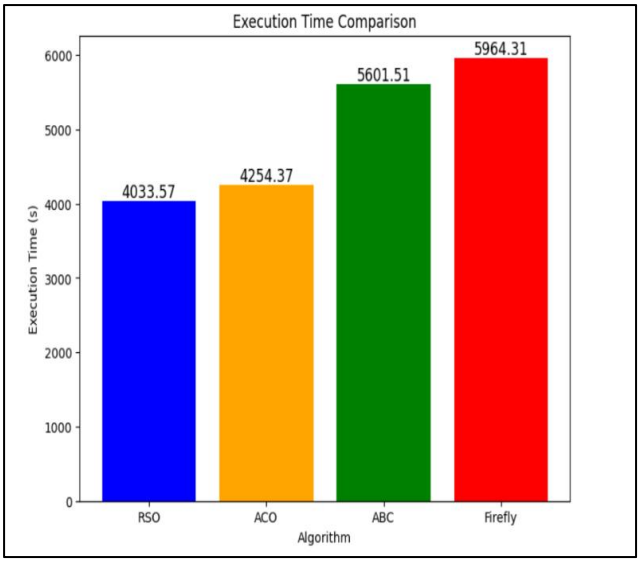

Fig 2. Execution Time Results

B. Fault Detection Rate (FDR)

Fault Detection Rate (FDR) measures the percentage of detected faults relative to the total number of faults in the dataset. All algorithms achieved an FDR of 100%, indicating that the prioritization strategies were effective in covering all faults within the test case sequences (Table 2). This uniformity underscores that RSO, ACO, ABC, and Firefly algorithms can ensure comprehensive fault coverage, even in large datasets.

While FDR reflects complete fault detection, it does not measure how early faults are detected, which is where APFD becomes significant

C. Average Percentage of Faults Detected (APFD)

APFD evaluates the rate at which faults are detected during the execution sequence. As shown in Fig. 3, ACO and ABC achieved the highest APFD scores of 100%, while Firefly and RSO scored lower at 0.13% and 0.09%, respectively.

The lower APFD score for RSO suggests that while it ensures complete fault detection, it prioritizes fault detection later in the sequence. This result highlights a potential trade-off in RSO's optimization process: its focus on overall fault detection rather than early prioritization. Enhancing RSO's ability to detect critical faults earlier could further improve its applicability in time-sensitive scenarios.

TABLE II. EXPERIMENT DATA RESULTS

Algorithm	Best APFD	FDR
RSO	0.09%	100%
ACO	100%	100%
ABC	100%	100%
Firefly	0.13%	100%

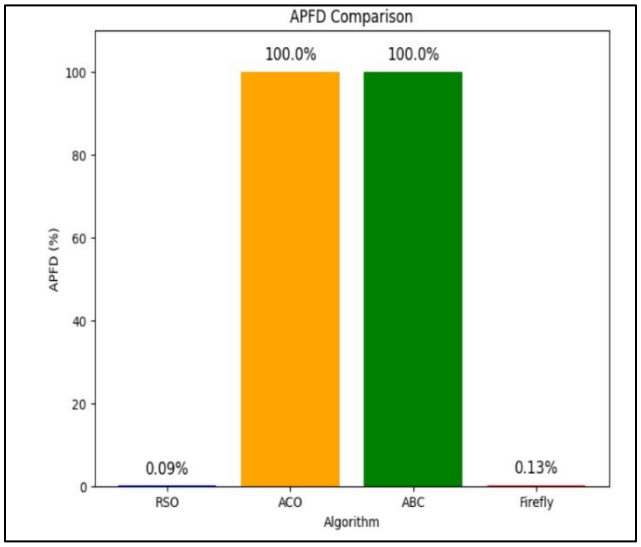

Fig 3. Average Percentage of Faults Detected comparison

The results indicate that RSO performs exceptionally well in terms of execution time and fault detection rate, making it a suitable choice for large-scale regression testing. Its performance in APFD, however, highlights areas for improvement. The adaptive exploration and exploitation mechanism of RSO contributed to its faster execution time, but it may require adjustments to prioritize earlier fault detection.

Compared to ACO, ABC, and Firefly, RSO demonstrates a balanced trade-off between computational efficiency and comprehensive fault coverage. While ACO and ABC excel in early fault detection (APFD), their longer execution times make them less practical for large datasets. Firefly, though comparable to ABC in terms of APFD, is the slowest algorithm, limiting its applicability.

V. Discussion and Conclusions

The experimental results demonstrate that the Rat Swarm Optimizer (RSO) is a promising approach for addressing the Test Case Prioritization (TCP) problem. RSO outperformed the comparison algorithms in terms of execution time, achieving the shortest time among the four tested algorithms. This highlights its computational efficiency, which is critical for large-scale regression testing scenarios.

RSO's ability to achieve a fault detection rate (FDR) of 100% underscores its effectiveness in covering all faults within the dataset. However, its performance in terms of the Average Percentage of Faults Detected (APFD) reveals an area for improvement. With an APFD score of 0.09%, RSO tends to prioritize fault detection later in the sequence. This trade-off between execution time and early fault detection reflects the algorithm's focus on overall optimization rather than prioritizing critical faults at the start of the sequence.

Compared to the Ant Colony Optimization (ACO) and Artificial Bee Colony (ABC) algorithms, RSO demonstrated superior execution time but lagged behind in APFD. The Firefly Algorithm exhibited similar characteristics to ABC but required significantly more computational time, making it less practical for large datasets. These results suggest that while RSO excels in computational efficiency and fault coverage, further refinements are needed to enhance its prioritization capabilities for earlier fault detection.

One of the key strengths of RSO lies in its adaptability, allowing it to balance exploration and exploitation dynamically [15], [16]. This feature not only contributed to its efficiency but also demonstrated its scalability when applied to a dataset of 2,000 test cases. However, the relatively lower APFD score points to a potential limitation in RSO's current design, which could be addressed by incorporating mechanisms to prioritize critical faults earlier in the sequence.

This study has successfully demonstrated the application of the Rat Swarm Optimizer (RSO) algorithm to the Test Case Prioritization (TCP) problem. RSO was shown to be highly efficient in terms of execution time and fault detection coverage, making it a viable option for large-scale regression testing. The comparative analysis with ACO, ABC, and Firefly Algorithm further highlights RSO's advantages, particularly in computational efficiency.

However, the findings also reveal a trade-off in RSO's prioritization strategy, as evidenced by its lower APFD score. This limitation underscores the need for further research to refine RSO's capabilities in early fault detection. Enhancements such as adjusting the fitness function to prioritize faults detected earlier in the sequence or incorporating additional optimization mechanisms could improve its performance in this area.

Future research will focus on embedding the Rat Swarm Optimizer (RSO) algorithm with other metaheuristic or machine learning-based approaches to enhance its optimization capabilities further. While RSO has demonstrated promising results in test case prioritization, integrating it with algorithms such as Particle Swarm Optimization (PSO) for improved exploration, or Genetic Algorithms (GA) for robust solution diversity, could address limitations like premature convergence and scalability in larger datasets. This hybridization could leverage the strengths of each technique, such as PSO's velocity-based updates or GA's crossover and mutation mechanisms, to create a more adaptive and efficient prioritization framework.

This research contributes to the growing body of knowledge on swarm intelligence-based optimization techniques and their application to software testing. By demonstrating RSO's potential for TCP, this study lays the groundwork for further advancements in automated testing and optimization algorithms, paving the way for more efficient and effective software quality assurance practices.

Acknowledgment

Communication of this research is made possible through monetary assistance by University Tun Hussein Onn Malaysia and the UTHM Publisher's Office via Publication Fund E15216.

References

[1] S. Yoo, and M. Harman. "Regression testing minimization, selection and prioritization: a survey." Software testing, verification and reliability 22, no. 2 (2012): 67-120.

[2] A. Bajaj, and O. P. Sangwan. "A systematic literature review of test case prioritization using genetic algorithms." Ieee Access 7 (2019): 126355-126375.

[3] M. Tyagi, and S. Malhotra. "Test case prioritization using multi objective particle swarm optimizer." In 2014 International Conference on Signal Propagation and Computer Technology (ICSPCT 2014), pp. 390-395. IEEE, 2014.

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

[5] S. Sengupta, S. Basak, and R. A. Peters. "Particle Swarm Optimization: A survey of historical and recent developments with hybridization perspectives." *Machine Learning and Knowledge Extraction* 1, no. 1 (2018): 157-191.

[6] S. Guodong, H. Mingmao, L. Yanfei, F. Jian, G. Aihong, and G. Qingshan. "A multi-strategy fusion-based Rat Swarm Optimization algorithm." Soft Computing (2024): 1-19.

[7] P. Padmnav, G. Pahwa, D. Singh, and S. Bansal. "Test case prioritization based on historical failure patterns using ABC and GA." 2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence). IEEE, 2019.

[8] M. Khatibsyarbini, M. A. Isa, D. N. A. Jawawi, H. N. A. Hamed, and M. D. M. Suffian. "Test case prioritization using firefly

algorithm for software testing." IEEE access 7 (2019): 132360-132373.

[9] G. Dhiman, M. Garg, A. Nagar, V. Kumar, and M. Dehghani. "A novel algorithm for global optimization: rat swarm optimizer." Journal of Ambient Intelligence and Humanized Computing 12 (2021): 8457-8482.

[10] S. F. Ahmad, D. K. Singh, and P. Suman. "Prioritization for regression testing using ant colony optimization based on test factors." In *Intelligent Communication, Control and Devices: Proceedings of ICICCD 2017*, pp. 1353-1360. Springer Singapore, 2018.

[11] A. Bajaj, A. Abraham, S. Ratnoo, and L. A. Gabralla. "Test case prioritization, selection, and reduction using improved quantum-behaved particle swarm optimization." Sensors 22, no. 12 (2022): 4374.

[12] S. Elbaum, A. G. Malishevsky, and G Rothermel. "Test case prioritization: A family of empirical studies". Software Engineering, IEEE Transactions on, 2002, 28(2):159-182.

[13] Z. Xu, X. Liang, M. He, and H. Chen. "Multiple adaptive strategies-based rat swarm optimizer." In 2021 IEEE 7th International Conference on Cloud Computing and Intelligent Systems (CCIS), pp. 159-163. IEEE, 2021.

[14] I. Zebiri, D. Zeghida, and M. Redjimi. "Rat swarm optimizer for data clustering." Jordanian Journal of Computers and Information Technology 8, no. 3 (2022).

[15] A. Toolabi Moghadam, M. Aghahadi, M. Eslami, S. Rashidi, B. Arandian, and S. Nikolovski. "Adaptive rat swarm optimization for optimum tuning of SVC and PSS in a power system." International Transactions on Electrical Energy Systems 2022, no. 1 (2022): 4798029.

[16] T. Lou, G. Guan, Z. Yue, Y. Wang, and S. Tong. "A Hybrid K-means Method based on Modified Rat Swarm Optimization Algorithm for Data Clustering." (2024).

[17] D. Paterson, J. Campos, R. Abreu, G.M. Kapfhammer, G. Fraser, and P. McMinn. " An Empirical Study on the Use of Defect Prediction for Test Case Prioritization." 2019 12th IEEE Conference on Software Testing, Validation and Verification (ICST).