Article

Test Case Prioritization, Selection, and Reduction Using Improved Quantum-behaved Particle Swarm Optimization

Anu Bajaj [1]*, Ajith Abraham [1], Saroj Ratnoo [2] and Lubna Abdelkareim Gabralla [3]

[1] Machine Intelligence Research Labs (MIR Labs), Auburn, WA, USA
[2] Department of Computer Science and Engineering, Guru Jambheshwar University of Science and Technology, Hisar, Haryana, India
[3] Department of Computer Science and Information Technology, College of Applied, Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia
* Correspondence: anu.bajaj@mirlabs.org

Abstract: Emerging area of IoT and sensor networks brings lots of software applications on a daily basis. To keep up with the ever-changing expectations of clients and the competitive market, the software must be updated. The changes may cause unintended consequences, necessitating retesting, i.e., regression testing, before being released out. The efficiency and efficacy of regression testing techniques can be improved with the use of optimization approaches. This paper proposes an improved quantum behave particle swarm optimization approach for regression testing. The algorithm is improved by employing a fix-up mechanism to perform perturbation for combinatorial TCP problem. Second, the dynamic contraction-expansion coefficient is used to accelerate the convergence. It is followed by an adaptive test case selection strategy to choose the modification-revealing test cases. Finally, the superfluous test cases are removed. Furthermore, the algorithm's robustness is analyzed for fault as well as statement coverage. The empirical results reveal that the proposed algorithm performs better than the genetic algorithm, bat algorithm, grey wolf optimization, particle swarm optimization and its variants for prioritizing test cases. The findings show that inclusivity, test selection percentage and cost reduction percentages are higher in case of fault coverage compared to statement coverage but at the cost of high fault detection loss (approx. 7%) at test case reduction stage.

Keywords: regression testing; nature-inspired algorithms; test case prioritization; test case reduction; test case selection; particle swarm optimization; QPSO

Citation: Bajaj, A.; Abraham, A.; Ratnoo S. and Gabralla, L.A. Test Case Prioritization, Selection and Minimization Using Improved Quantum-behaved Particle Swarm Optimization. *Sensors* **2022**, *1*, 0. https://doi.org/

Received:
Accepted:
Published:

1. Introduction

With the advent of healthcare applications and tremendous amount of information processing there is a need for fault handling [1]. Therefore, software testing is becoming essential for safety critical systems, e.g., IoT devices and the sensor networks are connected with it in one or other way and failure may lead to loss of money and life. In other words, it is an important part of the software development lifecycle since it ensures that the software is of high quality. It accounts for around half of the entire cost [2]. Testing during the evolution and maintenance phases becomes more important to assure the software's dependability. All of the test cases must be re-implemented to guarantee that the quality is not affected; this is known as regression testing [3]. In other words, software is continually changing to sustain in a competitive market by updating and maintaining to satisfy changing needs. Complete retesting accounts for around eighty percent of the entire maintenance cost [4]. On the other hand, it is difficult to test each upgraded version of software nowadays. Software becomes more complex with frequent upgrades, the amount of time and effort required for regression testing may increase. Test case reduction, selection, and priority strategies can help solve these bottleneck problems [5].

1. *Test Case Prioritization (TCP)*
   It ranks the test cases based on some predefined goals, such as maximum code coverage, fault coverage, and requirements coverage. It finds $T_i \in PT$ such that $(\forall T_j)\ (T_j \in PT)\ (T_j \neq T_i)\ [f(T_i) \geq f(T_j)]$; for a given test suite, T, its permutations set PT, and f denotes a function from PT to real numbers [2]. In other words, it claims to identify T_i from PT with a value of $f(T_i)$ larger than any other test case (T_j) in PT. The coverage rate is represented by f that calculates the performance of the permutation series in terms of real number.
2. *Test Case Selection (TCS)*
   It chooses essential test cases that are linked with the update of the software. In other words, it finds a subset of T, T_i for testing the modified version of P, P_i. [5].
3. *Test Case Reduction (TCR)*
   It focuses on removing redundant test cases by finding a representative test cases set, T_i, from T that satisfies test requirements set $R\ r1, \ldots, rn$, for the defined coverage of the program [2].

TCP is the most commonly used out of these three strategies by researchers because it doesn't eliminate or pick test cases. Instead, it simply rearranges them such that the most critical ones are checked first. The significance of these test cases is determined by a number of factors. It might be code coverage, fault coverage, requirement priority, or critical components [3]. TCS and TCR, on the other hand, may leave out certain crucial test cases that can be useful for upcoming versions of the product [2]. On the other side, finding the best order for the test cases, as well as the best way to limit or choose the test cases makes it NP-hard problem [6].

Optimization strategies can be used to successfully overcome these issues. The nature-inspired algorithms have been successfully employed to solve difficult optimization problems in many domains [7]. Alternatively, they can improve the cost-effectiveness of regression testing. Nature-inspired algorithms appeal to the researchers because of their basic structure and ease of usage. The methods are theoretically built by modelling natural events [3]. These algorithms are broadly classified into three classes: biology-inspired, physics/chemistry inspired and social-phenomena inspired algorithms. These techniques have also been applied in regression testing [2]. The most often used algorithms are evolutionary algorithms and swarm intelligence-based algorithms from the biology-inspired family of nature-inspired approaches [8].

PSO algorithms have been used by researchers for solving regression testing problems. We have also used similar approaches in our previous works. For Example, Dragonfly was hybridized with PSO for prioritizing the test cases using fault coverage information. It reduced the test suite to 87-96% which thereby removed some of the critical test cases. Therefore, tri-level regression testing was performed by layering the test case selection in between the test case prioritization and reduction. The promising results of the nature-inspired algorithms on statement coverage motivated us to validate the results on fault coverage as well. As a result, this work analyses the effect of fault and statement coverage criteria on the performance of the technique. It also suggests a swarm-intelligence based algorithm, Quantum-behaved particle swarm optimization (QPSO) and its improved version, IQPSO for tri-level regression testing to improve the quality of results. The main contributions of this research are:

- Improved QPSO algorithm to solve the TCP for fault and statement coverage criteria.
- Extended algorithm for selecting the modification revealing test cases using historical information and further reduction of the test suite size.
- Performance analysis of the algorithms using different testing goals, i.e., code coverage and fault coverage.
- Verified robustness of the proposed algorithm against Genetic Algorithm (GA), Bat Algorithm (BAT), Grey Wolf Optimization (GWO), Particle Swarm Optimization (PSO), Adaptive PSO (AdPSO) and hybrid of PSO with Gravitational Search Algorithm (PSOGSA) and Dragonfly Algorithm (DAPSO).

Alternatively, the nature-inspired algorithms prioritize test cases based on the most extensively used criteria: statement and fault coverage. The modification revealing test cases are included with the help of adaptive test case selection method. It takes into account test case history and picks failed test cases based on probabilistic potentials. Since, test selection percentage is large at the expense of high inclusiveness. So, the TCR is introduced to minimize test suite size by removing duplicate test cases. The empirical results show that the proposed technique works for both fault and statement coverage. Inclusivity, test selection percentage and cost reduction percentages are higher in case of fault coverage compared to statement coverage but at the cost of slight high fault detection loss at test case reduction stage.

The organization of the paper is structured as follows: Section 2 describes the research work done in the application of PSO algorithms for solving regression testing problems. In succession, Section 3 presents the working mechanism of the basic PSO and QPSO. The proposed algorithms are discussed in the Section 4. Sections 5 and 6 present the experimental setup and results analysis. It is concluded in Section 7.

2. Literature Review

Review of literature in the applications of nature-inspired algorithms for regression testing is presented in this section. For Example, Li *et al.* [4] compared search-based methods to traditional algorithms. It was discovered that the search space is better explored with GA. It prompted more research into the use of nature-inspired algorithms. For example, Zhang *et al.* [9] used a distance-based and index-based implementation of ACO to prioritize test cases, and the results were superior to GA, PSO, and RS. Using the Cuckoo Search Algorithm (CSA), a new fixup mechanism for permutation encoding was developed to address the TCP problem [3]. CSA was also used to reduce the test suite for configuration-aware software testing [10].

Mohapatra and Prasad [11] have employed ant colony optimization on Java programs, and analyzed the performance for reduced suite and complexity to traditional techniques. The quantum-inspired ACO approach for test suite reduction was developed by Zhang et al. [12]. The suggested method outperformed previous ACOs in terms of % decrease in size. NSGA-II was employed by Mondal et al. [13] for TCS by taking the test suite variety and code coverage as a fitness metric. With a maximum time limitation of 20%, it was discovered that diversity enhanced the defect detection rate by up to 16 percent.

Several researchers have employed PSO, such as Khatibsyarbini *et al.* [14], who used string distances to arrange the test instances and validated on the real-world TSL dataset. To choose test cases based on redundancy, binary constraint PSO and its hybrid variants with local search algorithms were developed [15]. PSO was implemented with local search to select test cases having goals to increase branch coverage and lower costs [16]. Because of the positive findings, PSO was also combined with harmony search, which performed better than NSGA-II [17]. Correia [18] developed a test suite diversity diagnosability measure and results were improved by applying local search algorithms with PSO to maximise requirement coverage while lowering associated costs.

Test case reduction was also implemented with TCP by hybridizing the PSO with Dragonfly algorithm. Observations suggested that hybrid algorithms outperformed other search algorithms [6]. Tri level regression tesing was proposed to prioritize, select and minimize the test cases based on statement coverage. It was observed that hybrid of PSO with gravitational search algorithm (PSOGSA) outperformed GA, PSO, and GSA [5]. The test suite was minimized using hybrid PSO and firefly algorithm considering the fault coverage [19]. Modified condition decision coverage criteria was employed as fitness measures in PSO for prioritizing the test cases [20]. Deneke et al., [21] also proposed PSO algorithm for reducing the test suite based on requirement coverage and cost. Samad et al., [22] proposed multi-objective PSO for optimizing the code, fault coverage with cost. Table 1 briefs the application of PSO algorithms for regression testing techniques along with their optimization criteria.

Table 1. Summary of Nature-Inspired Algorithms used in regression Testing

Author (Year)	**Technique**	**Nature-Inspired Algorithms**	**Criteria**
De Souza et al., 2013, 2014	TCS	Binary PSO	Requirement Coverage with Time
De Souza et al., 2015	TCS	Binary PSO-HS	Branch Requirement Coverage with Cost
Khatibsyarbini et al., 2018	TCP	PSO	String Distances
Agrawal and Kaur 2018	TCS	PSO	Fault Coverage and Time
Correia, 2019	TCS	PSO-LS	Requirement Coverage
Nayak and Ray 2019	TCP	PSO	Modified Condition Decision Coverage
Samad et al., 2021	TCP	MOPSO	Code and Fault Coverage with Cost
Bajaj and Abraham, 2021	TCP and TCR	DAPSO	Fault Coverage
Bajaj and Sangwan, 2021	TCP, TCS, TCR	PSOGSA	Statement Coverage
Bharathi, 2022	TCR	PSO-FFA	Fault Coverage
Deneke et al., 2022	TCR	PSO	Requirement Coverage and cost

PSO algorithms become one of the state-of-the-art algorithms and show promising results in various domains, e.g., reduction of CO_2 emissions in air baggage systems [24]. The original PSO, on the other hand, had issues such as getting trapped in local optima and premature convergence [25]. According to the findings, upgraded and hybrid versions of PSO outperformed PSO for complicated systems [15]. One of such algorithm is Quantum-behaved PSO (QPSO). It is based on quantum mechanics in which particles can travel through a large search space for global convergence [26]. The method has shown good results in a variety of applications like cancer classification [27], feature extraction ([28]-[30]) and constrained engineering problems [31] and others [32]. However, it has not been investigated in the TCP domain, which might be due to the fact that it was originally designed for continuous problems. So, to shift infeasible solutions into feasible ones, we suggested a discrete QPSO method based on a adaptation strategy. It does, however, have significant drawbacks, such as early convergence. As a result, we have improved it with dynamic contraction-expansion coefficient to speed up the performance in the last iterations [31].

Besides this, we have extended the algorithm for selecting the modification-revealing (MR) test cases from the current best solution of TCP. It is occasionally necessary to reduce the test suite by reducing redundancy because of time limits, thus we employed the TCR approach in the end. Our key focus in this research is on performing regression testing in three steps, including TCP, TCS, and TCR process for fault and statement coverage. Alternatively, the effect of different testing criteria on the overall performance of the algorithms is analyzed. Observations suggest that the tri-level regression technique is effective for both coverage criteria. The proposed algorithm IQPSO is statistically not significant than PSOGSA, however, its variance and mean fitness values are better.

3. Preliminaries

This Section briefly explains the working mechanism of Particle Swarm Optimization (PSO) and Quantum behaved PSO.

3.1. Particle Swarm Optimization

PSO is inspired by particle behaviour such as flocking, swarming, and herding. Each particle changes its flight based on self or companion's previous flight experience. Each particle, based on its own experience, is aware of the location of food, which is referred to as personal best position (P). Simultaneously, the particle has knowledge of the swarm's best discovered position, global best position (G). This phenomenon is reproduced in order to solve real-world issues. In other words, the swarm is made up of particles that fly randomly in the solution space with velocity v_i at position x_i and change positions based on personal experience, social behaviour, and cognitive behaviour ([33]). The position and velocity of each particle i at t^{th} generation are defined mathematically as:

$$v_i(t+1) = wv_i + c_1 r_1(P_i(t) - x_i(t)) + c_2 r_2(G(t) - x_i(t)) \tag{1}$$

$$x_i(t+1) = x_i(t) + v_i(t+1) \tag{2}$$

w is the inertia weight used to regulate the impact of prior velocity; c_1 and c_2 are the constants used to adjust the attractiveness speeds among these social and cognitive elements; and r_1 and r_2 are uniform random values in the range $[0, 1]$.

3.2. Quantum Behaved PSO

A more robust variant of PSO called QPSO is created [25] as PSO cannot ensure global convergence [32]. It determines the quantum-behaved particles' route by assuming that N particles with $delta$ potential and specified energy are well-centered in each dimension of the n-dimensional Hilbert search space. The j^{th} component of the particle's position at t^{th} iteration is given by the Monte Carlo technique:

$$x_{ij}(t+1) = a_{ij}(t) \pm \frac{L_{ij}(t)}{2} \ln\left(\frac{1}{u_{ij}(t)}\right) \tag{3}$$

$$L_{ij}(t) = 2\theta|mean_j(t) - x_{ij}(t)| \; and \; mean_j(t) = \frac{1}{N}\sum_i^N x_{ij}(t) \tag{4}$$

Where $u_{ij}(t)$ is a uniformly distributed value between 0 and 1, $a_{ij}(t)$ is the individual's local attractor, and $theta$ is the contraction–expansion coefficient. As a result, the particle's location in the QPSO algorithm may be calculated as follows:

$$x_i(t+1) = \begin{cases} a_{ij}(t) - \theta|Mbest_j(t) - x_{ij}(t)|\ln(1/u_{ij}(t)) & : r(0,1) > 0.5 \\ a_{ij}(t) + \theta|Mbest_j(t) - x_{ij}(t)|\ln(1/u_{ij}(t)) & : otherwise \end{cases} \tag{5}$$

In each generation, the particles travel around the local attractor $a_{ij}(t)$, which is formed with the P and G optimal locations as follows:

$$a_{ij}(t) = \phi_{ij}(t)P_{ij}(t) + (1 - \phi_{ij}(t))G_j(t), \; \phi_{ij}(t) \sim (0,1) \tag{6}$$

The next generation's particle position distribution is computed using the mean $Mbest$ of the P best locations of the particles.

$$Mbest_j(t) = \frac{1}{N}\sum_i^N P_{ij}(t) \tag{7}$$

The fundamental difference between the PSO and the QPSO is twofold: 1) a large search space owing to the exponential distribution of the particles; and 2) The particle's distance from its partners is considered, whereas in PSO, particles move freely to converge to the global best. Another benefit is that it only has one parameter, $theta$, which must be managed for convergence and whose value is reduced linearly using the equation:

$$\theta = (\theta_{max} - \theta_{min}) * (Maxit - t)/Maxit + \theta_{min} \tag{8}$$

Because it is simple to use and has been tried and tested on a variety of applications, [32]. As a result, we sought to apply the QPSO method to a discrete optimization problem in this study and compare its performance to that of state-of-the-art techniques.

4. Proposed Work

This section explains an improved QPSO algorithm. It is described in three stages. First, it incorporates the asexual reproduction operator into the population (ARO). Second, adaptive contraction-expansion coefficient is used to alleviate the issue of stagnation. Finally, the adaptive TCS method is then used to choose MR test cases. It is followed by the TCR technique for minimizing the size of the test suite as follows:

4.1. Population Update [3]

Appropriate mapping increases the algorithm's speed and efficacy, so, the real numbers are being updated to permutation series by applying the asexual reproduction method. The fix-up process creates a link between real numbers and test case sequences such that the current solution acquires the parent solution's properties [3]. When forming the bud from the parent, it keeps the offspring's possible values (larva). Alternatively, the algorithm recalculates and rounds the output to natural values. Don't care conditions (*) are used to replace out of range and identical particles.

4.2. Dynamic Contraction-Expansion Coefficient (θ) [34]

The value of the contraction-expansion coefficient θ in QPSO indicates the population's search radius. The bigger the value, the wider the particle search range; on the other hand, the smaller value narrow the search range. The evolution velocity coefficient α is introduced to adaptively alter θ:

$$\alpha = \frac{Gfit(t)}{Gfit(t-1)} \tag{9}$$

$\alpha \in (0,1)$ since the global optimal solution is always replaced by the solution with superior fitness as the iteration advances, specifically, $Gfit(t) >> Gfit(t-1) > 0$. The fitness value of global best varies significantly when the value of is α tiny. The evolution is speeding up at small α due to significant changes in $Gfit$ as the particles are beyond ideal location. As a result, θ needs to be increased for ensuring a quick optimization. The evolution gets slow down with large α and the particle search range is reduced, so is the θ for better optimization. The solution converges and the evolution stops at $\alpha = 1$. So (8) is replaced with:

$$\theta = \theta_{max} - \alpha\theta_{min} \tag{10}$$

here θ_{min} and θ_{max} are the minimum and maximum values and α is evolution velocity coefficient's weight. Also, the difference between $P(t) - G(t)$ approaches to zero as the iterations proceed, so this value is replaced with the mutation operator $x_{rj}(t) - x_{sj}(t)$, where r and s are the particles selected randomly from the population [34]. It is mathematically formulated as:

$$x_{ij}(t) = \phi_{ij}(t)(x_{rj}(t) - x_{sj}(t)) + (1 - \phi_{ij}(t))G_j(t) \pm \theta|Mbest_j(t) - x_{ij}(t)|\ln(1/u_{ij}(t)) \tag{11}$$

4.3. Adaptive Test Case Selection [5]

The test case selection approach was proposed [5] that takes into account the statements they cover in the modified version and the impact of failed test cases. It's a dynamic algorithm that picks test cases depending on their pass or fail information after each iteration of TCP step. During the selection process, it requires exact input information, which is quite important in uncovering errors. It's called adaptive as it revises the fault detection capabilities $(P(t))$ of unallocated test cases and chooses test cases based on existing and earlier historical data. The technique is described as:

$$Pot(s) = \begin{cases} Pot'(s) & : s \text{ is not run by } t' \\ Pot'(s) * q & : s \text{ is run by } t' \text{ and } t' \text{ is failed} \\ Pot'(s) * p & : s \text{ is run by } t' \text{ and } t' \text{ is passed} \end{cases} \tag{12}$$

Algorithm 1 Adaptive Test Case Selection (ATCS) Algorithm

```
Define potentials p and q, p+q=1
Initialize PT = Prioritized suite ST = empty test set with capacity mx
select the first p test cases that covers all faults and statements
st=1 and t=PT(1)=ST(1)
Find Pot(s) and empty PT(1)
While size(PT)>0
Find other test cases needed for full coverage
if (st=mx) then
    break
end if
st=st+1
for t=1:PT Calculate P(t)
end for
t=max(P)
Update Pot(s), PT=PT-t, ST=ST+t
Empty P array for reassignment
end while
Return selected test cases
```

Figure 1. ATCS Algorithm

Before picking the test case t', $Pot(s)$ is the chance of any statement s having additional errors. p and q are constants with values between 0 and 1. such that $p + q = 1$. It measures the influence of pass or fail status on the $Pot(s)$ of any statement s. The values of p and q to 0.15 and 0.85, and q is set high since the goal is to acquire a larger proportion of failed test cases than passed ones. It also uses (13) to update the $P(t)$ of the unallocated test cases t:

$$P(t) = \sum_{s \text{ is run by } t} Pot(s) \tag{13}$$

The test case t with the highest $P(t)$ is chosen, and other test cases' potentials are updated using the chosen test case's state. $P(t)$ of unassigned test cases t is also updated depending on their revised potentials. In other words, the unassigned test cases' fault detection capacity is recalculated. The shortlisting sequence is based on the most recent data obtained, and it picks the test case t having highest rank in the modified $P(t)$. If the test cases are tied, the initial copy of the test case order is used to break the tie. ST is created by removing the specified test case from PT. Continue the previous steps till the stopping requirements are fulfilled, i.e., it creates a sufficient test cases to achieve 100 percent statement and fault coverage (mx) as presented in the Figure 1. As the algorithm contains two loops that runs upto the size of test suite, so the time complexity of the algorithm is $O(n^2)$.

4.4. Test Case Reduction (TCR)

To reduce suite size and cost, the current best solution of every generation is passed for the duplicate verification, and the very first k test cases that cover the faults/statements early are opted. This technique has the advantage of exposing how the test cases are ranked precisely [6]. The pseudo code of TCR is given in Figure 2. Since it contains one loop for the test cases (n) and another for finding the faults or statements (m) so the time complexity of the algorithm can be calculated as $O(nm)$.

Figure 3 presented an Improved IQPSO algorithm consists of two for loops $Maxit$ and Pop. It also contains ATCS and TCR algorithms so overall complexity of the algorithm is $O(Maxit * Pop * (n^2 + nm))$.

Algorithm 2 Test Case Reduction Algorithm

Define test fault matrix M, Ranked test cases T,
Initialize Faults position array Pos and test cases indices array I
for $t = 1, 2, \ldots, size(T)$ **do**
  for $f = 1, 2, \ldots, size(Pos)$ **do**
    if (M(T(t),Pos(f))=1 and I(f)=0)) **then**
      I(f)=t
    end if
  end for
end for
Reduced array R = T(I)

Figure 2. TCR Algorithm

Algorithm 3 IQPSO Algorithm

Define Pop, $Maxit$, θ_{max}, θ_{min} and δ
Initialize random population x_i
for $t = 1, 2, \ldots, Maxit$ **do**
  for $i = 1, 2, \ldots, Pop$ **do**
    Calculate fitness $f(x_i)$
    Update P_i and G solutions
    Update $alpha$ and $theta$ using 9 and 10
    if $\alpha == 1$ for δ attempts
    Update $x_i(t + 1)$ using (11)
    end if
    Update $x_i(t + 1)$ using (5)
  end for
  Apply ATCS Algorithm 1
  Apply TCR Algorithm 2
end for
Return: Final solution

Figure 3. IQPSO Algorithm

5. Experimental Setup

This Section outlines an empirical study, including research questions, datasets, evaluation metrics, and the algorithms with which the proposed algorithm is compared. The formulated research questions are:

RQ1. What is the performance of the proposed algorithm for TCP?

The objective is to see if the suggested algorithm outperforms others. It also identifies which algorithm produces the best results, as well as the effect of various testing settings on algorithm performance.

RQ2. What is the performance of the proposed algorithm for TCS?

The goal is to investigate the efficiency of the provided strategies for the ATCS method, namely, test selection percentage, inclusivity of MR test cases, and reduction in cost percentage.

RQ3. What is the performance of the proposed algorithm for TCR?

The aim is to evaluate the effectiveness of the suggested algorithm to that of the other methods. Furthermore, to figure out which testing criteria improve TCR. Alternatively, to see how it impacts the coverage and fault detection capabilities of the test suite.

5.1. Experimental design

PSO, QPSO and latest variants of PSO, i.e., PSOGSA [5], DAPSO [6] and Adaptive PSO (AdPSO) [35] are the algorithms considered for comparison. Apart from these, the algorithm is also validated against the state-of-the-art algorithms like GA, BAT and recently proposed Grey Wolf Optimization (GWO) [36]. These methods were developed using MATLAB R2017 on a Dell laptop with an Intel i5 CPU, Windows 11, and 8GB of RAM. Due to their stochastic nature, the algorithms are performed for 30 times. These are used on 3 different Java applications (jtopas, ant, and jmeter) that are pulled from the software infrastructure repository (SIR) [37]. We have applied the algorithm on different versions of these programs. Table 2 provides more information.

Table 2. Subject Programs

Programs	Versions	KLOC	Classes	Methods	Test Cases	Type
ant	7	80.4	650	7524	878	JUnit
jmeter	5	43.4	389	3613	97	JUnit
jtopas	4	5.4	50	748	209	JUnit

The performance of the algorithms is influenced by parameter choices 2019b. As a result, we carefully choose the parameters based on a comprehensive review of related works as well as a trial-and-error process for determining optimal values. Table 3 also contains the data retrieved using the Taguchi approach.

Table 3. Parameter Settings of the algorithms

Algorithms	Parameters values
GA	$p_{cr} = 0.8$, $p_m = 0.1$, tournament selection, ordered crossover
BA	$r_o = 0.001$, $A_o = 1$, $f_{min} = 0$, $f_{max} = 1.5$, $\alpha = 0.9$, $\gamma = 0.99$
PSO, AdPSO	$c_1 = 1.5$, $c_2 = 2$, $w_{min} = 0.4$, $w_{max} = 0.9$
QPSO	$\theta_{min} = 0.5$, $\theta_{max} = 1.7$
IQPSO	$\theta_{min} = 0.5$, $\theta_{max} = 1.7$, $\delta = 5$
PSOGSA	$c1 = 1.5$, $c2 = 2$, $w_{min} = 0.4$, $w_{max} = 0.9$, $\alpha = 15$, $G_0 = 100$, $Sinemap$
DAPSO	$s = 0.2$, $a = 0.25$, $c = 0.6$, $f = 0.8$, $e = 0.8$, $c1 = 1.5$, $c2 = 2$, $w_{min} = 0.4$, $w_{max} = 0.9$
Common Parameters	$Pop = 100$, $Maxit = 1000$

5.2. Performance Measures

The following performance measures are used to validate the efficiency and efficacy of these algorithms:

5.2.1. Test Case Prioritization

To assess the robustness of the proposed technique, the test cases are selected using two separate testing criteria: fault and statement coverage. As a result, the commonly used fitness measurements and effectiveness measures are defined as follows:

Average Percentage of Fault Detection (APFD) is a measure of how well a system detects faults. It finds a weighted average of the detected defects based on where they are in the test suite [38]. It's computed as follows:

$$APFD = 1 - \frac{\sum_{i=1}^{m} TF(i)}{n * m} + \frac{1}{2 * n} \tag{14}$$

The location of the test case that detects the i^{th} fault is denoted by $TF(i)$, and the faults covered by n test cases is denoted by m. It's value lies in between 0 and 100, with greater being better. The Average Percentage of Statement Coverage (APSC) is calculated in the same way as the APFD.

5.2.2. Test Case Selection and Reduction

Test selection percentage, cost reduction percentage, fault detection percentage and coverage loss percentage are often used efficacy measures. In addition to these, Inclusivity measure is also used for TCS as follows:

Test Selection Percentage (TSP): It's the percentage selection in the size of the test suite.

$$TRP = \frac{st}{n} * 100 \tag{15}$$

Here st indicates the test cases selected from n test cases.

Inclusivity (I): The extracted MR test cases emr divided by the total MR test cases $totmr$ gives the inclusivity measure.

$$I = \frac{emr}{totmr} * 100 \tag{16}$$

Fault Detection Loss Percentage (FDLP): The ratio of the faults not covered by the minimized test suite nfl to the total faults covered by the original test suite tfc:

$$FDLP = \frac{nfl}{tfc} * 100 \tag{17}$$

Cost Reduction Percentage (CRP): It is a percentage of the test suite's cost that is reduced $rcost$ when compared to the original suite's cost $tcost$.

$$CRP = \frac{rcost}{tcost} * 100 \tag{18}$$

6. Results and Analysis

This Section experimentally assesses the proposed algorithm for TCP using statement and fault coverage criteria. TCS and TCR have been studied for their effects on fault coverage loss, statement coverage loss, and cost benefits. To determine the experimental results of a software, the cumulative average of all its iterations is employed. The performance metrics for each version are calculated using the average of 30 runs. For fitness metrics, boxplots and convergence curves are also shown. A one-way ANOVA test with a p-value $= 0.05$ is used to analyse the algorithms' output statistically. If $p < 0.05$ the null hypothesis is rejected, suggesting that the algorithms' difference is statistically significant. Further, Tukey simultaneous test is used to evaluate the pair - wise comparison of the methods.

6.1. Performance analysis of TCP (RQ 1)

Table 4 shows the mean fitness values and variance of the performance metrics as well as their corresponding Tukey group ranks for all the programs. Observations state that

IQPSO is statistically different from all other nature-inspired algorithms, with a p-value less than 0.05, for both statement and fault criteria except PSOGSA. Moreover, it suggests that there is no significant difference between means of 1) PSOGSA and QPSO 2) AdPSO and DAPSO 3) AdPSO and GWO 4) GA and PSO for both criteria. The convergence curves for one of the versions of the subject programs are illustrated in Figure 4. It shows that the proposed IQPSO algorithm possesses the high-quality solutions comparatively for both criteria.

Table 4. Comparisons of the algorithms for TCP over fault and statement coverages

Program Versions	Algorithms	Fitness functions wise TCP, Variance and Tukey Group Ranking (%)					
		APFD	Variance	TR	APSC	Variance	TR
jtopas	IQPSO	96.702	0.879	A	98.559	0.494	A
	PSOGSA	95.965	1.413	AB	98.229	0.842	AB
	QPSO	95.621	1.966	B	97.998	0.806	BC
	DAPSO	94.066	3.884	C	97.841	0.976	BCD
	AdPSO	93.93	4.576	CD	97.795	0.817	BCD
	GWO	93.118	4.299	D	97.706	1.895	DE
	GA	91.918	4.555	E	97.463	1.809	DE
	PSO	91.18	9.875	E	97.33	1.905	EF
	BAT	89.951	9.152	F	96.863	4.034	F
ant	IQPSO	95.337	7.283	A	98.105	0.976	A
	PSOGSA	94.344	10.394	AB	97.482	2.135	B
	QPSO	93.937	9.935	B	97.403	2.241	B
	DAPSO	93.292	14.327	BC	97.397	2.49	B
	AdPSO	92.53	15.674	C	97.239	2.85	B
	GWO	92.149	15.204	CD	96.586	4.991	C
	GA	91.304	17.977	DE	96.657	4.476	C
	PSO	90.577	19.975	EF	96.529	4.591	C
	BAT	90.094	20.42	F	95.924	6.397	D
jmeter	IQPSO	95.402	3.498	A	99.218	0.193	A
	PSOGSA	94.31	5.603	B	98.989	0.351	AB
	QPSO	93.696	6.492	B	98.987	0.358	AB
	DAPSO	92.137	10.134	C	98.916	0.359	B
	AdPSO	92.12	9.617	C	98.819	0.592	BC
	GWO	91.943	8.38	C	98.625	0.677	C
	GA	90.672	8.894	D	98.331	0.747	D
	PSO	89.396	10.309	E	98.22	0.703	D
	BAT	89.145	12.483	E	97.748	1.914	E

It is also observed that the most of the algorithms have equivalent variance in case of statement coverage. The boxplots in Figure 5 also depicts that the variation in algorithmic performance for fault coverage is larger than for statement coverage. It's because the faults are dispersed across the entire software. In other words, most test cases cover almost same statements, therefore, the boxplots of the statement coverage criteria are more compressed than the fault coverage criteria. Overall, the proposed IQPSO is superior to all other algorithms in terms of variance.

6.2. Performance analysis of TCS (RQ 2)

The performance of the test case selection is evaluated using test selection percentage, inclusivity and cost reduction percentages as follows:

6.2.1. Test Selection Percentage (TSP)

The full version study revealed a random pattern, indicating that all of the algorithms behave similarly. We are unable to determine which algorithm is superior to the others. However, according to the program analysis, DAPSO produced the best TSP for two out of three programs in both coverage (see Table 5). It can be observed that IQPSO, PSOGSA, QPSO, and AdPSO are better than PSO, GWO, GA, and BAT for fault coverage. On the

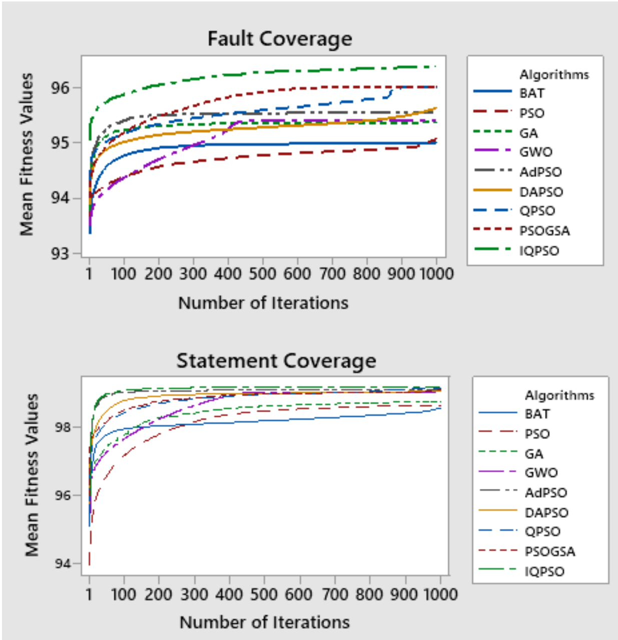

Figure 4. Convergence Curves of algorithms for fault and statement coverage criteria of TCP

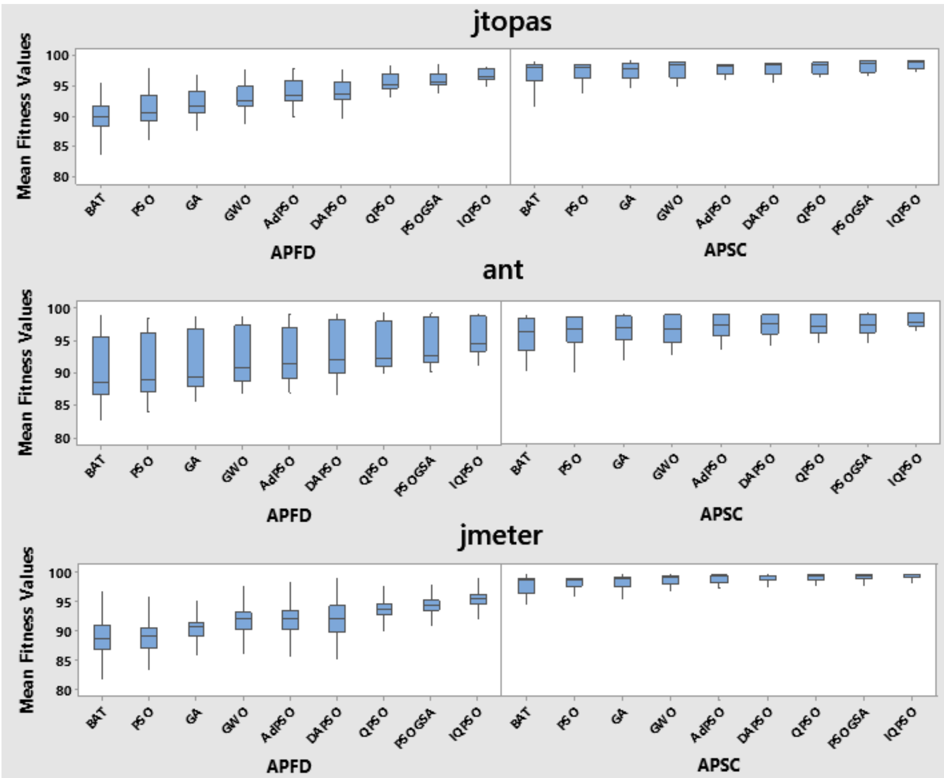

Figure 5. Boxplots of algorithms for fault and statement coverages of TCP

other side, GA, GWO, and PSO performed better than IQPSO, PSOGSA, AdPSO, QPSO and BAT in case of statement coverage. IQPSO performed better for *ant* which includes a significant number of test cases and statements. As a result, it can be said that the improved approach may outperform for the large programs. It is also observed that the selection percentage is less in case of fault coverage than the statement coverage.

6.2.2. Inclusivity (I)

All the algorithms are capable to incorporate over 78% and 76% *MR* test cases in statement and fault coverage. The proposed algorithm worked well for statement coverage, followed by PSOGSA, QPSO, DAPSO, AdPSO, PSO, GWO, GA, and BAT. However, IQPSO and PSOGSA performed least in case of fault coverage and the performance wise algorithms can be ranked as AdPSO, BAT, PSO, GA, QPSO, DAPSO, GWO, IQPSO, and PSOGSA. Alternatively, ATCS method picks a large number of test instances in case of statement coverage than the fault coverage criteria. Table 5 also showed that the fault coverage criteria is better for inclusiveness of the *MR* test cases. The inclusivity of variable state test cases is critical since they necessitate extra care because they do not produce same results for all versions. In other words, the ATCS method is based on the modification coverage so the fault coverage is more appropriate choice for inclusivity of the *MR* test cases over statement coverage.

6.2.3. Cost Reduction Percentage (CRP)

TCS has witnessed a cost reduction of 6.93–30.26% and 18.78-48.73% for statement and fault coverage criteria. In case of *ant* and *jmeter*, DAPSO delivers the best cost reduction % in most of the cases, whereas IQPSO outperformed DAPSO for statement coverage in *ant* and fault coverage in *jtopas*. In other words, DAPSO, GA, GWO and PSO performed better than the PSO variants in statement coverage, whereas IQPSO is the first runner up for fault coverage after DAPSO (see Table 5). It is also discovered that the TSP and the CRP have an indirect link. In other words, the lower the number of tests in the suite, the higher the CRP.

Table 5. Comparisons of the algorithms for TCS over fault and statement coverages

Program Versions	Algorithms	TSP		Inclusivity		CRP	
		TSP_{APSC}	TSP_{APFD}	I_{APSC}	I_{APFD}	CRP_{APSC}	CRP_{APFD}
jtopas	IQPSO	81.235	**51.750**	**89.024**	87.834	17.356	**48.731**
	PSOGSA	81.567	53.567	88.457	86.238	18.986	48.001
	QPSO	83.485	51.833	87.422	81.034	19.084	48.767
	DAPSO	**68.960**	59	87.644	84.387	**30.265**	41.83
	AdPSO	80.518	61.5	85.087	**92.083**	19.711	29.359
	GWO	74.035	64.75	82.39	75.67	26.125	34.918
	GA	77.926	77.867	84.263	85.544	21.829	23.966
	PSO	71.593	65.5	86.672	89.792	26.897	34.695
	BAT	83.368	73.858	81.536	90.424	17.451	32.993
ant	IQPSO	**85.333**	59.667	**80.449**	79.509	**14.321**	31.527
	PSOGSA	86.468	62.879	79.298	80.687	12.686	30.878
	QPSO	88.81	67.622	79.5	89.621	10.016	40.798
	DAPSO	87.995	**55.667**	79.423	85.064	11.378	**45.799**
	AdPSO	90.905	77	78.363	**95.803**	8.487	22.966
	GWO	90.057	72.533	78.506	88.889	8.961	27.57
	GA	89.914	72.944	77.68	93.953	9.054	28.057
	PSO	91.014	63.889	79.752	94.038	7.887	36.914
	BAT	89.676	79.333	78.583	94.915	9.259	20.893
jmeter	IQPSO	86.483	62.4	**96.940**	78.305	12.518	35.895
	PSOGSA	88.567	63.588	95.876	77.365	11.568	34.757
	QPSO	91.476	66.15	94.828	84.463	6.934	31.637
	DAPSO	**82.016**	**61.540**	94.149	83.014	**15.728**	**36.501**
	AdPSO	90.843	71.89	96.02	**96.367**	7.656	26.219
	GWO	82.903	74.19	91.695	87.447	15.025	23.725
	GA	82.363	77.51	91.557	95.509	15.251	21.195
	PSO	84.343	69.64	94.413	88.404	14.75	27.25
	BAT	86.223	78.37	93.147	92.617	12.894	18.783

6.3. Performance analysis of TCR (RQ 3)

The performance of the test case reduction is analyzed by calculating test selection percentage, cost reduction percentage and fault detection loss percentages as follows:

6.3.1. Test Selection Percentage (TSP)

Table 6 shows that all the methods perform almost equally well when it comes to reducing the test suite. Nonetheless, the proposed algorithm performed better than the other nature-inspired algorithms for both coverage. Comparatively, BAT has a higher selection percentage. TSP is larger in the case of statement coverage criterion than in the case of fault coverage. It is because there's a lot of redundancy in statement coverage, and the faults are spread over the whole program and to balance them APSC has slightly higher TSP than APFD.

6.3.2. Fault Detection Loss Percentage (FDLP)

Incorporation of ATCS helped the TCR in reducing the suite size with complete statement coverage and minimised the fault loss too. The findings reveal that direct application of TCR gave quite high fault loss, i.e., in between 5% and 40% [5]. AdPSO outperformed the other methods for statement (0-0.318%) as well as fault coverage (0-2.887%). Table 6 shows that IQPSO has least loss in statement coverage as compared to other algorithms except *jtopas* where DAPSO worked better. Observations also depict that the fault loss in APFD (0-8.134%) is higher compared to that in APSC (0-1.121%). The reason for this is that faults are spread over the software. Hence, the fault coverage loss by removing certain statement redundancy. It may be deduced that the loss of coverage and the reduction in test suite size are inversely proportionate.

6.3.3. Cost Reduction Percentage (CRP)

The experimental findings show that the cost reduction in fault coverage is more than the statement coverage as it reduces the test suite better too. It is also observed that the cost reduction is inversely proportionate to the test selection percentage, i.e., the larger the decrease in test suite size, the lower the test suite execution cost. TCM costs are estimated to be roughly 60 and 40 percent lower than TCS for statement and fault coverage. Table 6 clearly shows that the CRP of IQPSO outperformed all other algorithms followed by PSOGSA, QPSO and DAPSO. GA, PSO, GWO and AdPSO have nearly identical performance. Overall, IQPSO have demonstrated superior search capabilities to solve the regression testing problem in all the three subject programs.

Table 6. Comparisons of the algorithms for TSR over fault and statement coverages

Program Versions	Algorithms	TSP		FDLP		CRP	
		TSP_{APSC}	TSP_{APFD}	$FDLP_{APSC}$	$FDLP_{APFD}$	CRP_{APSC}	CRP_{APFD}
jtopas	IQPSO	**22.085**	**19.642**	0.974	1.707	**77.911**	**82.373**
	PSOGSA	22.138	19.711	0.902	1.909	77.656	81.876
	QPSO	22.341	19.717	0.905	2.302	77.85	81.253
	DAPSO	22.626	19.742	**0.000**	1.372	77.576	81.333
	AdPSO	23.593	23.45	0.974	**0.000**	76.44	76.45
	GWO	22.426	20.583	0.905	1.949	77.71	80.775
	GA	23.718	21.783	1.112	**0.000**	76.569	78.67
	PSO	22.626	21.842	0.835	1.064	77.505	79.164
	BAT	24.518	24.642	0.399	0.764	75.446	75.894
ant	IQPSO	**30.998**	**27.933**	**0.318**	8.134	**69.275**	**73.407**
	PSOGSA	31.755	28.234	0.412	6.689	68.587	73
	QPSO	31.425	27.933	0.434	5.975	67.295	72.07
	DAPSO	31.283	29	0.337	**2.887**	67.288	71.014
	AdPSO	32.278	30	**0.318**	3.998	66.886	70.209
	GWO	33.267	29.667	0.89	5.888	65.408	70.127
	GA	33.891	27.780	0.359	7.982	65.028	73.513
	PSO	32.61	28.333	1.121	4.572	65.855	72.17
	BAT	35.291	30.333	0.446	3.696	64.574	70.274
jmeter	IQPSO	**21.767**	**19.000**	**0.000**	1.387	**77.132**	**81.248**
	PSOGSA	22	19.354	**0.000**	1.076	76.453	80.653
	QPSO	22.272	19.94	**0.000**	0.752	76.541	79.73
	DAPSO	22.003	20	**0.000**	2.331	76.535	78.874
	AdPSO	23.088	21.2	**0.000**	**0.000**	75.643	77.755
	GWO	22.834	22.4	**0.000**	0.321	76.349	77.455
	GA	22.684	21.47	**0.000**	**0.000**	76.193	78.171
	PSO	22.044	21.4	**0.000**	0.752	76.859	77.102
	BAT	23.75	23.16	**0.000**	0.357	74.837	77.237

7. Conclusions

In this paper, we have suggested an improved QPSO algorithm for regression testing and validated it against GA, GWO, BAT, PSO, and its variants DAPSO, PSOGSA and AdPSO. Empirical results show that the proposed algorithm IQPSO have comparatively low variance than other variants for statement and fault coverage. Further, the adaptive test selection approach was able to successfully identify 77-96% of the MR test cases in both fault and statement coverage. The study also revealed that the adaptive test selection percentage of fault coverage is 40-60% lesser than the statement coverage with high inclusivity. IQPSO performed better than all other algorithms for test case reduction and cost reduction %. The algorithms showed approximate 7% difference in the fault detection capability loss for fault coverage over statement coverage. In the future, we will strive to reduce this fault detection loss to almost zero and validate the algorithm's results on a variety of large-scale real-world applications. We intend to investigate alternative variants of QPSO algorithms by modification and hybridization to improve the inclusivity and algorithm's performance even further.

Author Contributions: "Data curation, Anu Bajaj; Formal analysis, Anu Bajaj and Ajith Abraham; Funding acquisition, Lubna Abdelkareim Gabralla; Investigation, Anu Bajaj; Methodology, Anu Bajaj; Project administration, Ajith Abraham and Lubna Abdelkareim Gabralla; Resources, Anu Bajaj; Software, Anu Bajaj; Supervision, Ajith Abraham; Validation, Anu Bajaj; Visualization, Anu Bajaj, Ajith Abraham and Saroj Ratnoo; Writing – original draft, Anu Bajaj; Writing – review editing, Anu Bajaj, Ajith Abraham, Saroj Ratnoo and Lubna Abdelkareim Gabralla."

Funding: "The APC was funded by the Princess Nourah bint Abdulrahman University Researchers Supporting Project number (PNURSP2022R178), Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia".

Institutional Review Board Statement: "Not applicable"

Informed Consent Statement: "Not applicable"

Data Availability Statement: "Not applicable"

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

Abbreviations

ACO	Ant Colony Optimization
AdPSO	Adaptive Particle Swarm Optimization
ANOVA	Analysis of Variance
APFD	Average Percentage of Fault Detection
APSC	Average Percentage of Statement Coverage
ARO	Asexual Reproduction Operator
ATCS	Adaptive Test Case Selection
BAT	Bat Algorithm
CRP	Cost Reduction Percentage
CSA	Cuckoo Search Algorithm
DA	Dragonfly Algorithm
FDLP	Fault Detection Loss Percentage
FP	Fault Position Array
GA	Genetic Algorithm
GWO	Grey Wolf Optimization
I	Inclusivity
IQPSO	Improved Quantum behaved Particle Swarm Optimization
MR	Modification Revealing test cases
PSO	Particle Swarm Optimization
QPSO	Quantum behaved Particle Swarm Optimization
SIR	Software Infrastructure Repository
TCP	Test Case Prioritization
TCR	Test Case Reduction
TCS	Test Case Selection
TFP	Test Fault Matrix
TRP	Test Reduction Percentage

Nomenclature

α	evolution velocity coefficient
ϕ	random number
θ	Contraction-expansion coefficient
A	loudness of bat

a	local attractor
c_1, c_2	social and cognitive components
emr	extracted modification revealing test cases
$f()$	Fitness function
f_{min}, f_{max}	minimum and maximum frequency of bats
G	Global best particles
$Gfit()$	fitness value of global best particle
$Gus(x)$	Gaussian distribution of x
m	number of faults
max	maximum capacity of test suite to be selected
$Maxit$	Maximum number of iterations
$Mbest$	mean best of personal best particles
n	number of test cases
nfl	number of faults not covered
$P()$	Priority of test case
p, q	potential coefficients
p_{cr}	crossover probability
P_i	Personal best particles
p_m	mutation probability
Pop	Maximum number of population
$Pot(s)$	Potential of statement
PT	Permuted Test Suite
R	Requirement Set
r_o	pulse emission rate of bat
$rcost$	reduced cost of test suite
$RSInd$	Indices of reduced array
s	statement covered by test case
ST	Selected Test Suite
st	Selected test cases
T_i	Test case of the test suite
$tcost$	total cost of test suite
tfc	total faults covered
$totmr$	total modification revealing test cases
u	uniform random number
v_i	Velocity of a particle
w	inertia weight
x_i	Position of a particle

References

1. Webert, H.; Döß, T.; Kaupp, L.; Simons, S. Fault Handling in Industry 4.0: Definition, Process and Applications. Sensors 2022, 22, 2205. https://doi.org/10.3390/s22062205.
2. Yoo, S. and Harman, M. Regression testing minimization, selection and prioritization: a survey. Software Testing, Verification and Reliability, vol. 22. no. 2, 2012, pp.67-120. https://doi.org/10.1002/stvr.430
3. Bajaj, A. and Sangwan, O.P. Discrete cuckoo search algorithms for test case prioritization. Applied Soft Computing, vol. 110, 2021, p.107584. https://doi.org/10.1016/j.asoc.2021.107584
4. Li, Z.; Harman, M. and Hierons, R.M. Search algorithms for regression test case prioritization. IEEE Transactions on Software Engineering, vol. 33, no. 4, 2007, pp.225-237. https://doi.org/10.1109/TSE.2007.38
5. Bajaj, A. and Sangwan, O.P., Tri-level regression testing using nature-inspired algorithms. Innovations in Systems and Software Engineering, 17(1), pp.1-16. https://doi.org/10.1007/s11334-021-00384-9
6. Bajaj A. and Abraham A. Prioritizing and Minimizing Test Cases Using Dragonfly Algorithms. International Journal of Computer information Systems and Industrial Management Applications, 13, 2021, pp. 062-071. ISSN 2150-7988.
7. Shaukat, N., Ahmad, A., Mohsin, B., Khan, R., Khan, S.U.D. and Khan, S.U.D., Multiobjective Core Reloading Pattern Optimization of PARR-1 Using Modified Genetic Algorithm Coupled with Monte Carlo Methods. Science and Technology of Nuclear Installations, 2021. https://doi.org/10.1155/2021/1802492
8. Fister, J.I., Yang, X.S., Fister, I., Brest, J., and Fister, D., A brief review of nature-inspired algorithms for optimization. arXiv preprint arXiv:1307.4186, 2013, pp. 116-122.
9. Zhang, W., Qi, Y., Zhang, X., Wei, B., Zhang, M. and Dou, Z. 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), 2019, pp. 2767-2773. 10.1109/HPCC/SmartCity/DSS.2019.00388
10. Ahmed, B.S. Test case minimization approach using fault detection and combinatorial optimization techniques for configuration-aware structural testing. Engineering Science and Technology, an International Journal, vol. 19, no. 2, 2016, pp.737-753. https://doi.org/10.1016/j.jestch.2015.11.006
11. Mohapatra S.K., Prasad S., Test case reduction using ant colony optimization for object oriented program. Int J Electrical Comput Eng, vol. 5, no. 6, 2015, pp.2088–8708. oai:ojs.www.iaescore.com:article/5766.
12. Zhang Y.N., Yang H., Lin Z.K., Dai Q., Li Y.F., A test suite reduction method based on novel quantum ant colony algorithm. In: 2017 4th international conference on information science and control engineering (ICISCE). IEEE, 2017, pp 825-829. https://doi.org/10.4018/IJAMC.2022010106
13. Mondal D, Hemmati H, Durocher S, Exploring test suite diversification and code coverage in multi-objective test case selection. In: 2015 IEEE 8th international conference on software testing, verification and validation (ICST). IEEE, 2015, pp 1-10. https://doi.org/10.1109/ICST.2015.7102588
14. Khatibsyarbini, M., Isa, M.A. and Jawawi, D.N.A., Particle swarm optimization for test case prioritization using string distance. Advanced Science Letters, vol. 24, no. 10, 2018, pp.7221-7226. https://doi.org/10.1166/asl.2018.12918
15. De Souza, L.S., Prudêncio, R.B., Barros, F.D.A. and Aranha, E.H.D.S. Search based constrained test case selection using execution effort. Expert Systems with Applications, vol. 40, no. 12, 2013, pp.4887-4896. https://doi.org/10.1016/j.eswa.2013.02.018
16. De Souza L.S., Prudêncio R.B., Barros F.D.A., A hybrid binary multi-objective particle swarm optimization with local search for test case selection. In: Brazilian conference on intelligent systems. IEEE, 2014, pp 414-419. 10.1109/BRACIS.2014.80
17. De Souza L.S., Prudêncio R.B.C., De Barros F.A., A hybrid particle swarm optimization and harmony search algorithm approach for multi-objective test case selection. J Brazilian Comput Society 21(1), 2015, pp. 1-20, Springer. https://doi.org/10.1186/s13173-015-0038-8
18. Correia D. An industrial application of test selection using test suite diagnosability. In: Proceedings of the 2019 27th ACM joint meeting on european software engineering conference and symposium on the foundations of software engineering, 2019, pp. 1214-1216. https://doi.org/10.1145/3338906.3342493
19. Bharathi M. Hybrid particle swarm and ranked firefly metaheuristic optimization-based software test case minimization. Int J Appl Metaheuristic Comput, 13(1), 2022, pp. 1-20. https://doi.org/10.4018/IJAMC.290534
20. Nayak, G. and Ray, M., Modified condition decision coverage criteria for test suite prioritization using particle swarm optimization. International Journal of Intelligent Computing and Cybernetics. Vol. 12 No. 4, 2019, pp. 425-443. https://doi.org/10.1108/IJICC-04-2019-0038
21. Deneke, A., Assefa, B.G. and Mohapatra, S.K., Test suite minimization using particle swarm optimization. Materials Today: Proceedings, 2022, pp. 1-5. https://doi.org/10.1016/j.matpr.2021.12.472
22. Samad, A., Mahdin, H.B., Kazmi, R., Ibrahim, R. and Baharum, Z., Multiobjective Test Case Prioritization Using Test Case Effectiveness: Multicriteria Scoring Method. Scientific Programming, vol. 2021, 2021, pp. 1-13. https://doi.org/10.1155/2021/9988987
23. Agrawal, A.P. and Kaur, A., A comprehensive comparison of ant colony and hybrid particle swarm optimization algorithms through test case selection. In Data engineering and intelligent computing, 2018, pp. 397-405. Springer, Singapore. https://doi.org/10.1007/978-981-10-3223-338.
24. Lodewijks, G., Cao, Y., Zhao, N. and Zhang, H., Reducing CO Emissions of an Airport Baggage Handling Transport System Using a Particle Swarm Optimization Algorithm. IEEE Access, 9, 2021, pp.121894-121905. https://doi.org/10.1109/ACCESS.2021.3109286

25. Sun, J., Xu, W. and Feng, B., A global search strategy of quantum-behaved particle swarm optimization. In IEEE Conference on Cybernetics and Intelligent Systems, vol. 1, 2004, pp. 111-116. https://doi.org/10.1109/ICCIS.2004.1460396
26. Lukemire, J., Mandal, A. and Wong, W.K., d-qpso: a quantum-behaved particle swarm technique for finding d-optimal designs with discrete and continuous factors and a binary response. Technometrics, vol. 61, no. 1, 2019, pp.77-87. https://doi.org/10.1080/00401706.2018.1439405
27. Iliyasu, A.M.; Fatichah, C. A Quantum Hybrid PSO Combined with Fuzzy k-NN Approach to Feature Selection and Cell Classification in Cervical Cancer Detection. Sensors 2017, 17, 2935. https://doi.org/10.3390/s17122935.
28. Peng, C.; Yan, J.; Duan, S.; Wang, L.; Jia, P.; Zhang, S. Enhancing Electronic Nose Performance Based on a Novel QPSO-KELM Model. Sensors 2016, 16, 520. https://doi.org/10.3390/s16040520.
29. Guo, X.; Peng, C.; Zhang, S.; Yan, J.; Duan, S.; Wang, L.; Jia, P.; Tian, F. A Novel Feature Extraction Approach Using Window Function Capturing and QPSO-SVM for Enhancing Electronic Nose Performance. Sensors 2015, 15, 15198-15217. https://doi.org/10.3390/s150715198.
30. Wen, T.; Yan, J.; Huang, D.; Lu, K.; Deng, C.; Zeng, T.; Yu, S.; He, Z. Feature Extraction of Electronic Nose Signals Using QPSO-Based Multiple KFDA Signal Processing. Sensors 2018, 18, 388. https://doi.org/10.3390/s18020388.
31. Coelho, L. dos S., Gaussian quantum-behaved particle swarm optimization approaches for constrained engineering design problems. Expert Systems with Applications, vol. 37, no. 2, 2010, 1676–1683. https://doi.org/10.1016/j.eswa.2009.06.044
32. Omkar, S.N., Khandelwal, R., Ananth, T.V.S., Naik, G.N. and Gopalakrishnan, S., Quantum behaved particle swarm optimization (QPSO) for multi-objective design optimization of composite structures. Expert Systems with Applications, vol. 36, no. 8, 2009, pp.11312-11322. https://doi.org/10.1016/j.eswa.2009.03.006
33. Kennedy, J. and Eberhart, R., Particle swarm optimization. In Proceedings of ICNN'95-international conference on neural networks, vol. 4, 1995, pp. 1942-1948. IEEE. https://doi.org/10.1109/ICNN.1995.488968
34. Guo, X., Song, X. and Zhou, J.T., A synergic quantum particle swarm optimisation for constrained combinatorial test generation. IET Software, 2022, pp. 1-22. https://doi.org/10.1049/sfw2.12054
35. Nabi, S.; Ahmad, M.; Ibrahim, M.; Hamam, H. AdPSO: Adaptive PSO-Based Task Scheduling Approach for Cloud Computing. Sensors 2022, 22, 920. https://doi.org/10.3390/s22030920
36. Gupta, D., Gupta, V. (2017). Test Suite Prioritization Using Nature Inspired Meta-Heuristic Algorithms. In: Madureira, A., Abraham, A., Gamboa, D., Novais, P. (eds) Intelligent Systems Design and Applications. Advances in Intelligent Systems and Computing, vol 557. Springer, Cham. https://doi.org/10.1007/978-3-319-53480-022
37. Do, H., Mirarab, S., Tahvildari, L. and Rothermel, G., The effects of time constraints on test case prioritization: A series of controlled experiments. IEEE Transactions on Software Engineering, vol. 36, no. 5, 2010 pp.593-617. https://doi.org/10.1109/TSE.2010.58
38. Elbaum, S., Malishevsky, A.G., and Rothermel, G., Test case prioritization: A family of empirical studies. IEEE Transactions on Software Engineering, vol. 28, no. 2, 2002, pp.159-182. https://doi.org/10.1109/32.988497.