ISSN: 2633-4828 Vol. 5 No.3, September, 2023

International Journal of Applied Engineering & Technology

HGA WITH DIVERSITY MEASURE FOR TEST CASE PRIORITIZATION

Shrankhla Saxena*[1], Javed Wasim[2], Abhinav Juneja[3] and Rashi Gupta[4]
[1,2]FET, Department of Computer Engineering & Application, Mangalayatan University, Aligarh, India.
[3]Computer Science & Information Technolgy, KIET Group of Institutions, Ghaziabad, India.
[4]Research Scholar, Deparment. of Computer Engineering & Applications, Faculty of Engineering & Technology, Mangalayatan University, Aligarh, India.

ABSTRACT

Regression testing is utmost important and crucial phase of software development process as it tests the validity of software after making changes in some parts of software. There are various ways to perform regression testing. Due to its high cost and long-time lot of research is done to automate the process of regression testing. To reduce the cost and time of regression testing, automation of test case prioritization (TCP) is required. Various machine learning (ML) techniques have been proposed by researchers to automate regression testing so that maximum issues can be uncovered at initial stage of testing. This can significantly improve the cost, time, and efforts of regression testing. Many ML techniques like greedy algorithms, NSGA-II, genetic algorithms, and others are proposed by various authors and researchers for prioritization of test cases in single and multi-objective optimization scenarios. In our previous work we have proposed the use of hypervolume based genetic algorithm (HGA) with diversity measure for test case prioritization in regression testing. We have presented the evaluation of proposed algorithm also. In this paper, we will show the detailed comparison of HGA including diversity measure with other single objective and multi objective optimization methods of regression testing. This paper will give better understanding for the selection of different ML approaches in different scenario of test case prioritization.

Keywords: Hypervolume, diversity measure, genetic algorithm, regression testing.

1. INTRODUCTION

Software development can not be completed without regression testing. The purpose of regression testing is to ensure that recent modifications in the code does not affect the existing features of software adversely. It involves re-running tests that have already been executed to verify that existing functionalities continue to work correctly after a code change, addition, or fix [1]. Overall, regression testing helps maintain the overall quality and stability of software product by ensuring that new developments do not inadvertently introduce problems into existing functionalities. Regression testing can be performed through three different ways which are retest all, selective regression testing and priority-based regression testing. In first method that is retest all, all existing test cases are re-executed to ensure the comprehensive coverage. Hence retest all method of regression testing consumes more resources and time especially for large applications. In the second method of regression testing i.e. selective regression testing, only those parts of application are required to be retested which have the impact of recent code change, therefore only selected test cases are re-executed. This method is efficient in terms of time and resources but needs careful identification of affected areas of software [2]. In the third method i.e. test case prioritization, test cases are prioritized depending on their importance or criticality of the software system. Higher priority test cases are executed first to ensure essential functionalities are not affected by the modifications in the software.

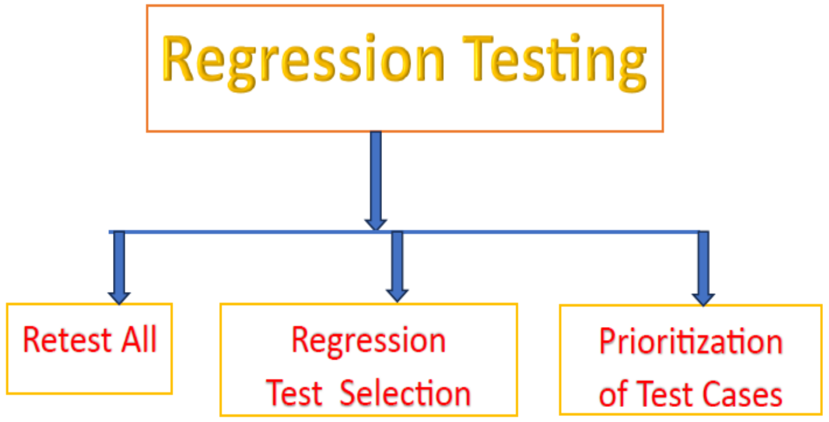

Figure 1. Approaches of regression testing

Various approaches are available to automate the prioritization process for test cases in regression testing such as greedy algorithm, genetic algorithm, meta heuristic approach etc. Di Nucci et al. [3] proposed the use of hypervolume indicator with genetic algorithm for prioritizing the test cases.

2. PROPOSED METHODOLOGY

We propose the application of HGA in test case prioritization. Test case prioritization in regression testing is a crucial process that aims to order test cases in a manner that maximizes the chances of detecting faults early, minimizing the time and resources needed for testing. Traditional methods of TCP often suffers with the problem of optimizing multiple conflicting objectives, such as improving fault detection rate and reducing execution time. This is where a hypervolume-based genetic algorithm (HGA) proves to be a powerful tool [4].

a. Genetic Algorithm in Test Case Prioritization

Genetic Algorithms (GAs) are search heuristics that is based on the process of natural law of evolution to solve optimization problems. In the context of prioritizing the test cases, a GA can evolve a population of test case orderings to maximize specific objectives, such as code coverage or fault detection rate. The algorithm iteratively selects the best individuals (test case orderings), applies crossover and mutation operations, and creates new generations of solutions that are closer to the optimal solution [5].

b. Hypervolume as a Performance Metric

The role of hypervolume metric in multi-objective method of optimization is measuring the performance of solutions set by calculating the volume of the objective space affected by the solutions In test case prioritization, multiple objectives such as fault revealing rate, code coverage, and time must be optimized simultaneously. The hypervolume provides a scalar measure that captures the trade-offs between these objectives, giving a comprehensive assessment of the solution set's quality [6].

c. Hypervolume-Based Genetic Algorithm (HGA)

The HGA utilizes the hypervolume metric to direct the search process in genetic algorithms. Instead of relying on traditional fitness functions that which focus on a single objective, HGA uses hypervolume as a measure to evaluate and rank solutions based on their performance across multiple objectives [7][8]. Figure 2 represents the detailed procedure of hypervolume based genetic algorithm applied in test case prioritization. This allows HGA to balance trade-offs effectively and find a diverse set of high-quality solutions.

1. **Initialization:** The algorithm begins with a randomly generated population of test case orderings.

$P_0 = x_1, x_2, \ldots\ldots, x_N$ be the initial population of N individuals.

Each individual x_i is a vector representing a specific ordering of test cases , initialized randomly.

Mathematically:

$P_0 = random\_initialize(N)$

2. **Fitness Evaluation:** Each individual (test case ordering) is evaluated based on multiple objectives, such as fault revealing rate, execution time, and code coverage.

$f_1(x_i): fault\ revealing\ rate$

$f_2(x_i): execution\ time$

$f_3(x_i): code\ coverage$

The objective vector f(x_i) for an individual xi can be represented as:

$$f(x_i) = \begin{bmatrix} f1(x_i) \\ f2(x_i) \\ f3(x_i) \end{bmatrix}$$

3. **Hypervolume Calculation:** The hypervolume metric is computed for each individual, quantifying how well it performs across all objectives compared to a reference point. The hypervolume HV(x_i) measure the volume of the objective space dominated by an individual x_i relative to reference point r. For each individual:

$$HV(x_i) = \int_{x_i \leq y \leq r} dV$$

Where $x_i \leq y$ indicates y is dominated by xi (better or equal in all objectives, and dV represents the differential volume element.

4. **Selection:** Individuals with higher hypervolumes are selected for reproduction, ensuring that solutions that perform well across multiple objectives are favoured. For a subset S⊂P of the population, the probability of selecting an individual x_i is:

$$P(x_i) = \frac{\mathrm{HV}(x_i)}{\sum_{x_j \in S} HV(x_j)}$$

5. **Crossover and Mutation:** Crossover and mutation are performed on selected orderings with the aim of producing new offspring, introducing diversity and exploring new potential solutions. Given two parent individuals x_a and x_b , offspring $x_{offspring}$ is generated as:

$x_{offspring} = crossover(x_a, x_b)$

Mutation is applied to the offspring, modifying the ordering (e.g., by swapping two elements):

$x'_{offspring} = mutation(x_{offspring})$

6. **Iteration:** The process is repeated over multiple generations until convergence or a stopping criterion is met, continually improving the population of test case orderings.

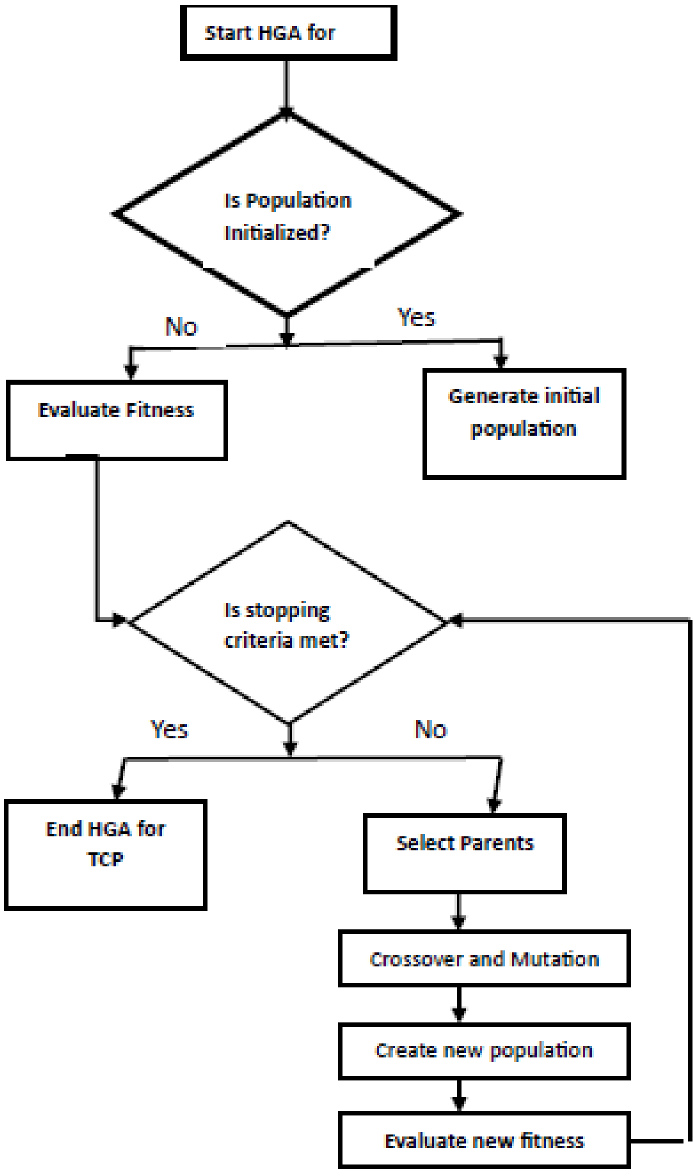

Figure 2. Process of HGA for TCP

d. Benefits of HGA with Diversity Measure in Test Case Prioritization

- **Multi-Objective Optimization:** HGA efficiently handles multiple conflicting objectives, ensuring a balanced prioritization of test cases.
- **Diversity Preservation:** By focusing on hypervolume, HGA maintains a diverse set of solutions, reducing the risk of premature convergence to suboptimal solutions.
- **Better Trade-off Management:** HGA provides a more holistic view of the solution space, allowing for better trade-offs between competing objectives, such as fault detection and execution time.
- **Flexibility:** It offers flexibility in adjusting objective weights or adding/removing objectives, making it adaptable to changing testing priorities or requirements.

- **Robustness:** HGA employs evolutionary mechanisms that provide robustness against getting stuck in local optima, allowing for the exploration of diverse solutions.

e. Limitations of HGA with diversity measure in Test Case Prioritization

- **Computational Intensity:** The algorithm can be computationally intensive, particularly for large-scale problems, due to the need to maintain diversity and optimize multiple objectives simultaneously.
- **Complexity:** Handling multiple objectives and managing solution diversity increases the complexity of implementation and requires expertise in multi-objective optimization techniques.
- **Parameter Tuning:** Balancing convergence and diversity requires careful parameter tuning, which can be challenging and impact the effectiveness of the algorithm.
- **Interpretation Challenges:** Evaluating results based on hypervolume metrics and interpreting the significance of solutions can be complex, requiring a thorough understanding of the optimization process.

Hence, while HGA with diversity measures offers significant advantages in optimizing test case prioritization for regression testing, such as comprehensive coverage and flexibility, it also presents challenges related to computational complexity, parameter tuning, and result interpretation. These factors should be carefully considered while choosing HGA for test case prioritization in practice.

3. HGA VS SINGLE OBJECTIVE APPROACHES FOR TCP

There are various single objective ML methods for prioritizing the test cases. This section gives the overview of few of such techniques.

a. Greedy Algorithm:

Objective: Prioritizes test cases based on a single criterion (e.g., code coverage or fault detection rate).

Diversity: Typically lacks diversity since it focuses solely on optimizing one metric.

Effectiveness: Can be effective for maximizing a specific criterion but may miss out on covering diverse scenarios or edge cases [9].

Applicability: Simple and computationally efficient but may not provide the best balance between coverage and risk prioritization.

This algorithm proceeds by selecting test cases with the highest value, one by one, until all test cases are ordered. At each step, it selects the test case that provides the maximum additional value.

The greedy algorithm equation for prioritizing test cases in a set T (where T= {T_1, T_2, …., T_n}) can be written as follows:

$$\text{Select } T_i \in T \text{ such that } \max(V(T_i))$$

This selection process is repeated by removing T_i from T each time a test case is chosen until T is empty.

b. Weighted Sum Model:

Objective: Combines multiple criteria into a single objective function using weighted coefficients.

Diversity: Depends on how weights are assigned; tends to prioritize the criterion with the highest weight, potentially sacrificing diversity.

Effectiveness: Provides flexibility in balancing multiple objectives but may not handle trade-offs between conflicting objectives as effectively as multi-objective approaches.

Applicability: Suitable for scenarios where clear priorities can be defined but may overlook diverse coverage requirements.

Given a set of test cases T={T$_1$, T$_2$, …. , T$_n$} and criteria C$_1$, C$_2$, …. , C$_m$ (such as code coverage, historian fault detection, and execution time), with corresponding weights w$_1$, w$_2$, …. , w$_m$ (where $\sum_{j=1}^{m} w_j = 1$), the weighted sum score S(T$_i$) for each test case T$_i$ is calculated as:

$$S(T_i) = \sum_{j=1}^{m} w_j . C_j(T_i)$$

Where:

- $S(T_i)$ is weighted sum score for test case T_i.
- $C_j(T_i)$ is the score of test case T_i for criteria C_j.
- w_j is the weight assigned to test criteria C_j.

Each criteria C_j should be normalized to ensure comparability, especially if criteria are measured on different scales.

c. Genetic Algorithm (Single Objective):

Objective: Uses genetic operators (selection, crossover, mutation) to optimize a single fitness function (e.g., code coverage).

$$f(x) = w_1 . Coverage(x) + w_2 . Faultdetection(x) - w_3 . ExecutionTime(x)$$

Where:

- w$_1$, w$_2$, w$_3$ are weights to balance different criteria based on their importance.
- Coverage is the percentage of code covered by executing the test sequence x.
- Faultdetection measures the number of faults detected by executing x.
- ExecutionTime is the total time taken to run x, subtracted to represent minimization.

Diversity: Like the greedy algorithm, tends to converge towards a subset of test cases that optimize the single objective, potentially lacking diversity [10].

Effectiveness: Efficient for optimizing a specific criterion but may not address the need for comprehensive coverage across various scenarios.

Applicability: Useful when the primary goal is to maximize a single metric and computational resources are limited.

In a single objective genetic algorithm, the goal is to either maximize or minimize f(x) of the best test case sequence. For maximizing fault detection and code coverage, the objective would be:

$$x_{best} = arg \max_{x} f(x)$$

Or, for minimizing for example execution time, the objective would be:

$$x_{best} = arg \min_{x} f(x)$$

Comparison between hypervolume based genetic algorithm with diversity measure and single objective TCP methods is presented in table 1. Hypervolume-Based Genetic Algorithms (HGA) with diversity measures offer significant advantages over traditional single-objective techniques for test case prioritization in regression testing scenarios. While single-objective approaches like greedy algorithms and weighted sum models are simpler and computationally efficient, they often sacrifice diversity and may not provide optimal coverage across diverse scenarios [11]. HGA, on the other hand, excels in maintaining diversity and optimizing multiple objectives simultaneously, making it well-suited for complex regression testing environments where comprehensive coverage and risk management are critical. HGA can adapt to changes in software and testing priorities by optimizing multiple criteria simultaneously, ensuring that critical functionalities and edge cases are covered. By maintaining diversity, HGA reduces the risk of missing critical defects or scenarios that could impact software reliability [12]. Therefore, the choice between these approaches should consider the specific needs of the testing process, balancing simplicity, and computational efficiency with the requirements for diverse and effective test case prioritization.

4. HGA VS MULTI-OBJECTIVE TECHNIQUES

There are several multi-objective techniques used for test case prioritization in regression testing. Here are some of the commonly used techniques:

a. Pareto Optimization: In multi-objective optimization, Pareto efficiency (or Pareto optimality) is the state where no objective can be improved without worsening at least one other objective. A Pareto front indicates the set of non-affected solutions where no single test prioritization dominates another in all objectives. Test cases can be prioritized by selecting those closest to the Pareto front. [3]. This approach seeks to find a set of solutions that are non-dominated, meaning no other solution is better in all objectives. Each test case is evaluated on multiple objectives, and the goal is to select test cases that form a Pareto front.

Let:

- $f_1(x)$: Fault detection capability of a test sequence x (to be maximized).
- $f_2(x)$: Coverage of a test sequence x (to be maximized).
- $f_3(x)$: Execution time of a test sequence x (to be minimized).

Thus, for pareto optimization, the aim is to optimize all three objectives simultaneously:

$max\ f_1(x)$, max $f_2(x)$, min $f_3(x)$

In Pareto optimization, a solution x pareto dominates another solution y if:

1. $f_i(x) \geq f_i(y)$ for all i (for example for fault detection, coverage, and execution time).
2. $f_j(x) > f_j(y)$ for at least one j.

The goal is to find a pareto front, which is the set of all non-dominated solutions. For test case prioritization, this front represents the best possible trade-offs between the objectives. The pareto optimal set X^* consist of solutions that represent the best trade-offs:

$X^* = \{x | \nexists y \text{ such that } y \text{ pareto dominates } x\}$

The final objective of pareto optimization for test case prioritization is to identify X^* that provides balance among the objectives without any individual solution dominating all others.

In summary, pareto optimization technique seeks to find:

$X^* = \{x \in Population | \nexists y \in Population \text{ such that } y \text{ Pareto dominates } x\}$

b. Multi-Objective Genetic Algorithms (MOGAs): Applies genetic algorithms to optimize multiple objectives simultaneously, balancing trade-offs between conflicting criteria such as coverage, fault detection rate, and execution time [7]. A multi-objective GA (e.g., NSGA-II or SPEA2) can be used to generate solutions that optimize multiple objectives, such as code coverage, fault revelation, and execution cost, while maintaining diversity in the solution space.

Let the set of test cases be $T = \{t_1, t_2, \ldots\ldots, t_n\}$, and let P represents a permutation of test cases (a prioritized order). Then the MOGA can be formulated as:

$$Maximize / Minimize\ \mathbf{F}(P) = \{f_1(P), f_2(P), \ldots, f_m(P)\}$$

Subject to:

$P \in T$, where T is the set of all possible permutations of T.

Where:

- $f_1(P)$: Total execution time of the prioritized test suite.
- $f_2(P)$: Fault detection capability of the prioritized test suite (for example cumulative faults detected).
- $f_3(P)$: Requirement coverage metric.

c. Multi-Objective Particle Swarm Optimization (MOPSO): MOPSO algorithms simulate the social behavior of particles (test cases) by flying through the solution space to find the optimal solution. This approach can balance multiple objectives such as minimizing test execution time and maximizing fault finding with a large search space. It uses particle swarm optimization to find solutions that optimize multiple objectives, adjusting particle positions based on their personal best and global best solutions [1].

Let the test suite be T= {t_1, t_2, ….., t_n}, and let P = {p_1, p_2, ….., p_n} represent a prioritized order of test cases. Then the objective functions are defined as:

$$Maximize / Minimize\ \mathbf{F}(P) = \{f_1(P), f_2(P), \ldots\ldots, f_m(P)\}$$

Where:

- $f_1(P)$: Execution time (to be minimized)
- $f_2(P)$: Fault detection rate (to be maximized)
- $f_3(P)$: Requirement coverage (to be maximized).

In MOPSO, each particle i represents a solution P^i(a prioritized order of test cases), and it moves through the search space using the following equations:

Velocity update:

$$v^i(t+1) = w.v^i(t) + c_1.r_1.(p^i_{best} - P^i) + c_2.r_2.(g_{best} - P^i)$$

Where:

- w: Inertia weight controlling exploration and exploitation.
- c_1, c_2: Acceleration coefficients for personal and global best influence.
- r_1, r_2: Random values uniformly distributed in [0,1].
- p^i_{best}: The personal best position of particle i.
- g_{best}: The global best position.

Position update:

$$P^i(t+1) = P^i(t) + v^i(t+1)$$

d. Multi-Objective Simulated Annealing (MOSA): Adapts the simulated annealing algorithm to optimize multiple objectives by exploring the solution space and gradually reducing the temperature to discover the Pareto optimal solutions set. Simulated Annealing is another heuristic method that seeks a global optimum by accepting worse solutions with a probability that decreases over time. In multi-objective test case prioritization, it can be applied to balance objectives like cost and effectiveness in finding faults.

Simulated annealing components:

I. Objective functions:

- Execution time

$$f_1(P) = \sum_{i=1}^{n} w_i \,.\, execution\_time(p_i)$$

Where w_i is the weight based on priority position of p_i.

- Fault detection rate

$$f_2(P) = \sum_{i=1}^{n} \frac{faults\_detected(p_i)}{execution\_time(p_i)}$$

- Requirement coverage

$$f_3(P) = \sum_{i=1}^{n} requirement\_coverage(p_i)$$

I. Initial solution: Start with a random test case ordering P_0.

II. Neighbour solution: Generate a new solution P' by slightly modifying P, such as swapping the order of two test cases.

III. Energy difference (ΔEk): The energy difference for objective k between solutions P and P' is calculated as:

$$\Delta E_k = f_k(P') - f_k(P)$$

I. Acceptance probability: To decide whether to accept P' , acceptance probability is calculated:

$$P_{accept} = \begin{cases} 1, if\ \Delta E_k \geq 0\ (better\ solution) \\ e^{\frac{\Delta E_k}{T}}, if\ \Delta E_k < 0\ (worse\ solution) \end{cases}$$

Where:

- T : current temperature, gradually decreased over iterations.

II. Pareto dominance: A solution P' dominates P if:

$$\forall k, f_k(P') \geq f_k(P) and\ \exists k, f_k(P') > f_k(P)$$

III. Cooling schedule: Update the temperature T using a cooling schedule:

$$T_{new} = \alpha . T_{current}$$

Where α is the cooling rate.

e. Preference based Evolutionary Algorithm for Multi-Objective Optimization (PEAMO): Incorporates preferences or constraints into evolutionary algorithms to direct the search for solutions that satisfy specific objectives or criteria.

The PEAMO for test case prioritization aims to find P* (an optimal or near optimal permutation) that optimizes the following:

$$\min F(P) = \langle f_1(P), -f_2(P), -f_3(P)\rangle$$

Where:

- $f_1(P) = \sum_{i=1}^{n} exec\_time(p_i)$
- $f_2(P) = \sum_{i=1}^{n} fault\_detect(p_i).w_i$
- $f_3(P)$ evaluates how well P aligns with user preferences, often defined as a weighted satisfaction score.

For preference incorporation, PEAMO modifies the dominance relationship based on stakeholder preferences using a weighted aggregation method or reference-point based approach.

$$g(P) = \sum_{j=1}^{3} w_j f_j(P)$$

Where $w = \langle w_1, w_2, w_3\rangle$ represents user-assigned weights for the objectives.

The PEAMO framework optimizes:

$$P^* = \arg\min_{P\in \mathcal{P}} F(P)$$

Subject to:

$$F(P) = \langle f_1(P), -f_2(P), -f_3(P)\rangle, w.F(P) \le \tau$$

Where Ƥ is the search space of all permutations, w are user-defined weights, and τ is a threshold for preference satisfaction.

f. Non-dominated Sorting Genetic Algorithm (NSGA-II): This algorithm sorts solutions into non-dominated fronts and uses genetic operators to evolve towards Pareto optimal solutions [3][10]. NSGA-II is one of the most widely used evolutionary algorithms for many-objective optimization. It maintains a balance between exploration (finding diverse solutions) and exploitation (refining known solutions) by sorting test cases into different non-dominated levels based on multiple objectives.

The optimization problem solved by NSGA-II for test case prioritization can be expressed as:

$$P^* = \arg\min_{P\in \mathcal{P}} F(P) = \arg\min_{P\in \mathcal{P}} \langle f_1(P), -f_2(P)\rangle$$

Where:

- Ƥ is the set of all permutations of *T*.
- P* is the set of non-dominated pareto optimal solutions.

g. Multi-Objective Ant Colony Optimization (MOACO): Adapts ant colony optimization techniques to handle multiple objectives, utilizing pheromone trails to guide ants towards solutions that optimize diverse criteria. In this approach, artificial ants simulate the process of searching for optimal paths (test case orderings). MOACO allows prioritizing test cases by optimizing criteria like fault revelation and execution time.

Let T={t_1,t_2,….,t_n} represent the set of n test cases. The goal is to prioritize these test cases.

$P = \langle p_1, p_2, \ldots .., p_n \rangle$ to optimize:

1. Minimizing total execution time(f$_1$):

$$f_1(P) = \sum_{i=1}^{n} exec\_time(p_i)$$

2. Maximizing fault detection rate(f$_2$):

$$f_2(P) = \sum_{i=1}^{n} fault\_detect(p_i)\ .w_i$$

Where w_i represents the weight emphasizing early fault detection.

MOACO Framework:

I. Pheromone Representation:
It defines a pheromone matrix τ where $\tau_{i,j}$ represents the pheromone intensity between test case t_i and t_j.

II. Heuristic Information:
It define heuristic information $\eta_{i,j}$ to guide ants:

$$\eta_{i,j} = \frac{fault\_detect(t_j)}{exec\_time(t_j)}$$

This combines fault detection capability and execution efficiency.

I. Ant Movement Probability:
An ant at node i selects the next node j based on a probability:

$$P_{i,j} = \frac{\tau_{i,j}^{\alpha} \cdot \eta_{i,j}^{\beta}}{\sum_{k \in allowed} \tau_{i,j}^{\alpha} \cdot \eta_{i,j}^{\beta}}$$

Where:

- α controls the influence of pheromone ($\tau_{i,j}$).
- β controls the influence of heuristic information $\eta_{i,j}$.

I. Solution Construction:
Each ant builds a solution P (a sequence of test cases) iteratively by choosing nodes based on $P_{i,j}$.

II. Objective Evaluation:
Each solution P is evaluated using the objective vector:

$$F(P) = \langle f(P_1), -f(P_2) \rangle$$

III. Non-dominated Sorting:
Non-dominated sorting is used to classify solutions into pareto fronts:

- A solution P_a dominates P_b($P_a < P_b$) if:

$$f_k(P_a) \leq f_k(P_b) \quad \forall k \text{ and } \exists k \text{ such that } f_k(P_a < f_k(P_b)$$

IV. Pheromone Update:
Pheromone trails are updated based on the quality of the solutions:

$$\tau_{i,j} = (1-\rho).\tau_{i,j} + \sum_{Pareto-optimal\, P} \Delta\tau_{i,j}(P)$$

Where:

- ρ is evaporation rate
- $\Delta\tau_{i,j}(P)$ is contribution of solution P, calculated as:

$$\Delta\tau_{i,j}(P) = \begin{cases} Q/f_1(P) & if (i,j) \in P \\ 0 & Otherwise \end{cases}$$

Here Q is constant.

V. Termination:
Repeat until a termination condition is met (like max iterations or convergence).

h. Multi-Objective Differential Evolution (MODE): Uses differential evolution algorithms to optimize multiple objectives by exploring the solution space through mutation, crossover, and selection mechanisms.

These techniques vary in their approach and complexity but share the common goal of finding a set of solutions that balance multiple objectives effectively in regression testing scenarios. The choice of technique depends on factors such as the nature of objectives, computational resources, and the specific requirements of the software under test.

I. Population Initialization:
A population of candidate solutions is initialized, where each individual represents a potential prioritization of test cases:

$$X_i^t = \{x_{i1}^t, x_{i2}^t, \ldots., x_{iD}^t\},\ i = 1,2, \ldots., NP$$

Where:

- X_i^t is the i-th individual in generation t.
- D is the dimension of solution (number of test cases).
- NP is the population size.

II. Mutation:
For each individual X_i^t, a mutant vector V_i^{t+1} is generated:

$$V_i^{t+1} = X_{r1}^t.F(X_{r2}^t - X_{r3}^t)$$

Where:

- $X_{r1}^t, X_{r2}^t, X_{r3}^t$ are distinct random values from the population (r1≠r2≠r3≠i).
- $F \in [0,2]$ is the mutation scaling factor.

III. Crossover:
A trial vector U_i^{t+1} is generated the by combining the mutant vector V_i^{t+1} and the target vector X_i^t:

$$U_{ij}^{t+1} = \begin{cases} V_{ij}^{t+1} & if\, rand_j \le CR\, or\, j = j_{rand} \\ X_{ij}^t & Otherwise \end{cases}$$

Where:

- CR is crossover probability ($0 \leq CR \leq 1$).
- $rand_j$ is random number in [0,1].
- j_{rand} ensures at least one parameter is taken from V_i^{t+1}.

IV. Selection:

The next generation vector is chosen based on the pareto dominance of objectives:

$$X_i^{t+1} = \begin{cases} U_i^{t+1} & if\ U_i^{t+1}\ dominates\ X_i^t \\ X_i^t & Otherwise \end{cases}$$

Solution A dominates B if :

$$\forall m, f_m(A) \leq f_m(B)\ and\ \exists m, f_m(A) < f_m(B)$$

Where f_m are the objective functions, such as:

Fault detection rate(maximize):

$$f_1(X) = \sum_{i=1}^{D} FDR(x_i)/D$$

Execution time(minimize):

$$f_2(X) = \sum_{i=1}^{D} ET(x_i)$$

Code coverage(maximize):

$$f_3(X) = \sum_{i=1}^{D} CC(x_i)/D$$

V. Pareto Front:

At the end of optimization, the non dominated solutions are stored in pareto front.

$$PF = \{X_i \mid \nexists X_j\ such\ that\ X_j\ dominates\ X_i\}$$

The choice between the multi objective approaches and HGA with diversity factor depends on factors such as the complexity of objectives, computational resources, and the specific requirements of the testing environment. The detailed comparison is given in table 2.

5. RESULT ANALYSIS

Table 1. Comparison between HGA and single objective ML approaches for TCP

Criteria	Single-Objective Approaches	Hypervolume-Based Genetic Algorithm (HGA)
Objective	Optimizes a single criterion (e.g., coverage, fault detection rate)	Optimizes multiple objectives simultaneously using hypervolume as a measure of solution quality
Diversity	Typically lacks diversity, focuses	Actively maintains diversity within the

	on optimizing one metric	population, ensuring coverage across various scenarios and edge cases
Optimization Strategy	Uses algorithms like Greedy, Weighted Sum Model, or single-objective Genetic Algorithms	Uses multi-objective optimization techniques to balance conflicting objectives effectively
Effectiveness	Efficient for maximizing a specific criterion but may miss diverse scenarios	Balances trade-offs between multiple objectives, leading to a more robust test suite
Risk Management	May overlook critical scenarios due to lack of diversity	Reduces risk by covering diverse scenarios and critical functionalities simultaneously
Flexibility	Limited flexibility in adapting to changing testing priorities	Flexible in adjusting to changes by modifying objective weights or adding/removing objectives
Applicability	Simple and computationally efficient	Particularly effective in regression testing for comprehensive coverage and risk prioritization

Table 2. Comparison between HGA and multi objective ML approaches for TCP

Criteria	**Multi-Objective Test Case Prioritization Approaches**	**Hypervolume-Based Genetic Algorithm**
Objective	Optimizes multiple objectives simultaneously (e.g., coverage, fault detection rate, execution time)	Optimizes multiple objectives simultaneously using hypervolume as a measure of solution quality
Diversity Management	Includes mechanisms to maintain diversity within the solution set	Actively maintains diversity within the population of solutions
Optimization Strategy	Uses Pareto optimization techniques to generate a Pareto front	Focuses on improving the hypervolume—a measure of solution quality
Trade-offs	Seeks trade-offs between conflicting objectives	Balances trade-offs between objectives effectively
Computational Complexity	Can be computationally intensive due to handling multiple objectives	May require substantial computational resources due to hypervolume maintenance
Effectiveness in Coverage	Provides comprehensive coverage by balancing multiple criteria	Reduces the risk of missing critical scenarios or defects
Flexibility	Allows adjustments in objective weights to adapt to changing priorities	Offers flexibility in adjusting objective weights or adding/removing objectives
Applicability	Suitable for scenarios requiring comprehensive coverage and trade-off analysis	Particularly effective in regression testing and complex optimization scenarios

6. CONCLUSION

This paper has discussed every aspect of applying hypervolume based genetic algorithm with diversity measure for test case prioritization in regression testing in much detail. This paper also presented the comparison of HGA with other single-objective and multi-objectives ML approaches for prioritization of test cases in regression testing. Hence, this paper concludes that hypervolume based genetic algorithm with the use of diversity measure is better than many single-objective ML techniques for TCP in terms of effectiveness, risk management, optimization strategy and many more factors. HGA with diversity measure is more effective than many multi-objectives ML methods of TCP as it maintains the diversity of population and unlike many-objectives ML methods, HGA focuses on improving the hypervolume for quality solution instead of generating the pareto front for the solution. HGA is also better in coverage and risk management.

We propose that HGA should be compared with some hybrid approaches of machine learning for test case prioritization as the future work. Also, HGA should be studied and applied for other methods of regression testing also like test case selection.

REFERENCES

[1] S. Saxena., J. Wasim., A. Juneja., R. Gupta. 2022. "Test case prioritization based on hypervolume indicator with diversity measure". In Neuroquantology, vol. 20, no. 22, 5407-5419. doi: 10.48047/nq.2022.20.22.NQ10556.

[2] S. Yoo., M. Harman. 2012. "Regression testing minimization, selection and prioritization: a survey". In Software testing, verification and reliability, Vol. 22, 67-120. doi: https://doi.org/10.1002/stv.430.

[3] D. Di Nucci., A. Panichella., A. Zaidman., A. De Lucia. 2020. "A test case prioritization genetic algorithm guided by the hypervolume indicator". In IEEE transactions on software engineering, vol. 46, no. 6, pp. 674-696, doi: 10.1109/TSE.2018.2868082.

[4] A. Arrieta. 2022. "Is the revisited hypervolume an appropriate quality indicator to evaluate multi-objective test case selection algorithms?" In proceedings of the genetic and evolutionary computation conference (GECCO 22). Association for computing machinery, New York, NY, USA, 1317-1326. https://doi.org/10.1145/3512290.3528717.

[5] R. Malhotra., A. Bhardwaj. 2014. "Test case prioritization using genetic algorithm". In International journal of computer science & informatics. 151-154. doi: 10.47893/IJCSI.2014.1136.

[6] A. Auger., J. Bader., D. Brockhoff., E. Zitzler. 2009. "Theory of the hypervolume indicator: optimal μ-distributions and the choice of the reference point". In proceedings of the tenth ACM SIGEVO workshop on foundations of genetic algorithms (FOGA 09). Association for computing machinery, New York, NY, USA, 87-102. doi: https://doi.org/10.1145/1527125.1527138.

[7] D. Di Nucci., A. Panichella., A. Zaidman., A. De Lucia. 2015. "Hypervolume-based search for test case prioritization. In M. Barros., Y. Labiche. (eds) search-based software engineering. SSBSE 2015. Lecture notes in computer science(). vol. 9275. Springer. Cham. doi: https://doi.org/10.1007/978-3-319-22183-0_11.

[8] C. Wang., H. Pan., Y. Su. 2019. "A many-objective evolutionary algorithm with diversity-first based environmental selection". In Swarm and evolutionary computation. vol. 53. doi: 10.1016/j.swevo.2019.100641.

[9] A. Kaur., S. Goyal. 2011. "A genetic algorithm for regression test case prioritization using code coverage". In International journal of computer science and engineering. vol. 3. no. 5. 1839-1847.

[10] J. A. Nuh., T. W. Koh., S. Baharom., M. H. Osman., L. Babangida., S. Letchmunan., S. N. Kew. 2023. "Diversity based test case prioritization technique to improve faults detection rate". In International journal of advanced computer science and applications (IJACSA). Vol. 14. Issue. 6. doi: 10.14569/IJACSA.2023.0140698.

[11] M. Khatibsyarbini., M. A. Isa., D. N. A. Jawawi., M. L. M. Shafie., W. M. N. Wan-Kadir., H. N. A. Hamed., M. D. M. Suffian. 2021. "Trend application of machine learning in test case prioritization: a review on techniques". In IEEE Access. vol. 9. pp. 166262-166282. doi: 10.1109/ACCESS.2021.3135508.

[12] A. De Lucia., M. D. Penta., R. Oliveto., A. Panichella. 2012. "On the role of diversity measures for multi-objective test case selection". In 7th international workshop on automation of software test (AST), Zurich, Switzerland, 145-151. doi: 10.1109/IWAST.2012.6228983.

[13] Z. Li., Y. Xi., R. Zhao. 2020. "A hybrid algorithms construction of hyper-heuristic for test case prioritization". IEEE 4th annual computers , software and applications conference (COMPSAC), Madrid, Spain. 1749-1754, doi: 10.1109/COMPSAC48688.2020.000-2.

[14] R. Pan., M. Bagherzadeh., T. A. Ghaleb., L. Briand. 2022. "Test case selection and prioritization using machine learning: a systematic literature review". In Empirical software engineering, 27,29., doi: https://doi.org/10.1007/s10664-021-10066-6.

[15] A. Panichella., R. Oliveto., M. D. Penta., A. De Lucia. 2015. "Improving multi-objective test case selection by injecting diversity in genetic algorithms". In IEEE transactions on software engineering, vol. 41, no. 40, 358-383. doi: 10.1109/TSE.2014.2364175.

[16] J. J. Durillo., A. J. Nebro. 2011. "jMetal: A java framework for multi-objective optimization". In Advances in engineering software, vol. 42, 760-771. doi: https://doi.org/10.1016/j.advengsoft.2011.05.014.

[17] M. Ray., D. P. Mohapatra. 2014. "Multi-objective test prioritization via a genetic algorithm". In Innovations syst softw eng 10. 261-270. doi: https://doi.org/10.1007/s11334-014-02342-2.

[18] A. Bajaj., O. P. Sangwan. 2019. "A systematic literature review of test case prioritization using genetic algorithms". In IEEE Access. vol. 7. pp. 126355-126375. doi: 10.1109/ACCESS.2019.2938260.

[19] D. Di Nucci. 2018. "Methods and tools for focusing and prioritizing the testing effort". In IEEE International conference on software maintenance and evolution (ICSME), Madrid, Spain. 722-726. doi: 10.1109/ICSME.2018.00089.

[20] A. Bajaj., O. P. Sangwan. 2019. "Study the impact of parameter settings and operators role for genetic algorithm based test case prioritization". In Proceedings of International conference on sustainable computing in science, technology and management (SUSCOM 2019). doi: https://dx.doi.org/10.2139/ssrn.3356318.