Iranian Journal of Numerical Analysis and Optimization
Vol. 14, No. 2, 2024, pp 417–448
https://doi.org/10.22067/ijnao.2024.85066.1333
https://ijnao.um.ac.ir/

Research Article

An innovative particle physics optimization algorithm for efficient test case minimization in software testing

U. Jaiswal*, and A. Prajapati

Abstract

Software testing is a crucial step in the development of software that guarantees the dependability and quality of software products. A crucial step in software testing is test case minimization, which seeks to minimize the number of test cases while ensuring maximum coverage of the system being tested. It is observed that the existing algorithms for test case minimization still suffer in efficiency and precision. This paper proposes a new optimization algorithm for efficient test case minimization in software testing. The proposed algorithm is designed on the base parameters of the metaheuristic algorithms, inspired by scientific principles. We evaluate the performance of the proposed algorithm on a benchmark suite of test cases from the

*Corresponding author

Received 1 November 2023; revised 16 December 2023; accepted 7 January 2024

Updesh Jaiswal
Department of Computer Science Engineering and Information Technology, Jaypee Institute of Information Technology Noida, India. e-mail: updesh1984@gmail.com

Amarjeet Prajapati
Department of Computer Science Engineering and Information Technology, Jaypee Institute of Information Technology Noida, India. e-mail: amarjeetnitkkr@gmail.com

How to cite this article
Jaiswal, U. and Prajapati, A., An innovative particle physics optimization algorithm for efficient test case minimization in software testing. *Iran. J. Numer. Anal. Optim.*, 2024; 14(2): 417-448. https://doi.org/10.22067/ijnao.2024.85066.1333

literature. Our experimental results show that the proposed algorithm is highly effective in reducing the number of test cases while maintaining high coverage of the system under test. The algorithm outperforms the existing optimization algorithms in terms of efficiency and accuracy. We also conduct a sensitivity analysis to investigate the effect of different parameters on the performance of the proposed algorithm. The sensitivity analysis results show that the performance of the algorithm is robust to changes in the parameter values. The proposed algorithm can help software testers reduce the time and effort required for testing while ensuring maximum coverage of the system under test.

AMS subject classifications (2020): Mathematics Education 97-XX; Computer Science 97P10, 97P40, 97P80.

Keywords: Metaheuristic optimization; Test cases minimization; Software engineering; Nature inspired algorithm.

1 Introduction

Software testing is a crucial step in the creation of software that tries to guarantee the dependability and quality of software products. Test case reduction, which seeks to lower the number of test cases necessary to obtain maximum coverage of the system under test, is one of the crucial jobs in software testing. The cost and work associated with software testing can be considerably decreased by test case minimization, which can also increase the process's effectiveness and efficiency.

Test case minimization includes selecting a subset of test cases from a bigger set of test cases to realize the same scope. The chosen subassemblies must be able to distinguish all absconds and disappointments of the framework beneath the test. The test case diminishment preparation is frequently done physically, which can be time-consuming and error-prone.

To computerize the method of test case minimization, different optimization calculations have been created [21]. These calculations utilize scientific and computational methods to produce an ideal subset of test cases that accomplish the most extreme scope with the least number of test cases. A few of

the well-known optimization calculations utilized for test case minimization incorporate genetic algorithm (GA) [2], Ant colony optimization (ACO) [32], particle swarm optimization (PSO) [14], and simulated annealing. The work presented in [17] gives a description of the use of search-based techniques in test suite reduction with a detailed systematic review.

Whereas reduction in the testing time, lower testing expenses, increased testing efficiency, and higher-quality software products are some of the advantages of test case minimization. Test case minimization can also help identify redundant and unnecessary test cases, which can be removed from the test suite to further improve the efficiency of the testing process.

The existing optimization algorithms for test case minimization have some limitations in terms of efficiency and accuracy. Therefore, there is a need for an innovative optimization algorithm that can overcome these limitations, provide a more efficient, and accurate solution for test case minimization in software testing.

The utilization of metaheuristic calculations for test case minimization can offer assistance to overcome the impediments of conventional optimization calculations, such as thorough look and covetous calculations. Metaheuristic calculations are planned to handle complex optimization issues with expansive look spaces and nonlinear objective capacities, making them well-suited for test case minimization. However, based on the particular needs and demands of the software development project, the goals of test case minimization utilizing metaheuristic algorithms can change. The goals can incorporate lessening the number of test cases, diminishing the testing time, making strides in the proficiency of the testing handle, moving forward the quality of the program item, and lessening the fetching of testing.

This gives rise to the motivation to work on the optimization of the test cases in software engineering. This inspiration for test case minimization utilizing metaheuristic calculations is to optimize the choice of test cases to decrease the number of test cases, guaranteeing that the chosen test cases accomplish the most extreme scope of the framework beneath the test. By lowering the cost and labor needed for software testing, these fewer test cases increase the efficacy and efficiency of the testing process.

The motivation behind the development of optimization algorithms for software testing stems from the need to address the challenges posed by the complexity of modern software, the demand for efficient resource utilization, the drive for higher software quality, the acceleration of development cycles, and the imperative to reduce testing costs. These algorithms offer a strategic and automated approach to testing that aligns with the dynamic nature of contemporary software development practices.

As software applications become increasingly intricate, with intricate interactions and dependencies, traditional manual testing methods struggle to provide adequate coverage and efficiency. Optimization algorithms aim to address these challenges by automating and improving the test case selection process. One key motivation is the need for effective resource utilization. Software testing can be resource-intensive, requiring significant time and human effort. Optimization algorithms help streamline this process by intelligently selecting a subset of test cases that provide maximum coverage and effectiveness. This efficiency not only saves time but also ensures that testing resources are allocated to the most critical areas of the software.

With such a motivation, the main contributions of the proposed metaheuristic algorithm for test case minimization are as follows:

1. The proposed algorithm produces an ideal subset of test cases that accomplishes the greatest coverage with the least number of test cases.

2. The algorithm is outlined to handle complex optimization issues with expansive look spaces and nonlinear objective capacities, making it more proficient than conventional optimization algorithms.

3. The proposed algorithm takes into account various factors such as code coverage, test suite size, and execution time to generate an optimal test suite that achieves maximum coverage with the minimum number of test cases.

4. The proposed algorithm can be customized to fulfill the particular demands and needs of the software development project.

5. The proposed algorithm is evaluated on a benchmark suite of test cases from the literature.

The rest of the paper is arranged in the subsequent section, highlighting important modules as the background-related work is presented in Section 2. The description of the proposed metaheuristic algorithm is presented in Section 3. In Section 4, the experimentation results of the proposed algorithm are presented. Later, a comparative study is presented in Section 5. Finally, the paper gives the key summarization of the work and is presented in Section 6.

2 Literature review

In the past, there has been significant work done on test case minimization using metaheuristic algorithms in the field of software engineering. There are several works on test case prioritization using particle swarm optimization, where the authors propose a test case prioritization technique using PSO. The PSO algorithm is used to select the most critical test cases for early execution, reducing the testing time and effort required [33, 31, 20]. However, there are many other works that focus on the reverse aspect, that is, generation of the test cases using the PSO algorithm. One such work is demonstrated in [29], where the PSO-based search technique is applied with an improved combined function to generate test cases for the critical paths. Another work that gives the comparative study on metaheuristic algorithms for test case generation is presented in [28]. The authors present the details of using the various optimization algorithms and their fitness functions to generate the test cases.

Some of the authors have also focused on the method of hybridization to minimize the number of test cases. One such work is proposed in [9], where the authors used the collective information of the two metaheuristic algorithms to minimize the feature description size. Another work by the authors of [8] presents the study on the test case selection, and reduction using Quantum PSO. Also, the work using the PSO for regression testing and its test case minimization is presented in [34]. This paper proposes a comparative analysis approach for test case selection for regression testing using PSO. The PSO algorithm is used to optimize the selection of test cases

based on various criteria, such as coverage and fault detection capability, while minimizing the size of the test suite.

One of the works published on test case prioritization used the PSO with the modified conditional criteria for test case minimization and selection is [25]. A recent work [15] presents the multi-objective modified PSO algorithm for the reduction of the test suite. In an overall assessment of the work done on PSO-based test case selection, it is found that the authors claimed that in terms of increasing the effectiveness and efficiency of the testing process, PSO had demonstrated encouraging results. Additionally, while retaining complete coverage of the system under test, PSO algorithms can considerably minimize the number of test cases needed for software testing. Indeed, in critical analysis, it is observed that PSO can become trapped in local optima, limiting its ability to explore the entire search space. This can lead to suboptimal solutions and reduce the effectiveness of the test suite. Also, PSO can become computationally expensive as the number of test cases increases. This can limit its scalability and make it impractical for larger software systems. In addition, the performance of PSO is highly sensitive to the choice of parameters, such as the number of particles, the maximum number of iterations, and the inertia weight. Poor parameter settings can lead to suboptimal performance and slow convergence. Thus, the PSO may not be found to be suitable for all types of software systems or test scenarios. Its effectiveness may depend on the nature of the system under test and the objectives of the testing process.

Some of the authors have used the ACO algorithm for the minimization of the test cases. One such work is reported in [24], where the authors used the complete graph to represent all the test cases in the test suite. An empirical analysis of using the ant systems for multi-faceted test case prioritization is presented in [26]. The authors validated their proposed approach on a freely available widely used benchmark dataset, called software infrastructure repository. Also, there is a work from early 2017, where the authors used the ACO approach for regression test case prioritization [11]. The authors here used the epistatic test case segment approach for multiobjective search-based regression test case prioritization reflects the correlation between genes in

the evolution process. Furthermore, the approach is validated on the three benchmarks and found satisfactory on industry standards.

In the literature, there are some other optimization algorithms as well that were used for the minimization and prioritization of the test cases [16]. On this, the work on using test case minimization with GA is presented in [30]. The creators utilized the center concept of the GA to construct an algorithm called *Test Reduce* to discover an optimized and negligible set of test cases for Internet applications. The modified strategy for test case minimization in the COTS method using the GA was employed by the authors of one of the research works [10]. To maximize fitness values in test suit development, the authors applied the GA in the boundary value analysis and partitioning testing.

Besides this, one of the papers proposes a hybrid approach that combines GA and Fuzzy logic for the minimization of the test cases. The authors used the GA to filter out and find the set of candidate test cases. Whereas, the fuzzy logic is used to select the optimal subset of test cases that minimize the size of the test suite while maximizing the fault detection ability [19].

In many other works, the literature suggests the use of the various other metaheuristic algorithms that were used in the optimization of the test data in a variety of applications. On this, the use of the gravitational search algorithm is also found useful in one of the articles for the test case minimization [6]. The authors here utilized the discrete and combinatorial gravitational look calculation to illuminate the test case prioritization and minimization. The try is carried out in a controlled environment with the benchmark dataset, and the results are compared with the GA. In comparison, it is found valuable within the ideal choice of the test cases.

On a similar aspect, the data optimization and prioritization using the novel BAT algorithm is proposed in [7]. The authors used the two-step procedure to enhance the work and modified the traditional BAT algorithm with the inclusion of the new objective function. The algorithm is tested on the dataset with different evaluation criteria. Later, the author presented the comparative analysis with the existing algorithms and found that the proposed approach is performing well.

Some of the authors used the hybrid approach as well for the minimization of the test cases. This hybridization is seen in many aspects by collaborating the multiple metaheuristic algorithms to form a new objective function. On a similar theme, the authors of [9] showed the hybridization of the two metaheuristic algorithms named molecule swarm and positioned firefly calculation for the program test case minimization. Some of the authors also used the standalone firefly algorithm [18] for the test case minimization. However, on comparative, analysis of these two variants suggests that the hybrid algorithm results are better than the use of the standalone method.

The writing moreover depicts the utilization of a half-breed technique for fault detection and combinatorial optimization procedures for configuration-aware basic testing [1] and the minimization utilizing the test code closeness and evolutionary search [27].

In an overall assessment of test case minimization, it is found that it is an important problem in software engineering that aims to reduce the size of the test suite while maintaining its effectiveness in detecting faults in the software system. To address this issue, analysts have connected different metaheuristic algorithms, such as GA, PSO, and ACO, to optimize the test suite. Some authors also explored the design framework of the novel design of the metaheuristic algorithm for software mutation testing in a controlled environment [4].

The previous contributions on test case minimization using metaheuristic algorithms have shown promising results in improving the efficiency and effectiveness of the testing process as also discussed in [12] presenting with a detailed decade review. These approaches have minimized the count of test cases that were actually required for software testing while maintaining maximum coverage of the system under test. The contributions have also highlighted the importance of considering the limitations of these algorithms, such as limited scalability, sensitivity to parameter settings, and limited diversity of solutions.

The comparative study of the related works used for the software test case minimization is presented in Table 1. The table presents the summarized results of the reviewed works and highlights the advantages and the limitations of the studies.

Table 1: Comparative study of the related works on software test case minimization

Reference	Optimization Approach	Advantages	Limitations
[16, 30, 10, 19]	Genetic Algorithms (GAs)	- Automatically explores diverse test cases	- May get stuck in local minima, especially for complex landscapes
		- Handles discrete search spaces	- High computational cost for large-scale problems
		- Captures interactions between test cases	- Difficult to interpret results
[33, 31, 20, 34, 29]	Particle Swarm Optimization (PSO)	- Efficient for continuous and discrete spaces	- Sensitive to parameter tuning
		- Simplicity in implementation	- Convergence to suboptimal solutions
		- Ability to escape local minima	-Slow convergence
[7]	BAT optimization	- Global search capability	- Slow convergence for certain problem landscapes
[9]	Molecule Swarm and Firefly Optimization	- Versatility in handling various objectives	- Parameter sensitivity
[18]	Firefly Optimization	- Provides a probabilistic framework	-Slow convergence
[24, 26, 11]	Ant Colony Optimization (ACO)	- Inspired by natural foraging behavior	- Limited scalability for large problem instances
		- Robustness to changes in the environment	- Convergence highly dependent on parameters
		- Handles discrete and combinatorial problems	-Slow convergence
[28, 9]	Hybrid algorithms and Comparative studies	- Synergy between exploration and exploitation	- Increased complexity in implementation
		- Improved global search capability	- Inherit limitations of individual algorithms
		- Enhanced convergence characteristics	-Slow convergence

3 Proposed metaheuristic optimization algorithm

The main requirement for an optimization algorithm in software test case minimization is to reduce the number of test cases required to test a system while still ensuring that the system's functionality is thoroughly tested. This is important because testing is a crucial step in software development, and reducing the number of test cases required can save time and resources. In addition to reducing the number of test cases, the optimization algorithm should also ensure that the remaining test cases provide maximum coverage of the system's functionality. This means that the algorithm should be able to identify the most critical and relevant test cases, rather than just selecting them randomly or based on simple criteria such as code coverage.

Furthermore, the optimization algorithm should be able to handle large and complex software systems and be efficient enough to provide results in

a reasonable amount of time. This is important because testing can be a time-consuming process, and any optimization algorithm used should not significantly increase the overall testing time.

Based on this, here we developed a new metaheuristic optimization algorithm called the *Physics-informed particle decay Algorithm* (PPDA), based on the nature of the physics to study the decay of the particles. The proposed approach is new in terms of idea and objective function.

3.1 Physics-informed particle decay algorithm (PPDA)

The PPDA is a metaheuristic optimization algorithm that is inspired by principles from physics, specifically regarding stability and particle decay. PPDA simulates the behavior of particles, where each particle represents a potential solution to the optimization problem being solved. The basic flowchart of the proposed PPDA algorithm is shown in Figure 1.

The concept of stability is important in PPDA, as it helps to ensure that the particles converge toward a good solution. The particles in PPDA are attracted toward the energy valleys, which represent the best solutions to the optimization problem. The valley itself represents a stable state, while the particles represent the energy fluctuations that occur in the system. The proposed algorithm uses the concept of stability to ensure that the particles move and converge toward the best solution.

The different modes of decay in PPDA are also important for the algorithm's performance. In physics, particle decay refers to the process by which a particle breaks down into smaller particles or other forms of energy. In the proposed algorithm, particle decay is used to simulate the process by which particles move toward each other.

There are several modes of particle decay in PPDA, each of which represents a different behavior of the particles. For example, the mode of "attractive decay" causes the particles to move toward each other and converge to the best solution. On the other hand, the mode of "repulsive decay" causes the particles to move away from each other and explore different areas of the search space.

3.1.1 Background preliminaries for PPDA

The base of the proposed algorithm is particle physics. In particle physics, the process of colliding two particles to generate new particles is known as particle collision. Particle collisions are used to study the properties of subatomic particles, such as quarks and leptons, and to probe the fundamental forces of nature. In a particle collision, two particles are accelerated to very high speeds using particle accelerators, such as the large hadron collider at CERN. When the particles collide, they can create a shower of new particles that are detected by particle detectors.

The collision itself is a physical reaction that involves the exchange of energy and momentum between the colliding particles. Depending on the energy of the colliding particles and the angle of the collision, different types of particles can be generated. One type of particle collision is known as elastic scattering, in which the colliding particles bounce off each other without generating any new particles. Inelastic collisions, on the other hand, can generate new particles, such as mesons and baryons.

The collision can also generate antiparticles, which are particles that have the same mass as their corresponding particles but have opposite charges. For example, a collision between a proton and an antiproton can generate a shower of particles, including pions, kaons, and other mesons.

Thus, we can see the two forms of the particles in the universe, that is, *stable* and *unstable* particles. Stable particles are particles that do not decay into other particles, while unstable particles decay into other particles after a certain amount of time.

Stable particles include fundamental particles such as electrons, neutrinos, and photons, as well as composite particles such as protons and neutrons. These particles have lifetimes that are much longer than the age of the universe and do not decay spontaneously. On the other hand, unstable particles, have lifetimes that are shorter than the age of the universe and decay into other particles through various physical processes.

Furthermore, as a fundamental feature of these particles, unstable particles emit energy through disintegration or decay. When an unstable particle decays, it transforms into one or more other particles, releasing energy in the

process. The decay of an unstable particle can occur through various physical processes, depending on the nature of the particle and the forces that govern its behavior. During the decay process, the unstable particle loses energy, which is released in the form of radiation, such as γ rays, or particles, such as electrons or positrons. This energy can be measured and used to study the properties of the unstable particle and the forces that govern its behavior.

The decay rate of an unstable particle is described by its half-life, which is defined as the amount of time it takes for half of the particles in a sample to decay. The half-life of a particle can range from fractions of a second to billions of years, depending on its nature.

However, determining the stability of a particle involves considering various factors, including the count of neutrons (Ne) and protons (Ze) in the nucleus, and the Ne/Ze ratio is one of the most crucial factors. The balance between the strong nuclear force, which binds protons and neutrons together, and the electromagnetic force, which repels positively charged protons, determines the stability of a nucleus. If the number of neutrons and protons is such that the strong force is greater than the electromagnetic force, the nucleus is stable. However, if the strong force is not strong enough to overcome the electromagnetic force, the nucleus is unstable and may undergo radioactive decay.

The Ne/Ze ratio is a key aspect in determining a nucleus's stability. It is the ratio of the number of neutrons to the number of protons in the nucleus. In general, a stable nucleus has a Ne/Ze ratio that is close to 1. If the Ne/Ze ratio is too high or too low, the nucleus is unstable and may undergo radioactive decay. For light nuclei, the stable Ne/Ze ratio is close to 1:1. However, as the atomic number increases, the stable Ne/Ze ratio increases slightly. In addition to the Ne/Ze ratio, the size of the nucleus also plays a role in its stability. Larger nuclei tend to be less stable than smaller nuclei and may undergo radioactive decay to become more stable.

There are three sorts of emissions that define the particle's level of stability during the decay process, that is, α *decay,* β *decay*, and γ *decay* [22]. All three decay particles are emitted from the nucleus, which can result in a change in the Ne/Ze ratio of the nucleus. In α decay, a nucleus emits an α particle, which consists of two protons and two neutrons. This emission

reduces the number of protons (Ze) and neutrons (Ne) in the nucleus by 2 and 4, respectively. As a result, the Ne/Ze ratio of the nucleus decreases, making it more stable.

In β decay, a neutron in the nucleus is changed into a proton or vice versa, and the nuclear material emits an electron (β particle), antineutrino, or positron (positron emission). Depending on the type of β decay, this alters the number of protons and neutrons in the nucleus, causing the Ne/Ze ratio to either rise or fall.

In γ decay, a nucleus in an excited state emits a γ ray, which is a high-energy photon. Moreover, γ decay does not change the number of protons or neutrons in the nucleus, and thus, it does not directly affect the Ne/Ze ratio. However, γ decay can occur as a result of α or β decay, which may change the Ne/Ze ratio.

Within the proposed approach, we have utilized the standards of the decay process through different particles that can serve as a starting point for the planning of the modern metaheuristic algorithm in which the arrangement candidates' execution can be improved by utilizing the thought that particles have the inclination to reach a steady position.

3.1.2 Mathematical foundation of PPDA

The mathematical structure of the proposed optimization algorithm uses the previously defined concept and preliminaries. The first step of the proposed algorithm is the initialization, wherein in the search space, the solution candidates (A_i) are assumed to be particles with changing degrees of solidness and expected to be a particular locale of the universe as appeared in supposed to be particles with varying degrees of stability and assumed to be a specific region of the universe as shown in the following equation (1):

$$A = \begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_i \\ \vdots \\ A_n \end{bmatrix} = \begin{bmatrix} A_1^1 & A_1^2 & \dots & A_1^j & \dots & A_1^m \\ A_2^1 & A_2^2 & \dots & A_2^j & \dots & A_2^m \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ A_i^1 & A_i^2 & \dots & A_i^j & \dots & A_i^m \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ A_n^1 & A_n^2 & \dots & A_n^j & \dots & A_n^m \end{bmatrix}, \tag{1}$$

where the symbolic representation of the used variables is defined as the total number of particles given by variable n, the problem dimension is represented by m, A_i^j is deciding the beginning position of the ith candidate for jth decision variable.

Moreover, the mathematical form of the decision variable is shown by the following equation (2):

$$A_i^j = A_{i,\min i}^j + R_m(A_{i,\max i}^j - A_{i,\min i}^j), \tag{2}$$

where $A_{i,\min i}^j$ and $A_{i,\max i}^j$ speaks to the lower and upper bounds of the jth variable within the ith candidate and the R_m is the irregular generator within the extend $[0, 1]$.

The core difference between neutron-rich and neutron-poor particles is then calculated using the objective function set for each particle utilizing the analysis with the Enrichment Level of Neutron (NEL) of the particles. NEL of a particle can be calculated using the formula given in the following equation (3):

$$NEL = \frac{(Ne - Ze)}{(Ne + Ze)}, \tag{3}$$

where Ne represents the number of neutrons in the nucleus, and Ze represents the number of protons in the nucleus.

The NEL value indicates the neutron excess or deficiency in the nucleus relative to the number of protons. If the NEL value is positive, it indicates a neutron excess, meaning there are more neutrons than protons in the nucleus. A positive NEL value suggests a tendency towards neutron-richness. If the NEL value is negative, it indicates a neutron deficiency, meaning there are fewer neutrons than protons in the nucleus. A negative NEL value suggests a tendency towards proton-richness.

Whereas the following equation (4) gives the computation of the core difference between neutron-rich and neutron-poor particles using NEL:

$$CD = \frac{\sum_{i=1}^{n} NEL_i}{n}, \tag{4}$$

where CD is the core difference between neutron-rich and neutron-poor particles, NEL_i is the NEL for ith particle, and n is the total number of particles.

Thus, if the NEL of a particle is discovered to be more than the CD, that is $(NEL_i > CD)$, the associated particle is assumed to have the next Ne/Ze proportion, and the rot handle is expected to be utilizing α, β, or γ plans.

Next, in the third phase, we will predict the particle's stability levels based on the objective function evaluated using the following equation (5):

$$L_i = \frac{NEL_i - B_L}{W_L - B_L}, \qquad i \in [1, n], \tag{5}$$

where L_i is the ith particle stability level, B_L is the particle's best stability level, W_L is the particle's worst stability level, and n is the total number of particles in the system.

Furthermore, to calculate the stability of the system (SS), we generate a value in the interval of 0 to 1, probably with a random selection. In case the predicted particle stability level (L_i) is higher than the selected SS, that is, $(L_i > SS)$ then α and γ decay are to be considered.

Based on the theory of particle physics for α decay, it is observed that the α rays were emitted to enhance the particle's stability. Using the same theory in the proposed algorithm as a position-updating technique to generate the new solution particle, two α values were generated, that is, α_1 and α_2. The generated α_i values are given by the following equation (6):

$$\alpha_1 \in [1, r], \qquad \alpha_2 \in [1, \alpha_1], \tag{6}$$

where α_1 is for the number of emitted rays, r is a random number in between 1 to n, and α_2 is the number of to-be-emitted α rays. These emitted rays are considered as the decision variables to generate the new particle, which is given by the following equation (7):

$$A_i^{new_\alpha} = A_i(A_{B_L}(A_i^j)), \tag{7}$$

where $i \in [1, n],\ j = \alpha_2,\ A_i^{new_\alpha}$ is the new particles generated in the sample space, A_i is the ith particle's current position in the sample space, A_{B_L} is the particle's position with the leading soundness level, and A_i^j is the jth choice particle for the transmitted beam.

Aside from the α decay, there is also a γ decay in which γ rays are emitted to strengthen the stability of the excited particles. In the proposed algorithm, this is considered as another particle's position-updating process to generate the new particles. On a similar pattern, like for α decay, here also we use two values of γ, that is, γ_1 and γ_2. Here, γ_1 signifies the number of emitted photons, and γ_2 in the range of [1, γ_1] denotes the information on the photons to be examined in the particles. Thus, these photons in the particles are considered to be the decision variable in the sample space, where the old particle is replaced with the neighboring particle. Finally, the distance between the particle of interest and other leftover particles is calculated as we do in the K-nearest neighbor algorithm. Here we select the nearest particle having the value of $k = 1$ and is given by the following equation (8):

$$\min(Dist_i^k) = \sqrt{(X_2 - X_1)^2 + (Y_2 - Y_1)^2}, \tag{8}$$

where $i \in [1, n],\ j \in [1, n-1],\ Dist_i^k$ is the distance between the ith particle and the neighbor k particles, and the pair (X_i, Y_i) is the particles coordinate in the sample space.

Using the above information, the second particle generating process is given by using the following equation (9):

$$A_i^{new_\gamma} = A_i(A_N(A_i^j)), \tag{9}$$

where $i \in [1, n],\ j = \gamma_2,\ A_i^{new_\gamma}$ is the new particles generated in the sample space, A_i is the ith particle's current position in the sample space, A_N is the ith particles neighborhood position vector, and A_i^j is the jth decision particle in space (emitted ray).

It is to be noted here that if the particle's stability level (L_i) is lower than the SS, that is, ($L_i < SS$), then the β decay is then thought to occur with increasingly unstable particles with lower stability levels.

Now, the mathematical representation of the particle position update via reaching towards the stability band is given by the following equation (10):

$$A_i^{Update_1} = A_i + \frac{(a.A_{B_L} - b.A_C)}{L_i}, \tag{10}$$

where $i \in [1, n]$, $A_i^{Update_1}$ and A_i are the updated and current positions of the ith particle, a and b are the two numbers randomly chosen from the interval of $[0, 1]$ and controls the particle's movement, and A_C is the position of the center of the particle in space and is given by the following equation (11):

$$A_C = \frac{\sum_{i=1}^{n} A_i}{n}. \tag{11}$$

Another position overhauling preparation is carried out for particles utilizing β rot, in which a controlled development towards the particle or a candidate with the leading solidness level (A_{B_L}) and a neighboring particle (A_N) is performed, while the particle's stability level has no bearing on the movement process. This strategy points to extending the misuse and investigation levels of the calculation which is given by the following equation (12):

$$A_i^{Update_2} = A_i + (c.A_{B_L} - d.A_N), \tag{12}$$

where $i \in [1, n]$, $A_i^{Update_2}$ and A_i are the updated and current positions of the ith particle, c, and d are the two numbers randomly chosen from the interval of $[0, 1]$ and control the particle's movement, and A_{B_L}, A_N have their usual meaning as discussed earlier.

When the NEL of a particle is discovered to be lower than the CD, that is, $(NEL_i < CD)$, this indicates that the particle has a lower value of Ne/Ze ratio. It means that the particle tends to emit positrons in order to come closer to the stability band. Using a similar concept in the proposed algorithm, the new particle position is updated by using the following equation (13):

$$A_i^{update} = A_i + e, \tag{13}$$

where $i \in [1, n]$, A_i^{Update} and A_i are the updated and current positions of the ith particle, e is the numbers randomly chosen from the interval of $[0, 1]$ and control the particle's movement.

Finally, after the execution of the proposed algorithm completely, at the end, we are with any one of the solution steps, that is, (1) if ($NEL_i > CD$), the algorithms return two newly generated particle positions for each of the particles as given by (10) and (12), and (2) if ($NEL_i < CD$), the algorithms return a single new particle position as given by (13). Hence, in an overall analysis, three processes for updating positions are included in the algorithm. Whereas one position upgrading method happens in arrangement candidates, where abuse is fulfilled, two of these methods happen in choice factors, where the investigation is performed. This technique's problematic aspect is that while the exploration phase may direct the program to locally optimal answers, the other phase aims to fine-tune the prior solutions to find the best candidate globally.

3.1.3 Algorithmic design of the proposed methodology

The description of the proposed algorithm in the pseudo-code is presented in Algorithm 1. The given code provides the detailed flow of the algorithm step-by-step. At the end of the algorithm, the return is in the form of the particle positioning with the best stability prediction.

4 Experimentation results

4.1 Dataset for experimentation

To test the performance of the proposed algorithm we used the dataset from the open-source platform, which was having the test cases of size 100. The structure of the dataset is composed of several attributes, which were given as follows:

- Test ID
- Test cases
- Pre-Conditions
- Precedence

Algorithm 1: Pseudo-code of the proposed approach

Input: Initialize the population of particles randomly within the search space

Input: Evaluate the fitness of each particle in the population using NEL

Output: Particle with best stability level

Data: Set the global best position and fitness value as the position and fitness of the best particle.

while $iteration - count < Max_{iteration}$ **do**

- Compute the CD of the particle
- Compute the particle's best stability level A_{B_L}
- **for** $i \in [1, n]$ **do**
  - Compute the stability of the ith particle (L_i)
  - Compute the NEL_i of the ith particle
  - **if** $NEL_i > CD$ **then**
    - Compute the SS of the particle
  - **if** $L_i > SS$ **then**
    - Generate α_1 and α_2
    - **for** $j \in [1, \alpha_2]$ **do**
      - compute $A_i^{new_\alpha} = A_i(A_{B_L}(A_i^j))$
    - Generate γ_1 and γ_2
    - Determine neighboring particle (A_N)
    - **for** $j \in [1, \gamma_2]$ **do**
      - compute $A_i^{new_\gamma} = A_i(A_N(A_i^j))$
    - **else if** $L_i < SS$ **then**
      - Find the center of the particle (A_C)
      - compute $A_i^{Update_1} = A_i + \frac{(a.A_{B_L} - b.A_C)}{L_i}$
      - Find the neighboring particle (A_N)
      - compute $A_i^{Update_2} = A_i + (c.A_{B_L} - d.A_N)$
    - **if** $NEL_i < CD$ **then**
      - compute $A_i^{update} = A_i + e$;

Return the particle with stability level (A_{B_L})

- Pass/Fail

The downloaded dataset is in the form of the comma separated file having attribute information. The sample structure of the used dataset is given in Figure 2.

4.2 Parameters value for optimization algorithms

The various optimization algorithms that were used in the experimentation will be having some parameter values that is shown in Table 2.

Table 2: Optimization algorithms used in experimentation with their parameter values

Algorithm	**Parameter 1**	**Value 1**	**Parameter 2**	**Value 2**	**Parameter 3**	**Value 3**	**Parameter 4**	**Value 4**
ACO	Alpha	0.5	Beta	1	Evaporate	0.1	-	-
GA	Population	100	Crossover	0.8	Mutation	0.05	-	-
PSO	Swarm	100	Inertia wt	0.7	Personal Coff.	2.0	Social Coff.	1.5
GSK	Population	100	Knowledge factor	0.5	Knowledge ratio	0.9	Knowledge rate	10
WSO	Moment	0.7	Modality	0.6	Movement	0.3	Search Coff.	0.5
TOA	Team size	100	T factor	0.4	random num	0.2	-	-
CHIO	Population	100	Reproduction rate	0.4	Case age	0.3	-	-

4.3 Result computation

To evaluate the performance of the proposed algorithm here ten benchmark test functions were used. These functions are known as Bukin function, cross-in-tray function, Easom function, Goldstein-price function, Himmelblau's function, Levi function, six-hump camel function, Schaffer's function, Weierstrass function, and Xin-She Yang function. These test functions cover a range of characteristics, such as multimodality, irregularity, and high dimensionality, allowing researchers to assess the performance of metaheuristic algorithms in different problem landscapes.

It's worth noting that each test function has its own mathematical formulation and specific properties, making them suitable for evaluating different aspects of optimization algorithms. Also, each of the test functions is con-

sidered to have a fixed dimension of size 50. The set value of the global best is selected as 0 for all the test functions to make uniformity.

Here, we have utilized 100 partitioned optimization runs to calculate the cruel, the standard deviation, and the vital number of objective work assessments in order to assess the performance. The method utilizes a halting criterion with a tolerance of 1×10^{-12} for the globally optimal values of the alleged problems and a maximum of 50,000 evaluations of objective functions. To evaluate the performance of the proposed algorithm, all the tests were conducted with a fixed population of size 50. The iteration-wise convergence of the proposed algorithm with respect to the time taken is shown in Figure 3. The theoretical convergence of the NEL and the CD with +ve and -ve values on the iteration count of 50 is shown in Figure 4.

The fitness values recorded by using the dataset with the proposed algorithm are given in Table 3. The table shows the result of the calculation of fitness values computed on the basis of per epoch size.

Table 3: Result of the proposed algorithm on the dataset giving fitness values

Iteration count	**Test Case**	**Fitness function**
1	10	5.9488e(-010)
5	18	7.1491e(-010)
10	27	1.7217e(-009)
15	38	4.1091e(-009)
20	46	6.0999e(-009)
25	55	6.1999e(-009)
30	62	8.2628e(-007)
35	70	8.2101e(-007)
40	85	9.998e(-007)
45	93	9.9974e(-007)
50	100	9.998e(-007)

The experimentation result of the proposed algorithm on the selected ten benchmark test functions is presented in Table 4. All the benchmark test functions on the proposed algorithm are evaluated by using the three parameters, that is, mean, standard deviation, and the best value.

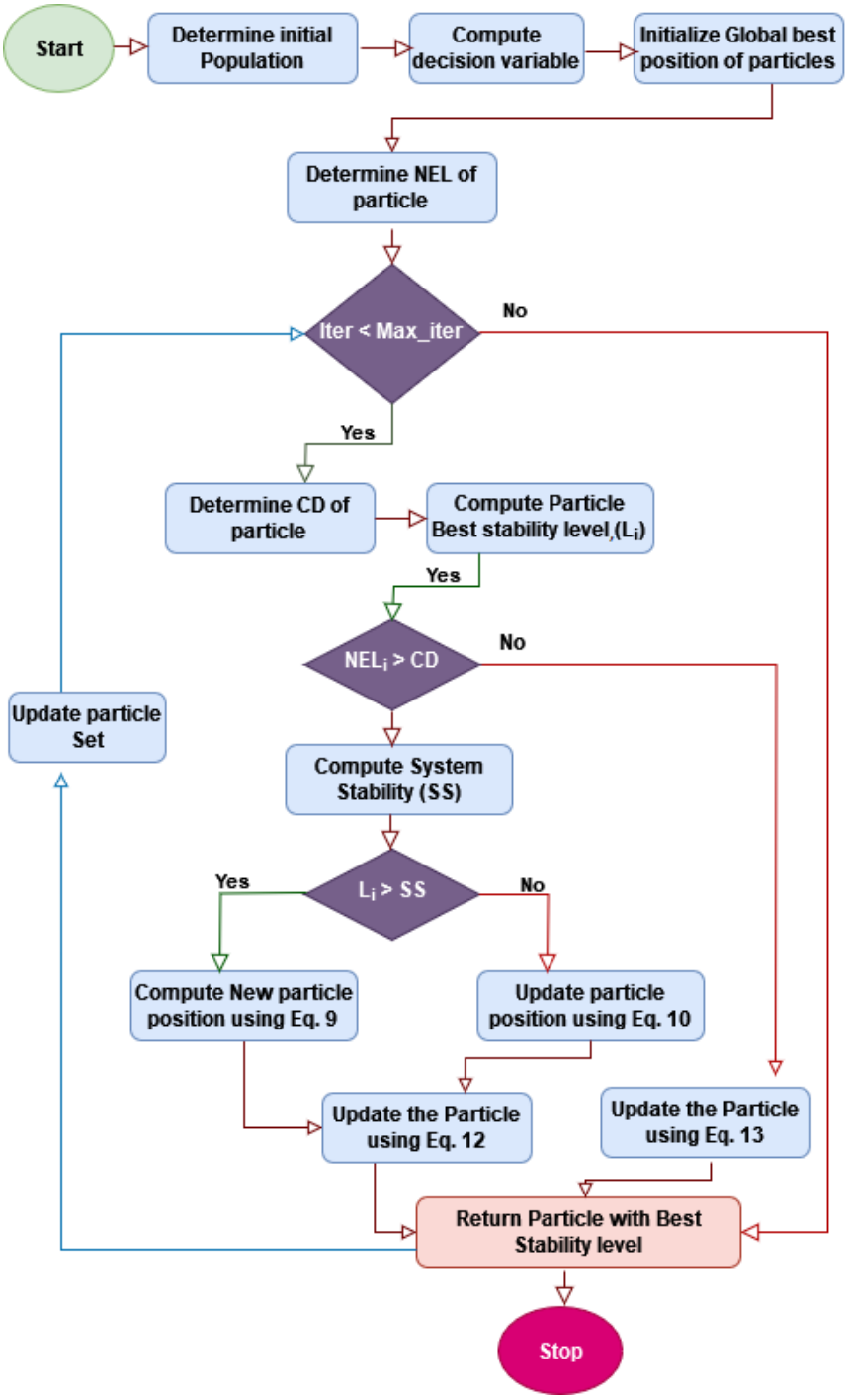


Figure 1: Flowchart of the proposed PPDA algorithm.

	TEST ID	TEST CASE	PRE-CONDITIONS	TEST STEPS	PRECEDENCE	TEST EFFORT	TEST DATA	EXPECTED RESULT	ACTUAL RESULT	PASS/FAIL
0	LOAD_001	Test Page load functionality through URL	None	Enter Invalid URL	H	8	NaN	NaN	NaN	NaN
1	LOAD_002	Test Page Reload without crashes	None	NaN	H	3	NaN	NaN	NaN	NaN
2	LOAD_003	Test 404 Error for Invalid URL in domain	None	NaN	L	7	NaN	NaN	NaN	NaN
3	SELECT_CITY_001	Test City Choice using Icons	LOAD_001	Choose city from name and icons	M	4	NaN	NaN	NaN	NaN
4	SELECT_CITY_002	Test Search City using Search Bar	LOAD_001	Choose city by Search Bar	M	6	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...
95	BOOK_EVENT_003	Test Add & Remove person functionality	BOOK_EVENT_001	NaN	M	6	NaN	NaN	NaN	NaN
96	BOOK_EVENT_004	Test choice of seats	BOOK_EVENT_003	NaN	H	7	NaN	NaN	NaN	NaN
97	EVENT_CONFIRM_001	Test transfer to Payment Init	BOOK_EVENT_004	NaN	H	4	NaN	NaN	NaN	NaN
98	EVENT_CONFIRM_002	Test transfer to Confirmation Page after payment	PAY_PORTAL_003	NaN	H	4	NaN	NaN	NaN	NaN
99	HELP_001	Test help link functionality	HOME_001	NaN	L	5	NaN	NaN	NaN	NaN

Figure 2: Sample description of the dataset used for experimentation.

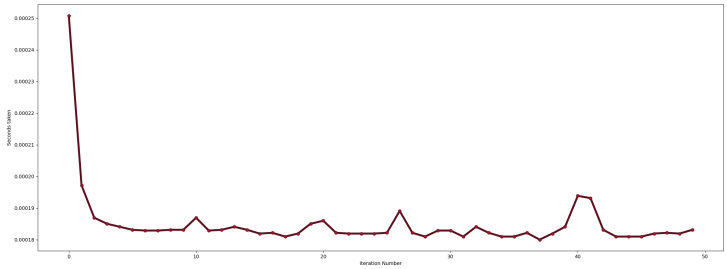
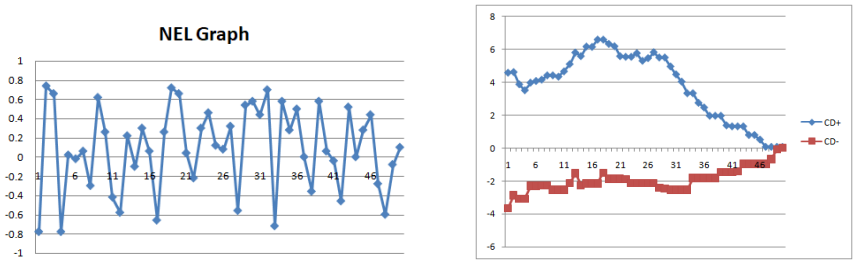


Figure 3: Performance analysis of the proposed algorithm in between the iterations count and time taken by the algorithm.



(a) NEL Curve (b) CD Curve

Figure 4: Convergence of the NEL and CD on the iteration count 50

Table 4: Performance analysis of the proposed algorithm with ten benchmark functions

Benchmark Function	**Parameters**	**Experimented value**
Bukin Function	Mean	0.11958
	Std. Dev	1.15982
	Best	3.24853
Cross-in-Tray Function	Mean	20.8456
	Std. Dev	4.12124
	Best	9.05642
Easom Function	Mean	27.5468
	Std. Dev	11.2619
	Best	8.4653
Goldstein-Price Function	Mean	12.5887
	Std. Dev	3.8911
	Best	4.5997
Himmelblau's Function	Mean	-2.805118
	Std. Dev	3.131312
	Best	3.584428
Levi Function	Mean	13.1087
	Std. Dev	7.7724
	Best	19.4376
Six-Hump Camel Function	Mean	0.0898
	Std. Dev	-0.7126
	Best	0.7126
Schaffer Function	Mean	5.47×10^{-13}
	Std. Dev	5.47×10^{-12}
	Best	0
Weierstrass Function	Mean	1.53×10^{-03}
	Std. Dev	1.41×10^{-03}
	Best	1.18×10^{-06}
Xin-She Yang Function	Mean	1.25×10^{-03}
	Std. Dev	1.12×10^{-03}
	Best	6.96×10^{-06}

5 Comparative analysis

The comparative analysis of the proposed algorithm with the other benchmark metaheuristic algorithms like ACO, GA, and PSO on the ten selected benchmark functions is presented in Table 5. We explored some recent works on the optimization of the problems and selected few named, as gaining-sharing knowledge-based algorithm (GSK) [23], war strategy optimization (WSO) [5], teamwork optimization algorithm (TOA) [13], and Coronavirus herd immunity optimizer (CHIO) [3]. The comparative analysis using some recently proposed metaheuristic algorithms is also presented in Table 5.

Table 5: Comparative analysis of the proposed algorithm with other metaheuristic optimizations on ten benchmark functions

Benchmark Functions	Parameters	Experimented value							
		Proposed Algorithm	ACO	GA	PSO	GSK	WSO	TOA	CHIO
Bukin Function	Mean	0.11958	0.0452	0.0845	0.1045	0.6424	0.7594	0.4645	0.8215
	Std. Dev	1.15982	0.8875	0.8952	0.9973	0.0353	0.5204	0.5329	0.9762
	Best	3.24853	1.2975	2.0128	2.9861	1.5612	1.8345	2.4699	2.9164
Cross-in-Tray Function	Mean	20.8456	12.5465	14.5612	17.7789	11.9246	12.3320	15.4976	18.5564
	Std. Dev	4.12124	2.2498	2.8167	3.4242	1.7783	1.9201	2.1286	2.5546
	Best	9.05642	5.1971	7.1267	8.9462	2.5610	3.8848	5.5585	6.4319
Easom Function	Mean	27.5468	17.5641	20.5731	23.1973	17.2564	21.5645	20.0843	23.4997
	Std. Dev	11.2619	8.8891	10.0245	11.1189	8.1469	10.1213	11.0046	11.1873
	Best	8.4653	5.2307	6.7239	7.8912	4.3469	4.8501	5.4321	6.0197
Goldstein-Price Function	Mean	12.5887	9.1327	10.8123	11.7831	7.8873	8.8213	9.4631	9.9799
	Std. Dev	3.8911	1.0891	1.9921	2.5618	1.2390	2.0049	2.7391	3.7721
	Best	4.5997	2.2080	3.0891	4.1129	1.9951	2.5739	3.0852	3.7894
Himmelblau's Function	Mean	-2.805118	-4.1978	-3.331946	-2.913354	-5.3349	-4.0167	-2.4223	-2.8521
	Std. Dev	3.131312	0.1974	1.079641	2.139734	0.5216	1.0841	1.9898	2.0173
	Best	3.584428	0.1941	1.943137	2.139424	1.0409	1.9744	2.3367	3.0889
Levi Function	Mean	13.1087	9.1346	11.4621	1.7921	7.2252	8.5545	10.1446	9.8878
	Std. Dev	7.7724	4.9134	6.1389	7.5134	4.4464	5.1973	6.5565	5.8552
	Best	19.4376	12.9434	16.4673	18.9124	10.2881	12.4652	17.8446	15.5550
Six-Hump Camel Function	Mean	0.0898	1.1913	0.1843	0.0661	0.9979	0.6732	0.2212	0.1101
	Std. Dev	-0.7126	-1.7349	-0.9943	-0.7112	-1.2333	-1.1249	-0.9999	-0.4156
	Best	0.7126	1.9431	0.9937	0.6574	0.7449	0.7552	0.7449	0.8510
Schaffer's Function	Mean	5.47E-13	9.77E-13	7.87E-13	6.34E-13	9.44E-13	8.14E-13	6.49E-13	5.11E-13
	Std. Dev	5.47E-12	7.71E-12	6.89E-12	6.07E-12	8.11E-12	7.57E-12	6.04E-12	6.01E-12
	Best	0	1.6972	0.9234	0.0569	0.6646	0.7411	1.1012	1.6887
Weierstrass Function	Mean	1.53E-03	3.33E-03	2.89E-03	1.943E-03	2.84E-03	2.41E-03	1.72E-03	1.50E-03
	Std. Dev	1.41E-03	2.61E-03	1.99E-03	1.65E-03	2.43E-03	1.90E-03	1.99E-03	1.57E-03
	Best	1.18E-06	4.81E-06	3.11E-06	2.01E-06	4.71E-06	3.11E-06	1.53E-06	1.96E-06
Xin-She Yang Function	Mean	1.25E-03	2.55E-03	2.012E-03	1.95E-03	1.53E-03	2.84E-03	2.61E-03	1.47E-03
	Std. Dev	1.12E-03	2.56E-03	1.99E-03	1.24E-03	1.55E-03	1.72E-03	1.30E-03	1.24E-03
	Best	6.96E-06	4.75E-06	5.09E-06	6.66E-06	4.21E-06	5.01E-06	5.78E-06	6.11E-06

To test the significance of the proposed model, the statistical analysis is also computed and compared. To do this, the well-known statistical mea-

sure test named Kolmogorov–Smirnov (KS) test is used in this work. The KS test is a nonparametric test that is used to determine if a sample comes from a specific distribution. It is particularly useful for assessing whether a sample follows a particular theoretical distribution, such as a normal distribution. The test is based on the cumulative distribution function of the empirical distribution and the theoretical distribution being tested. The KS test statistic, denoted as D, is the maximum absolute difference between the empirical distribution function of the sample and the cumulative distribution function of the theoretical distribution. If the calculated test statistic is greater than the critical value from the KS distribution table (or the p-value is less than the significance level), then the null hypothesis will be rejected. The experimentation results of the KS test are presented in Table 6. The KS test results are given in Table 6, and since the test's p-value is less than 0.05, the hypothesis regarding the data's normal distribution is satisfied. As a result, nonparametric statistical tests can be used for more research.

Table 6: The KS test values showing the significance of proposed method

Algorithm	**Data type**	**Existing Algorithms**						
		ACO	**GA**	**PSO**	**GSK**	**WSO**	**TOA**	**CHIO**
Proposed Algorithm	Best	8.22E-06	8.22E-06	8.22E-06	6.45E-06	8.22E-06	7.19E-06	7.84E-06
	Mean	1.65E-04	1.93E-04	1.65E-04	1.65E-04	2.01E-04	1.99E-04	1.65E-04
	SD	2.12E-02	2.12E-02	1.96E-02	2.76E-02	2.12E-02	2.12E-02	2.76E-02

6 Conclusion

In conclusion, the proposed metaheuristic algorithm based on the physics informed particle decay principle, specifically for test case minimization, shows promising potential. By leveraging concepts from physics, such as particle stability and different modes of decay, the algorithm offers a unique approach to optimize test case generation and reduce redundancy.

The algorithm's utilization of physics principles allows for the generation of stable particles that represent high-quality test cases, while unstable particles can decay and be replaced by new particles to explore different areas of

the search space. This enables the algorithm to efficiently converge towards optimal or near-optimal solutions.

Through the use of the proposed algorithm, the test case minimization process can benefit from the inherent properties of particles in physics, such as stability and decay. This approach can enhance the efficiency of test case generation by reducing the number of redundant or irrelevant test cases, leading to improved resource utilization and reduced testing time.

Also, the proposed metaheuristic algorithm for test case minimization, validated on ten benchmark functions, has demonstrated its effectiveness and potential in optimizing the generation and selection of test cases. The algorithm has been evaluated and compared against well-known mathematical functions, which serve as standard test cases for assessing the performance of optimization algorithms.

The results of the validation process indicate that the proposed metaheuristic algorithm performs well in terms of minimizing the test case suite while maintaining the required coverage and effectiveness. The algorithm has shown competitive performance in terms of convergence towards optimal or near-optimal solutions, as well as the ability to handle different types of objective functions.

Through the use of metaheuristic techniques, the algorithm has successfully explored the search space and identified high-quality test cases that meet the specified criteria. The algorithm's ability to adapt and evolve over iterations has been instrumental in achieving good performance across the benchmark functions.

The validation of ten benchmark functions has provided insights into the algorithm's strengths and weaknesses. The algorithm's robustness and generalization capabilities have been evaluated by testing its performance on various types of objective functions. The results indicate that the algorithm is capable of effectively minimizing test cases across different problem domains.

Future work will still need to be done in order to evaluate the algorithm on a larger variety of benchmark functions and real-world test situations. The algorithm's performance can also be increased by refining its parameters and taking into account knowledge particular to the situation. The work could

be extended with the design of the nonconvex objective function in place of the convex function. Introducing nonconvexity can make the optimization problem more challenging, as it increases the likelihood of multiple local minima.

Compliance with ethical standards

Conflict of Interest:

The authors of this manuscript declare that there is no conflict of interest.

Data Availability Statement:

The dataset and code generated during and/or analyzed during the current study are available from the corresponding author at a reasonable request.

Funding Information:

The author declares that there is no funding associated with this project.

Ethics Statement:

The author of this manuscript confirms that:

(i) Informed, written consent has been obtained from the relevant sources wherever required;

(ii) All procedures followed were in accordance with the ethical standards of the responsible committee on human experimentation (institutional and national) and with the Helsinki Declaration of 1964 and its later amendments.

(iii) Approval and/or informed consent were not required for the study as the dataset is collected from an open-source website which is freely available.

References

[1] Ahmed, B.S. *Test case minimization approach using fault detection and combinatorial optimization techniques for configuration-aware structural testing*, Eng. Sci. Technol. an Int. J. 19(2) (2016), 737–753.

[2] Akour, M., Abuwardih, L., Alhindawi, N. and Alshboul, A. *Test case minimization using genetic algorithm: pilot study*, In: 2018 8th International Conference on Computer Science and Information Technology (CSIT), 66 –70. IEEE (2018).

[3] Al-Betar, M.A., Alyasseri, Z.A.A., Awadallah, M.A. and Abu Doush, I. *Coronavirus herd immunity optimizer (chio)*, Neural Comput. Appl. 33 (2021), 5011–5042.

[4] Arasteh, B., Gharehchopogh, F.S., Gunes, P., Kiani, F. and Torkamanian Afshar, M. *A novel metaheuristic based method for software mutation test using the discretized and modified forrest optimization algorithm*, J. Electron. Test. (2023), 1–24.

[5] Ayyarao, T.S., Ramakrishna, N., Elavarasan, R.M., Polumahanthi, N., Rambabu, M., Saini, G., Khan, B. and Alatas, B. *War strategy optimization algorithm: a new effective metaheuristic algorithm for global optimization*, IEEE Access 10, (2022), 25073–25105.

[6] Bajaj, A. and Sangwan, O.P. *Discrete and combinatorial gravitational search algorithms for test case prioritization and minimization*, Int. J. Inf. Technol. 13 (2021), 817–823.

[7] Bajaj, A., Sangwan, O.P. and Abraham, A. *Improved novel bat algorithm for test case prioritization and minimization*, Soft Comput. 26(22) (2022), 12393–12419.

[8] Bajaj, A., Abraham, A., Ratnoo, S. and Gabralla, L.A. *Test case prioritization, selection, and reduction using improved quantum-behaved particle swarm optimization*, Sensors 22(12) (2022), 4374.

[9] Bharathi, M. *Hybrid particle swarm and ranked firefly metaheuristic optimization-based software test case minimization*, Int. J. Appl. Metaheuristic Comput. (IJAMC) 13(1) (2022), 1–20.

[10] Bhatia, P.K. *Test case minimization in cots methodology using genetic algorithm: a modified approach*, In: Proceedings of ICETIT 2019: Emerging Trends in Information Technology, 219 –228. Springer, 2020.

[11] Bian, Y., Li, Z., Zhao, R. and Gong, D. *Epistasis based aco for regression test case prioritization*, IEEE Trans. Emerg. Top. Comput. Intell. 1(3) (2017), 213–223.

[12] Boukhlif, M., Hanine, M. and Kharmoum, N. *A decade of intelligent software testing research: A bibliometric analysis*, Electronics 12(9) (2023), 2109.

[13] Dehghani, M. and Trojovsk'y, P. *Teamwork optimization algorithm: A new optimization approach for function minimization/maximization*, Sensors 21(13) (2021), 4567.

[14] Deneke, A., Assefa, B.G. and Mohapatra, S.K. *Test suite minimization using particle swarm optimization*, Mater. Today: Proc. 60 (2022), 229–233.

[15] Geetha, U. and Sankar, S. *Multi-objective modified particle swarm optimization for test suite reduction (mompso)*, Comput. Syst. Sci. Eng. 42(3) (2022), 899–917.

[16] Guizzo, G., Califano, F., Sarro, F., Ferrucci, F. and Harman, M. *Inferring test models from user bug reports using multi-objective search*, Empir. Softw. Eng. 28(4) (2023), 95.

[17] Habib, A.S., Khan, S.U.R. and Felix, E.A. *A systematic review on searchbased test suite reduction: State-of-the-art, taxonomy, and future directions*, IET Software 17(2) (2023), 93–136.

[18] Hashim, N.L. and Dawood, Y.S. *Test case minimization applying firefly algorithm*, Int. J. Adv. Sci. Eng. Inf. Technol. 8(4-2) (2018), 1777–1783.

[19] Joseph, A. and Radhamani, G. *Hybrid test case optimization approach using genetic algorithm with adaptive neuro fuzzy inference system for regression testing*, J. Test. Eval. 45(6) (2017), 2283–2293.

[20] Khatibsyarbini, M., Isa, M.A. and Abang Jawawu, D.N. *A hybrid weight-based and string distances using particle swarm optimization for prioritizing test cases*, J. Theor. Appl. Inf. Technol. 95(12) (2017).

[21] Khoshniat, N., Jamarani, A., Ahmadzadeh, A., Haghi Kashani, M. and Mahdipour, E. *Nature-inspired metaheuristic methods in software testing*, Soft Comput. (2023), 1–42.

[22] Kocher, D.C. *Radioactive decay data tables*, Tech. rep., Oak Ridge National Lab., TN (USA), 1981.

[23] Mohamed, A.W., Hadi, A.A. and Mohamed, A.K. *Gaining-sharing knowledge based algorithm for solving optimization problems: a novel natureinspired algorithm*, Int. J. Mach. Learn. Cybern. 11(7)(2020), 1501–1529.

[24] Mohanty, S., Mohapatra, S.K. and Meko, S.F. *Ant colony optimization (aco-min) algorithm for test suite minimization*, In: Progress in Computing, Analytics and Networking: Proceedings of ICCAN 2019, 55 –63. Springer, 2020.

[25] Nayak, G. and Ray, M. *Modified condition decision coverage criteria for test suite prioritization using particle swarm optimization*, Int. J. Intell. Comput. Cybern. 12(4) (2019), 425–443.

[26] Pachariya, M.K. *Building ant system for multi-faceted test case prioritization: An empirical study*, Int. J. Softw. Innov. (IJSI) 8(2)(2020), 23–37.

[27] Pan, R., Ghaleb, T.A. and Briand, L. *Atm: Black-box test case minimization based on test code similarity and evolutionary search*, 2023

IEEE/ACM 45th International Conference on Software Engineering (ICSE). IEEE, 2023.

[28] Sahin, O. and Akay, B. *Comparisons of metaheuristic algorithms and fitness functions on software test data generation*, Appl. Soft Comput. 49 (2016), 1202–1214.

[29] Sahoo, R.R. and Ray, M. *Pso based test case generation for critical path using improved combined fitness function*, J. King Saud Univ. - Comput. Inf. Sci. 32(4) (2020), 479–490.

[30] Sheikh, R., Babar, M.I., Butt, R., Abdelmaboud, A. and Eisa, T.A.E. *An optimized test case minimization technique using genetic algorithm for regression testing*, Comput. Mater. Contin. 74(3) (2023), 6789–6806.

[31] Sun, J., Chen, J. and Wang, G. *Multi-objective test case prioritization based on epistatic particle swarm optimization*, Int. J. Performability Eng. 14(10) (2018), 2441.

[32] Suri, B. and Singhal, S. *Analyzing test case selection & prioritization using ACO*, ACM SIGSOFT Software Engineering Notes 36(6) (2011), 1–5.

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

[34] Verma, A.S., Choudhary, A. and Tiwari, S. *Regression test case selection: A comparative analysis of metaheuristic algorithms*, In: Proceedings of Second Doctoral Symposium on Computational Intelligence: DoSCI 2021, 605–615. Springer, 2022.