



KPSO-Fuzzy: An AI-based Hybrid Approach for Software Requirements Prioritization

Rahila Anwar* , Muhammad Bilal Bashir 
*Corresponding author: raheelaanwar27864@iqraisb.edu.pk

Article info

Keywords:

artificial intelligence
clustering
elbow method
fuzzy logic
genetic algorithm: hybrid
technique
hybrid genetic algorithms
machine learning
multi-objective artificial bee
colony optimization
particle swarm optimization
requirement engineering
requirement prioritization
requirement analysis

Submitted: 4 Mar. 2026

Revised: 13 Mar. 2026

Accepted: 7 May 2026

Available online: 12 May 2026

Abstract

Context: Software requirements outline customer expectations for their software and are critical for successful project outcomes. As software becomes increasingly complex due to its size and diverse features, it is vital to prioritize these requirements to utilize development resources effectively. To address this challenge, researchers are exploring new strategies and improved solutions using artificial intelligence (AI) tools. In our systematic literature review conducted in 2023, we found that existing requirements prioritization techniques predominantly rely on human input and have several limitations. These include inaccuracies, overlapping results, scalability issues, and excessive time consumption. These challenges can be mitigated by incorporating key features from AI-based software requirements prioritization techniques.

Objective: The primary goal of this study is to develop a Hybrid AI-based requirements prioritization technique named as "KPSO-Fuzzy that effectively balances user preferences and technical dependencies.

Method: We propose a Hybrid prioritization method called Hybrid KPSO-Fuzzy. First, we use *K*-Means clustering to group technically dependent functional requirements into three distinct clusters. In the next phase, we apply the Particle Swarm Optimization (PSO) algorithm along with Fuzzy Logic to prioritize each cluster simultaneously. Clusters that achieve higher accuracy will be selected to create the most effective prioritized lists of requirements. We conducted extensive experiments to validate our proposed approach, focusing on accuracy, scalability, and computation time.

Results: The comparative analysis shows that our proposed Hybrid KPSO-Fuzzy method outperforms PSO, Fuzzy Logic, Multi-objective Artificial Bee Colony optimization and Hybrid Genetic Algorithms in terms of accuracy, scalability, and efficiency.

Conclusions: Overall, this study clarifies the application domains for various AI-based techniques, enabling users to maximize their benefits.

1. Introduction

Requirement Engineering (RE) is a fundamental phase of software engineering that identifies system requirements by considering client needs, stakeholder analysis, development context, negotiated outcomes, and evolving requirement management [1–4]. Requirement

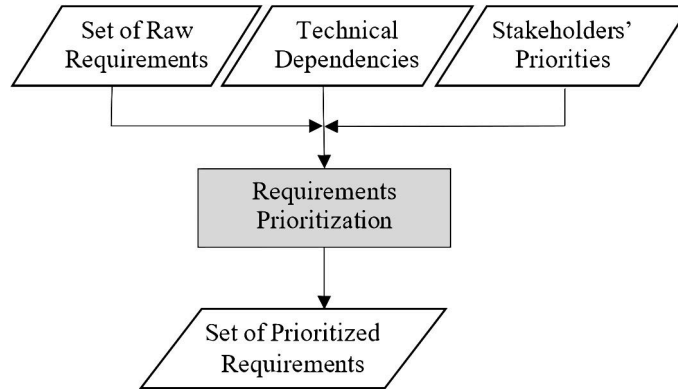


Figure 1. Software requirements prioritization process

Engineering also includes Requirement Prioritization, which supports analysis, negotiation, project planning, and timely software delivery [5–7]. Figure 1 presents the prioritization process involving inputs, processing, and outputs. Under limited resources, tight schedules, and high client expectations, prioritization enables effective evaluation of requirements while addressing time, cost, and resource constraints [4, 8, 9]. The execution order of requirements influences user experience and depends on technological and resource limitations, producing an ordered list that manages dependencies and feasible implementation sequences [4, 10–12]. Therefore, identifying critical requirements is essential [5, 13]. Many conventional requirement prioritization techniques have been proposed [1–3, 5, 8, 9, 14–16], including quantitative and subjective methods such as Analytical Hierarchy Process (AHP), Cumulative Voting, and Numerical Assignment. Ranking-based approaches are time- and cost efficient but limited in accuracy, while AHP requires expert involvement and high computational effort [5]. Existing techniques face challenges related to computational complexity, limited automation, scalability issues, ambiguity, uncertainty, and high computational cost, particularly in projects with conflicting risks and requirements [5, 6, 8, 15, 17]. Consequently, AI-based approaches – including fuzzy logic, genetic algorithms, simulated annealing, ant colony optimization, Artificial Bee Colony optimization, particle swarm optimization, and machine learning – have been introduced for requirement prioritization [14, 18–26].

Machine learning enables automated learning without explicit programming and improves scalability and prioritization accuracy compared to AHP [26, 27]. However, ML approaches still show disagreement in prioritization outcomes, including 4% disagreement [26], 10–20% disagreement [28], 74% recall with 16% precision [29], and 12–24% disagreement across criteria [27]. Fuzzy logic addresses imprecise and linguistic decision-making problems [30] and was introduced by Lotfi Zadeh [31]. Although scalable, fuzzy-based methods may lack optimal accuracy [2, 5, 30, 32–35]. Optimization algorithms also experience performance degradation as requirement size increases [9, 19, 21, 36–40] and may generate redundant or repeated solutions due to stochastic behavior [9, 11, 19, 36, 37, 39, 39–43].

In this study, two core parameters are considered: technical dependency [11, 43, 44] and user preferences [22, 37, 43]. Technical dependency captures interrelationships among requirements where implementation depends on prerequisite components [45, 46], while user preferences reflect stakeholder perceived importance and system value, aligning prioritization with user needs [19, 43]. These parameters are selected as generalizable inputs for computational prioritization frameworks [2, 5], as they can be quantified through expert elicitation, stakeholder surveys, or dependency matrices. Compared to domain-specific

factors such as resource constraints or volatility, empirical studies emphasize dependency management and user involvement as key drivers of effective prioritization [27, 45].

Although many studies evaluate prioritization approaches, no comprehensive divide and conquer framework integrating multiple AI-based techniques has been thoroughly investigated in terms of accuracy, scalability, and efficiency. Moreover, experimental guidance for selecting appropriate techniques remains limited. To address this gap, rigorous experimentation is conducted to obtain an effective hybrid prioritization solution.

The rest of this paper is structured as follows: in Section 2, we briefly discuss work done in similar areas by other researchers; in Section 3, we describe details of the research setup; in Section 4, we report the results of the study. Then, in Section 5, we discuss the collected results, as well as describe potential threats to the validity of the research, and we conclude the work in Section 6.

Research objective

The primary goal of this study is to develop a Hybrid AI-based requirements prioritization technique that balances user preferences and technical dependencies. The research integrates *K*-Means clustering, Fuzzy Logic, and Particle Swarm Optimization (PSO) to improve prioritization accuracy, efficiency, and scalability. The specific objectives are as follows:

1. **Conduct a literature review to analyze existing AI-based requirements prioritization techniques**, examining strengths, limitations, and evaluation parameters to identify research gaps guiding the hybrid technique development.
2. **Develop and implement a hybrid AI-based prioritization technique**, integrating technical dependency and user preference for functional requirement prioritization, evaluated using cross-validation based on accuracy, scalability, and efficiency.
3. **Perform a comparative analysis with state-of-the-art techniques** by benchmarking the proposed method against existing approaches using Dataset 1 and Dataset 2 with three parameter variants, assessing precision, scalability, efficiency, and statistical significance.

2. Related work

Software requirements prioritization is essential in requirements engineering to ensure early implementation of valuable and time-critical features. Traditional techniques such as MoSCoW, Analytic Hierarchy Process (AHP), and cumulative voting are widely used but face challenges related to uncertainty, scalability, and conflicting stakeholder preferences. To address these limitations, AI and soft-computing approaches integrating swarm intelligence, optimization, machine learning, and fuzzy logic have been explored to improve prioritization accuracy and adaptability. For example, Sadia and Faisal [32] proposed a fuzzy inference system for volatile requirement prioritization, Singh et al. [17] introduced a Hybrid LFPP-ANN model, Bisht and Kushwaha [47] incorporated stakeholder perspectives using fuzzy membership functions, and Abusaeed et al. [48] applied F-AHP for prioritizing cost factors in agile development. Jawale and Bhole [5] proposed hierarchical fuzzy cumulative voting but lacked scalability and explicit dependency modeling. Ahmad et al. [2] extended MoSCoW using fuzzy logic, though the approach relied heavily on subjective classification. Valsala and Nair [9] developed a Hybrid ILP-genetic model for release planning, achieving

effective optimization but with high computational cost and limited adaptability to dynamic requirements. In contrast, the proposed model introduces several enhancements:

- **Data-driven clustering using K -Means:** Unlike [5] and [2], clustering groups technically dependent requirements prior to prioritization, improving scalability and reducing cognitive complexity.
- **Optimization via PSO:** To address computational inefficiencies reported in [9], Particle Swarm Optimization (PSO) is applied to clustered data, enabling faster convergence and reduced time complexity.
- **Fuzzy inference for flexibility:** Fuzzy Logic integrates user preference and technical dependency, capturing subjective–objective trade-offs absent in [9] and [2].
- **Cross-validation and accuracy reporting:** The framework is evaluated on synthetic and real-world datasets, achieving over 95% accuracy with statistically validated performance metrics.

Additional studies have explored automation, machine learning, and dependency-aware prioritization. Shao et al. [44] proposed DRank integrating requirement dependencies using RankBoost and PageRank-Req, while Bollumpally et al. [23] introduced an XGBoost-based prioritization workflow that improves accuracy but requires retraining after system updates. Hujainah et al. [49] developed SRPTackle combining K -Means clustering and binary search trees for scalability, though with limited dependency handling, and Dang et al. [50] proposed MLPrior to reduce labeling costs through misclassification-based prioritization. Optimization-based approaches include a least-squares random genetic algorithm by Ahuja et al. [14], a parallel bee-foraging method by Alrezaamiri et al. [51], and a multi-objective PSO technique by Nazir et al. [25], achieving improved prioritization performance despite heuristic parameter tuning challenges. Hybrid optimization methods further combine complementary strengths of metaheuristics:

- **CDBR using PSO:** Gupta and Gupta [10] proposed a dependency-based collaborative prioritization approach achieving faster convergence than GA, though execution relations among dependencies were not considered.
- **Genetic algorithm with swarm intelligence:** Maghawry et al. [52] introduced a Hybrid GA–PSO framework incorporating K -Means-based selection and rehabilitation strategies to improve convergence and population diversity.
- **Hybrid swarm-based MOEA:** Marghny et al. [53] proposed HABCDE, integrating Artificial Bee Colony with Differential Evolution to solve the Next Release Problem, achieving improved Pareto-optimal solutions.

Although AI-based techniques improve requirement prioritization, many rely on complex black-box models that limit interpretability and real-time applicability. To address this, the proposed framework introduces a lightweight hybrid approach integrating K -Means clustering, PSO optimization, and fuzzy inference to jointly model technical dependencies and user preferences. However, real-world adoption of hybrid AI prioritization remains limited due to scarce benchmarking datasets, insufficient cross-validation, adaptive parameter tuning challenges [54], scalability issues, and inconsistent evaluation practices, highlighting the need for an adaptive, interpretable, and scalable framework as identified in recent studies [55] and summarized in Table 1.

Table 1. Strengths and limitations of AI-based RP techniques

Techniques	Strengths	Limitations
Fuzzy Logic-based RP	Scalable [2, 5, 32–35, 47]	Need improved accuracy [2, 5, 30, 32–35]
Optimization-based RP	Accurate results [11, 14, 43]	Less scalable [9, 19, 21, 36–40, 47]; Redundant solutions [9, 11, 19, 36, 37, 39–43]; Slow for large requirements [26]
Machine Learning-based RP	Scalable and optimal [10, 26–29, 44]; Less redundancy [23, 49]; Time-efficient [10, 26, 44]	Need improved accuracy [26–28]

3. Research method

A research design provides a structured framework guiding data collection and analysis for addressing specific research questions or hypotheses. The research paradigm determines how problems are formulated and methodologically addressed [56]. This study adopts an experimental quantitative research design within the Positivism paradigm, emphasizing objectivity and minimizing researcher bias to ensure knowledge independence from personal values and beliefs.

3.1. Research goal and questions/hypotheses

The primary objective is to develop a Hybrid AI-based framework for functional requirement prioritization through the integration of multiple AI techniques. The study addresses the following research questions (RQs):

1. **What are the strengths and limitations of existing AI-based software requirements prioritization techniques?**
2. **How can a Hybrid framework integrating *K*-Means clustering, Fuzzy Logic, and Particle Swarm Optimization (PSO) be designed and implemented for functional requirement prioritization?**
3. **Does the proposed Hybrid approach (KPSO-Fuzzy) improve accuracy, performance, and scalability compared to existing techniques?**

3.2. Research method/procedure

The proposed model operates as follows:

1. A set of n functional requirements is considered, each associated with technical dependency and user preference scores scaled on a five-point scale (1–5), where higher values indicate higher priority [50].
2. AI techniques including *K*-Means (Machine Learning), PSO (Optimization), and Fuzzy Logic were selected based on literature analysis and domain suitability [55].
3. **Phase 1: Requirement clustering using *K*-Means** *K*-Means clustering is applied during preprocessing to group requirements into three clusters (High, Medium, Low) based on technical dependency. This enables early identification of dependent requirements, reduces complexity through divide-and-conquer processing, and minimizes noise. The optimal number of clusters was determined as shown in Figure 2 using the Elbow Method [57]. The Within-Cluster Sum of Squares (WCSS) showed a clear elbow at

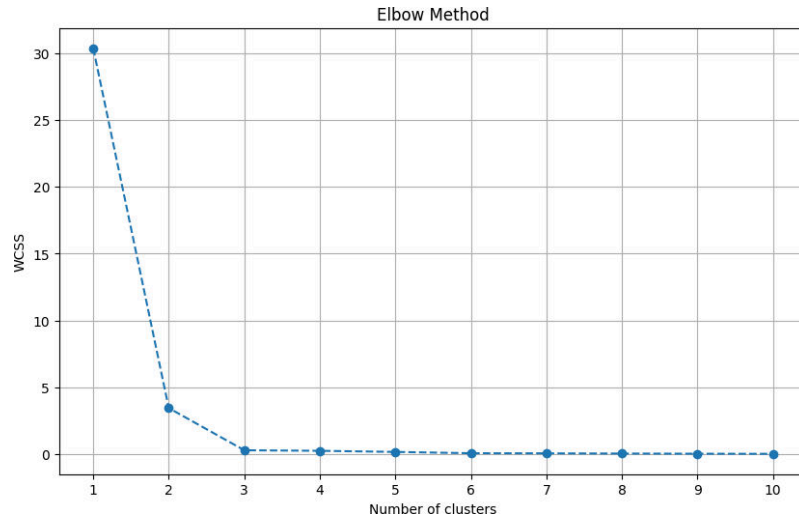


Figure 2. Elbow method for K -Means clustering

$K = 3$, providing the best balance between compactness and model complexity. This clustering improves PSO convergence and prioritization accuracy while enhancing fuzzy rule interpretability.

4. **Phase 2: Prioritization using Fuzzy Logic** PSO and Fuzzy Logic are applied independently and in parallel within each cluster. Fuzzy Logic prioritizes requirements using normalized Technical Dependency (TD) and User Preference (UP) inputs with trapezoidal membership functions [17, 47, 48]. Inputs are classified into Low, Medium, and High linguistic categories, producing a priority score through predefined fuzzy rules.
5. **Phase 3: Optimization with PSO** PSO identifies an optimal requirement sequence minimizing disagreement [27, 28] with a benchmark list derived from weighted TD and UP scores (e.g., 0.7 TD, 0.3 UP). Particle Swarm Optimization (PSO) is adapted to operate within a permutation-based search space to prevent requirement duplication in generated solutions. Each particle encodes a candidate priority sequence. After position updates, a repair mechanism identifies duplicate requirement IDs and replaces them with unassigned elements to preserve permutation feasibility. The adaptation builds upon the standard PSO framework [58, 59] while ensuring valid and non-redundant prioritization outcomes. The objective function minimizes positional mismatches:

```
def objective_function(particle_order, benchmark_order):
    return sum(p != b for p, b in zip(particle_order, benchmark_order))
```

6. **Accuracy and efficiency evaluation** Accuracy is computed using disagreement between prioritized lists and benchmark rankings. Cross-validation selects the highest cluster accuracy values, which are averaged to obtain final Hybrid accuracy. Efficiency is measured through computation time [34, 60], selecting the minimum time per cluster.
7. **Comparison across techniques** Outputs of PSO, Fuzzy Logic, and the Hybrid approach are compared using accuracy, scalability, and efficiency metrics. Ranking conflicts are resolved using normalized average priority scores and agreement thresholds, producing a merged global prioritized list.
8. **Statistical validation** Statistical significance is evaluated using paired t-tests [22] or Wilcoxon Signed-Rank tests [24, 27, 61], depending on data normality. Analysis includes

p -values, confidence intervals, and effect sizes to validate robustness against PSO, Fuzzy Logic, MOABC, and HGA methods.

The proposed AI-based Hybrid framework integrates K -Means clustering, PSO optimization, and Fuzzy Logic inference within a divide-and-conquer prioritization strategy to achieve accurate, scalable, and computationally efficient solutions. Figure 3 illustrates the sequential pipeline from clustering to optimization and fuzzy inference.

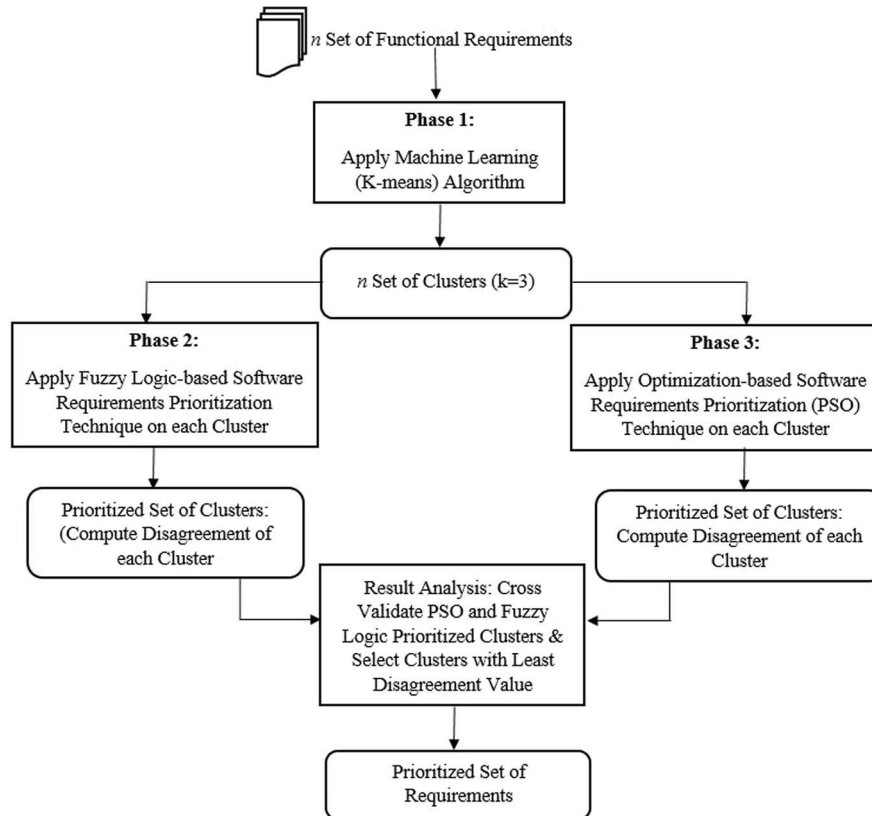


Figure 3. Proposed hybrid KPSO-fuzzy model for functional requirements prioritization

Dataset description

Two dataset categories were used to evaluate model performance and generalizability [20, 24]. **Dataset 1** contains 8083 functional requirements collected from 22 real-world case studies across domains such as Driver Drowsiness Detection, Smart Diet Monitoring, Parts Fitment, and Child Immunization systems. Each requirement includes technical dependency and user preference ratings on a five-point Likert scale [37]. **Dataset 2** is a synthetically generated dataset with TD and UP values uniformly distributed between 0 and 1, enabling controlled parameter tuning and sensitivity analysis of PSO and fuzzy

Table 2. Details of datasets used in the study

Dataset	Source	Type of requirements	Size of requirements
Dataset 1	22 Case Studies (Online)	Functional	Varying Size
Dataset 2	Random Numbers	Functional	Varying Size

inference mechanisms. Dataset sizes range from 20 to 2,000 requirements, representing realistic small-to-large software projects. A summary of dataset characteristics is presented in Table 2.

Evaluation measures

The proposed Hybrid KPSO-Fuzzy model was evaluated using three metrics: accuracy, efficiency, and scalability, selected based on established literature. Prior studies [10, 44] emphasize dependency minimization for improving prioritization accuracy; therefore, technical dependency and user preference were used as primary input criteria [22, 44, 51, 61]. Accuracy was measured through disagreement between predicted and benchmark rankings using normalized Kendall's Tau and Spearman's Rank Correlation, consistent with existing approaches [11, 23, 26–29, 43, 49, 60, 62]. Stakeholder–developer disagreement was also considered a precision indicator [10]. Efficiency was evaluated using average computation time required by PSO and fuzzy logic to achieve stable solutions [10, 13, 26, 29, 51, 63]. Scalability was analyzed across datasets of varying sizes to assess performance stability under increasing requirements [14, 34, 49]. All metrics were computed at the cluster level by independently evaluating PSO and fuzzy logic, while final hybrid performance combined maximum accuracy with minimum computation time across clusters.

4. Results

The proposed technique, Hybrid KPSO-Fuzzy (*K*-Means, PSO, and Fuzzy Logic), integrates clustering, optimization, and fuzzy inference for functional requirements prioritization. The process begins by initializing a population of solutions, which is provided as input to the *K*-Means algorithm [64] to group technically dependent requirements into three clusters. The resulting clusters are then used as input for both Particle Swarm Optimization (PSO) [25, 65] and fuzzy logic using a trapezoidal membership function [17] for further prioritization. Finally, the best solution obtained from PSO and fuzzy logic is selected as the final prioritization result and compared with standalone PSO and fuzzy logic approaches based on accuracy, efficiency, and scalability.

Algorithm implementation

In this section, we present the Hybrid KPSO-Fuzzy, *K*-Means, Fuzzy logic, and Particle Swarm Optimization (PSO) algorithms. These algorithms are divided into the following subsections:

Architecture of the proposed hybrid (KPSO-Fuzzy) technique

The Hybrid KPSO-Fuzzy tool simulates software requirements prioritization based on user preferences and functional dependencies to achieve an optimal solution. It provides an intuitive interface with results in tabular format, exportable as CSV files. The tool consists of three main modules: *K*-Means, PSO, and fuzzy logic, which share information to perform the simulation. Algorithm 1 presents the pseudo-code of the **Hybrid KPSO-Fuzzy**.

 Algorithm 1. Pseudo-code for the hybrid KPSO-fuzzy technique

```

BEGIN
Step 1: Import Required Libraries
  Import Pandas, NumPy, Sklearn, Skfuzzy, Pyswarm, Matplotlib, etc.
Step 2: Data Processing
  Upload Excel file and load data into dataframe df
  Rename columns to: ['Requirement ID', 'Technical Dependency', 'User Preference']
  Normalize values using MinMaxScaler
Step 3: Prioritization
  for each Requirement in df do
    Assign priority label (High, Medium, Low) based on Technical Dependency and User Preference
  end for
Step 4: Apply KMeans Clustering
  Extract feature: X = df[['Technical Dependency']]
  Initialize list: wcss = []
  Set max-clusters = 10
  for k = 1 to max-clusters do
    Initialize KMeans with k clusters
    Fit model to X
    Append inertia to wcss
  end for
  Plot WCSS vs k and determine elbow point
  Select optimal cluster number K (typically K = 3)
  Apply final KMeans clustering
Step 5: Apply Fuzzy Logic Prioritization
  for each cluster  $C_k$  in K do
    Define trapezoidal membership functions
    Compute fuzzy priority score
    Sort requirements by fuzzy priority
  end for
Step 6: Apply PSO Prioritization
  for each cluster  $C_k$  in K do
    Initialize particles
    Update velocities and positions
    Optimize ranking using disagreement metric
  end for
Step 7: Plot Results
  Plot Accuracy and Efficiency comparisons
END

```

Workflow of the K -Means model

The K -Means clustering algorithm is an iterative method that groups similar data points into K clusters based on their characteristics [64]. Algorithm 2 provides detailed information on how the K -Means model operates.

Workflow of Particle Swarm Optimization algorithm (PSO)

Particle Swarm Optimization (PSO) is a population-based optimization technique inspired by collective foraging behavior, where particles represent candidate solutions that iteratively update their positions and velocities by learning from personal and global best experiences.

Algorithm 2. *K*-Means-based prioritization of functional requirements

Begin

2: **Step 1: Data Collection**
Gather functional requirements, technical dependencies, and user preferences.

4: **Step 2: Data Pre-processing**
Normalize the dataset using `MinMaxScaler` from `sklearn.preprocessing`.

6: Scale values to the range $[0, 1]$ for uniform processing.

Step 3: Import Libraries

8: Import necessary libraries including NumPy and KMeans from `sklearn.cluster`.

Step 4: Load Dataset

10: Load dataset (X, Y) containing scaled values (1 to 5) for technical dependency and user preference.

Step 5: Determine Optimal Clusters (Elbow Method)

12: Initialize empty list `WCSS = []` to store within-cluster sum of squares.
Set maximum number of clusters: `max-clusters = 10`.

14: **for** $k = 1$ to $max - clusters$ **do**
 Apply KMeans with k clusters: `KMeans(n-clusters=k, random-state=0, n-init='auto').fit(X)`

16: Append the WCSS (inertia) to the list.
end for

18: Plot *WCSS* vs. number of clusters.
Identify elbow point where WCSS decrease slows down.

20: Set $K = 3$ as optimal based on elbow point.

Step 6: Initialize K-Means

22: Randomly initialize K centroids.
Set the cluster size to 3 ($K = 3$ for High, Medium, Low priority).

24: Define KMeans parameters:
`kmeans = KMeans(n-clusters=3, random-state=0, n-init='auto').fit(X)`

26: **Step 7: Assign Clusters**
Assign each data point to the nearest centroid.

28: Recompute centroids by averaging points in each cluster.
Repeat until convergence to minimize squared distances.

30: **Step 8: Priority Classification**

High Priority: Features with significant impact based on user preferences and technical dependencies.

32: **Medium Priority:** Features with moderate impact.

Low Priority: Features with the least impact.

34: **End**

The velocity and position updates are defined as:

$$\begin{aligned}
 v_{ij}(t+1) &= wv_{ij}(t) + c_1r_1(x_{ij}^{p-best}(t) - x_{ij}(t)) \\
 &\quad + c_2r_2(x_j^{g-best}(t) - x_{ij}(t)), \\
 x_{ij}(t+1) &= x_{ij}(t) + v_{ij}(t+1).
 \end{aligned} \tag{1}$$

where $\mathbf{x}_i = (x_{i1}, \dots, x_{iD})$ and $\mathbf{v}_i = (v_{i1}, \dots, v_{iD})$ denote particle position and velocity in a D -dimensional space; x_{ij}^{p-best} and x_j^{g-best} are personal and global best positions. Parameters: $w = 0.4$, $c_1 = c_2 = 2.0$, $r_1, r_2 \in [0, 1]$, $t = 1, \dots, T$, with velocity bounded by v_{max} . PSO offers fast convergence with limited parameter tuning and effectively explores the solution space by guiding particles toward optimal solutions [25]. In this study, the `pyswarm` library was used to implement the PSO component for efficient functional requirement prioritization, with Algorithm 3 detailing the workflow.

Algorithm 3. PSO-based optimization of prioritized functional requirements

Begin
Step 1: Input Data from K-Means Model
3: Load the clustered requirements categorized as high, medium, and low priority.
Step 2: Initialize PSO Optimization Function
Define the `pso-optimize` function:
6: Initialize the swarm with random positions and velocities.
Update each particle's best-known position.
Update the global best-known position.
9: Adjust velocities and positions using cognitive and social components.
Step 3: Define Fitness Function
Define a fitness function to evaluate solutions.
12: The fitness function sums priorities, which can be customized for functional requirements.
Step 4: Initialize PSO Parameters
Define the number of particles and iterations.
15: Tune PSO parameters:
 $w = 0.7$ ▷ Tuned inertia weight
 $c_1 = 2.0$ ▷ Tuned cognitive coefficient
18: $c_2 = 2.0$ ▷ Tuned social coefficient
Step 5: Update Particle Positions
Update personal best positions and global best position.
21: Adjust velocities using inertia, cognitive, and social components.
Step 6: Evaluate Fitness
Evaluate the fitness of each particle to determine the best solution.
24: **Step 7: Return Best Solution**
Output the best solution found for functional requirement prioritization.
End

Workflow of fuzzy logic

Fuzzy logic is employed to prioritize functional requirements by balancing user preferences and technical dependencies while handling uncertainty inherent in decision-making. The implementation uses the `scikit-fuzzy` library (including `skfuzzy` and `skfuzzy.control`), enabling efficient construction and simulation of fuzzy inference systems for rapid prototyping and experimentation. A trapezoidal membership function is adopted due to its simplicity and flexibility [17, 47, 48], defined as:

$$\text{trapmf}(x; a, b, c, d) = \begin{cases} 0, & x < a \\ \frac{x - a}{b - a}, & a \leq x < b \\ 1, & b \leq x \leq c \\ \frac{d - x}{d - c}, & c < x \leq d \\ 0, & x > d \end{cases} \quad (2)$$

where a and d denote zero-membership boundaries, b and c define the plateau with full membership, and intermediate regions vary linearly. The proposed Hybrid KPSO-Fuzzy model applies a Mamdani fuzzy inference system [13, 66] to compute requirement priority scores using normalized Technical Dependency (TD) and User Preference (UP). Rule evaluation employs minimum and maximum operators for logical aggregation. The aggregated fuzzy outputs are defuzzified using the centroid (center-of-gravity) method [31, 67],

Algorithm 4. Fuzzy logic-based prioritization

Begin
Input: Data generated by K -Means model
Fuzzification: Convert numerical scores into fuzzy sets using membership functions
4: Define linguistic variables (e.g., Low, Medium, High) for both user preference and technical dependency
Define Fuzzy Rules:
Define fuzzy rules to model the relationship between user preference and technical dependency
Example: "IF User Preference IS High AND Technical Dependency IS Low THEN Priority IS High"
8: **Inference Engine:**
Use a fuzzy inference engine to evaluate fuzzy rules
Combine fuzzy sets according to defined rules using fuzzy operators
Defuzzification: Convert fuzzy output back into crisp scores using a defuzzification method
12: **Prioritization:** Rank functional requirements based on defuzzified scores
End

producing a crisp priority score:

$$y = \frac{\sum_{i=1}^n \mu_i \cdot x_i}{\sum_{i=1}^n \mu_i}$$

where x_i represents the output priority value and μ_i denotes the corresponding membership degree. The workflow of the fuzzy logic algorithm is clearly illustrated in Algorithm 4.

Comparative analysis of proposed Hybrid KPSO-Fuzzy vs PSO, Fuzzy Logic, MOABC, and Hybrid-GA

To evaluate the proposed Hybrid KPSO-Fuzzy technique, experiments were conducted on two datasets: Dataset 1, comprising 22 real-world case studies ensuring domain relevance, and Dataset 2, synthetically generated to simulate functional requirements of varying sizes and complexities. The model was compared against PSO, Fuzzy Logic, MOABC, and Hybrid Genetic Algorithm (HGA) across up to 10 runs per method to ensure statistical reliability. Three parameter configurations (V1–V3), varying PSO parameters (particles, iterations, inertia weight w , cognitive c_1 , and social c_2 coefficients) and fuzzy membership ranges, were applied to examine robustness.

Comparative accuracy analysis across techniques

Figures 4–5 show that the **Hybrid KPSO-Fuzzy** approach consistently achieves the highest accuracy across both datasets and all requirement scales. For small datasets (20–100 requirements), it reaches $\geq 95\%$, outperforming PSO and Fuzzy Logic, while MOABC achieves 80%–95% and Hybrid-GA 52%–65%. On medium-scale datasets (620–1000 requirements), Hybrid KPSO-Fuzzy remains above 97%, with PSO at 88%–94%, MOABC near 82%, Fuzzy Logic 76%–83%, and Hybrid-GA below 60%. For large datasets (2000 requirements), it peaks at **96.67%** (Dataset 1) and **99.57%** (Dataset 2), whereas PSO reaches 87%–94%, MOABC $\sim 82\%$, Fuzzy Logic $\sim 68\%$, and Hybrid-GA $\sim 52\%$. Overall, the proposed method improves accuracy up to **47%** over Hybrid-GA, **17%** over MOABC, and at least **10%** over PSO, confirming its effectiveness for large-scale prioritization.

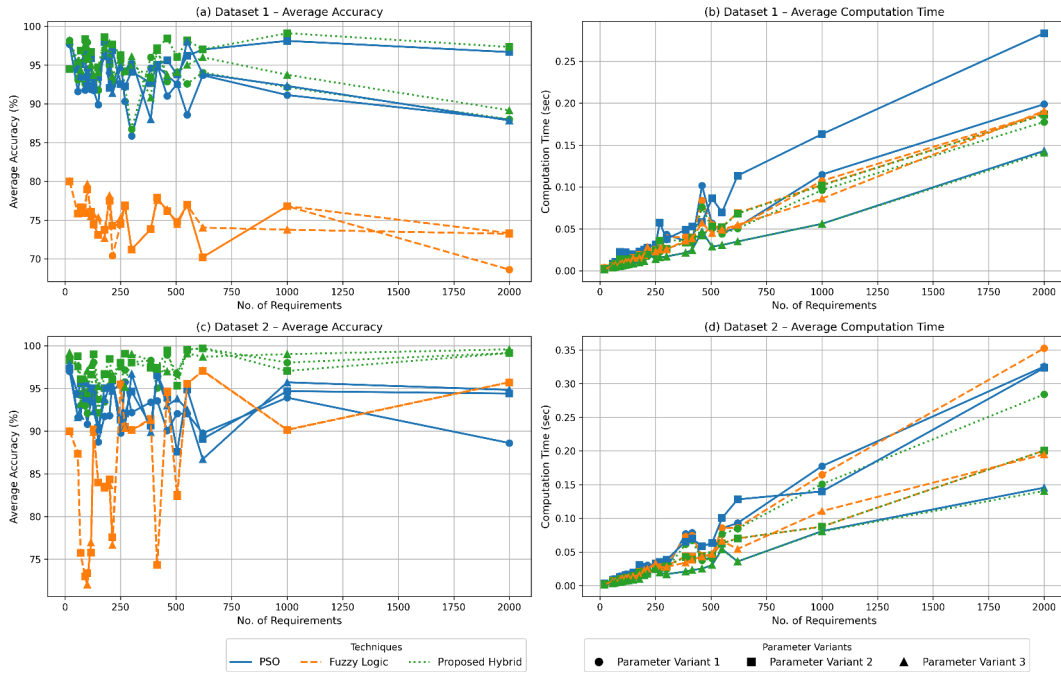


Figure 4. Performance trends of Hybrid KPSO-Fuzzy, PSO, and fuzzy logic across parameter variants 1-3

Comparison of Proposed Hybrid KPSO_Fuzzy, Hybrid_GA and MOABC Techniques

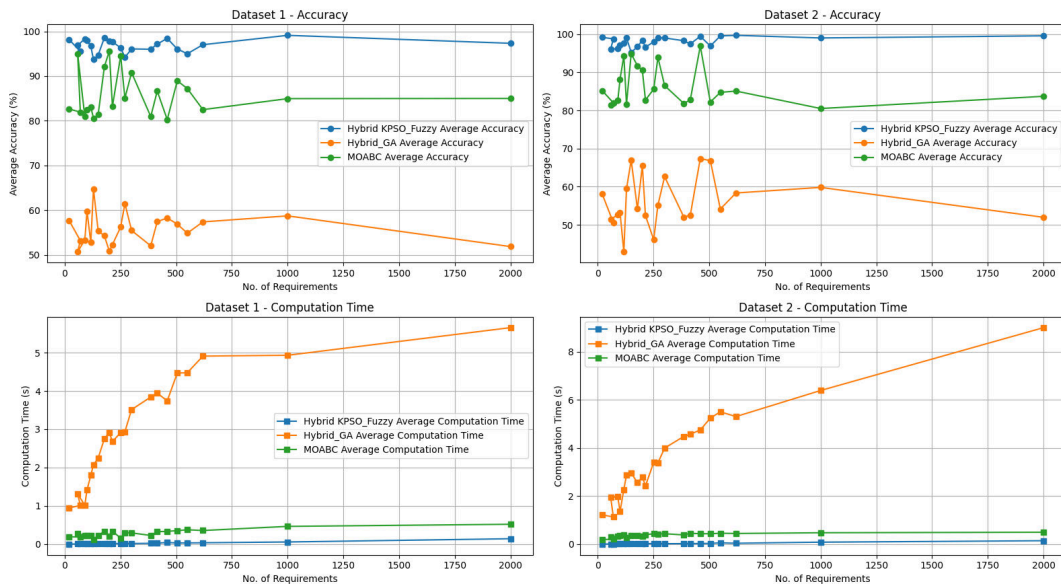


Figure 5. Comparison of hybrid KPSO-fuzzy, Hybrid-GA, and MOABC on dataset 1 and dataset 2

Comparative computation time analysis across techniques

For small datasets (20–100 requirements), computation times are minimal (0.03–0.08 s), with **Hybrid KPSO-Fuzzy** slightly faster than others – about 0.15–0.25 seconds faster than **Hybrid-GA** and 0.28 seconds faster than **MOABC**, also surpassing PSO and Fuzzy Logic. For medium datasets (620–1000 requirements), it remains most efficient (0.05–0.14 seconds for 1000 requirements) despite modest increases, while Hybrid-GA slows to 2.5–5.0 seconds and MOABC takes 0.18–0.35 seconds longer; PSO and Fuzzy Logic are faster but less accurate. For large datasets (2000 requirements), Hybrid KPSO-Fuzzy completes in 0.14–0.32 seconds, outperforming Hybrid-GA (6.5 seconds), MOABC (~ 0.52 seconds), and PSO/Fuzzy Logic (0.18–0.35 seconds). Overall, the proposed method demonstrates superior efficiency across all scales.

Scalability analysis for accuracy

Dataset 1: Fuzzy Logic shows poor scalability, declining from moderate accuracy for small datasets to 68%–76% for datasets over 1000 requirements. PSO maintains 87%–94% up to 1000 requirements but plateaus for larger sizes. MOABC is unstable on small datasets (80%–95%) and stabilizes at ~82% above 1000 requirements. Hybrid-GA starts at 51%–66%, reaching ~59% at 1000 and dropping to 52% at 2000. **Hybrid KPSO-Fuzzy** consistently outperforms all, maintaining 94%–98% for small datasets, 99% at 1000, and 96.67% at 2000, demonstrating robust scalability. **Dataset 2:** MOABC is unstable at small scales (80%–95%) and drops to ~82% beyond 620 requirements. Hybrid-GA starts at 50%–67% for small datasets and declines to ~52% at 2000 requirements, showing limited adaptability. **Hybrid KPSO-Fuzzy** consistently outperforms all, achieving 97%–98% at 1000 requirements and peaking at 99.57% for 2000, demonstrating robust scalability and high accuracy.

Scalability analysis for computational efficiency

Fuzzy Logic is the least efficient for large datasets due to variable computation times and declining accuracy, while PSO is faster but compromises precision. MOABC demonstrates limited scalability, taking 0.25–0.35 seconds for 20–100 requirements, rising to ~0.55 seconds at 1000 and stabilizing at 0.6 seconds for 2000. Hybrid-GA shows steep time growth: Dataset 1 increases from 1.0–1.3 seconds for 100 requirements to over 4.8 seconds at 1000 and ~5.8 seconds at 2000, while Dataset 2 rises from 1.5 seconds to above 6.5 seconds. In contrast, **Hybrid KPSO-Fuzzy** consistently delivers high efficiency, with Dataset 1 starting at 0.01 seconds, reaching 0.12–0.15 seconds at 1000 and remaining under 0.2 seconds at 2000, and Dataset 2 staying below 0.22 seconds. On average, it is 25–30 times faster than Hybrid-GA for datasets over 1000 requirements, highlighting superior computational efficiency and scalability.

Tables 3 and 4 show that proposed Hybrid KPSO-Fuzzy balances accuracy and computation time, making it the most scalable option for real-world implementation.

Statistical analysis

Statistical tests were conducted to evaluate the significance of the proposed Hybrid KPSO-Fuzzy technique by comparing its performance with Particle Swarm Optimization

Table 3. Performance breakdown of five prioritization techniques (hybrid KPSO-Fuzzy, Hybrid-GA, MOABC, PSO, and fuzzy logic)

Dataset size	Best technique	Worst technique	Observations
20–100 Req.	Hybrid KPSO-Fuzzy	Hybrid-GA/Fuzzy	Hybrid KPSO-Fuzzy shows a slight lead; Hybrid-GA and Fuzzy fluctuate more. MOABC performs moderately (80–95%). All techniques are relatively close.
620 Req.	Hybrid KPSO-Fuzzy	Hybrid-GA/Fuzzy	Hybrid’s lead becomes clearer; MOABC remains unstable (82%), Hybrid-GA still underperforms especially in Dataset 2.
1000 Req.	Hybrid KPSO-Fuzzy	Hybrid-GA/Fuzzy	MOABC struggles to maintain accuracy (82%), Hybrid-GA shows lower stability; Hybrid KPSO-Fuzzy dominates.
2000 Req.	Hybrid KPSO-Fuzzy (99.57%)	Fuzzy (68.61%), Hybrid-GA (~ 55%)	MOABC remains around 82%, Hybrid-GA fails to scale effectively. Hybrid KPSO-Fuzzy is the only technique maintaining both accuracy and efficiency at scale.

Table 4. Comparison of Hybrid KPSO-Fuzzy, PSO, and fuzzy logic techniques across three parameter variants and two benchmark datasets

Parameter variant	Dataset	Highest accuracy	Lowest accuracy	Key observations
(10, 15, 1.2, 1.2)	Dataset 1	Hybrid KPSO-Fuzzy Logic (92.15%)	Fuzzy Logic (76.8%)	Hybrid KPSO-Fuzzy outperforms PSO by 16%; Fuzzy Logic lags significantly. Hybrid KPSO-Fuzzy leads by 4% over PSO and 8% over Fuzzy.
	Dataset 2	Hybrid KPSO-Fuzzy (98.02%)	PSO (93.91%)	
(15, 15, 1.5, 2)	Dataset 1	Hybrid KPSO-Fuzzy (99.11%)	Fuzzy Logic (76.8%)	Hybrid KPSO-Fuzzy achieves near-perfect accuracy, Fuzzy Logic under performs. Hybrid KPSO-Fuzzy is 3% ahead of PSO and 7% ahead of Fuzzy.
	Dataset 2	Hybrid KPSO-Fuzzy (97.04%)	Fuzzy Logic (90.13%)	
(10, 10, 1.2, 1.9)	Dataset 1	Hybrid KPSO-Fuzzy (93.75%)	Fuzzy Logic (73.76%)	Hybrid KPSO-Fuzzy wins, with PSO slightly behind (92.31%). Hybrid KPSO-Fuzzy maintains the lead, Fuzzy Logic slightly better than PSO.
	Dataset 2	Hybrid KPSO-Fuzzy (99.01%)	PSO (95.71%)	

(PSO), Fuzzy Logic, Multi-Objective Artificial Bee Colony (MOABC) [51], and Hybrid Genetic Algorithm (HGA) [52] approaches. Normality of the experimental results was assessed using the Shapiro-Wilk test [44, 63]. Datasets satisfying the normality assumption ($p > 0.05$) were analyzed using paired t -tests, whereas non-normal distributions ($p \leq 0.05$) were evaluated using the Wilcoxon signed-rank test [24, 61]. All statistical evaluations were performed for both average accuracy and average computation time across Dataset 1 and Dataset 2.

Figures 6–7 present histogram distributions, boxplots, and Q-Q plots that illustrate the statistical behavior of the compared techniques. The sub-figures corresponding to Dataset 1 and Dataset 2 demonstrate consistent distribution patterns and validate the suitability of parametric and non-parametric hypothesis testing procedures. The hybrid KPSO-Fuzzy achieved statistically significant improvements in prioritization accuracy in both datasets compared to all baseline techniques ($p < 0.01$). The method also demonstrated consistently lower computation time and large effect sizes, confirming superior efficiency, robustness, and overall performance.

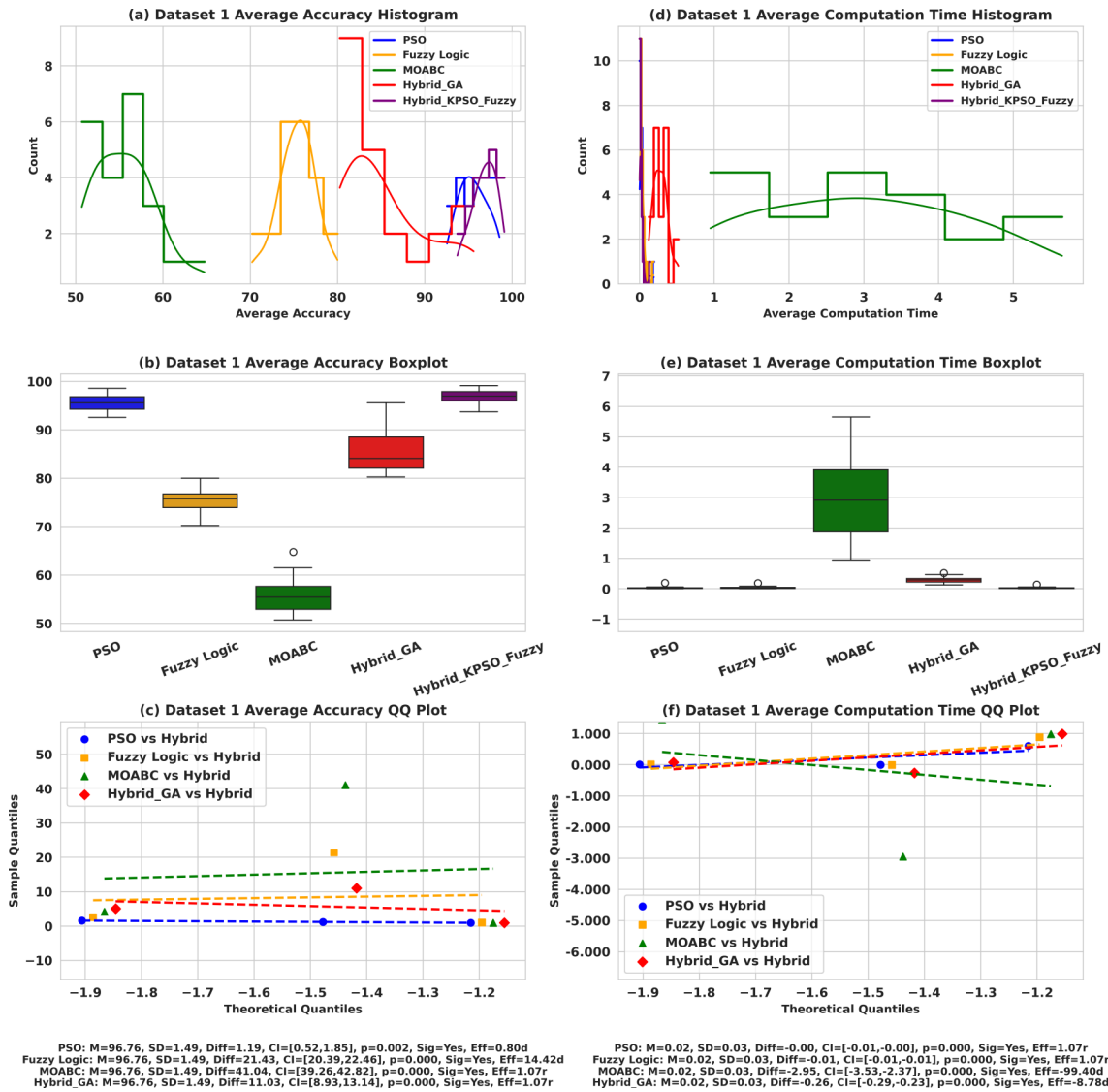


Figure 6. Statistical Analysis of Hybrid KPSO-Fuzzy versus baseline Techniques for Average Accuracy and Computation Time on Dataset 1

5. Discussion

This study investigates the Hybrid KPSO-Fuzzy technique, which combines *K*-Means clustering, PSO, and Fuzzy Logic for functional requirement prioritization. The approach is evaluated using accuracy, computation time, and scalability, and compared with PSO, Fuzzy Logic, MOABC, and Hybrid Genetic Algorithms (HGA).

The proposed Hybrid KPSO-Fuzzy method demonstrates superior overall performance, achieving up to 99.57% accuracy while maintaining stable scalability and low computation time as dataset size increases. The integration of clustering, swarm optimization, and fuzzy reasoning enhances prioritization stability and robustness. Performance gains are less pronounced for small datasets (20–100 requirements), and the hybrid design introduces higher implementation complexity compared with standalone techniques.

PSO provides efficient computation and relatively stable accuracy but shows performance degradation beyond 1000 requirements due to local optima tendencies. Fuzzy Logic performs

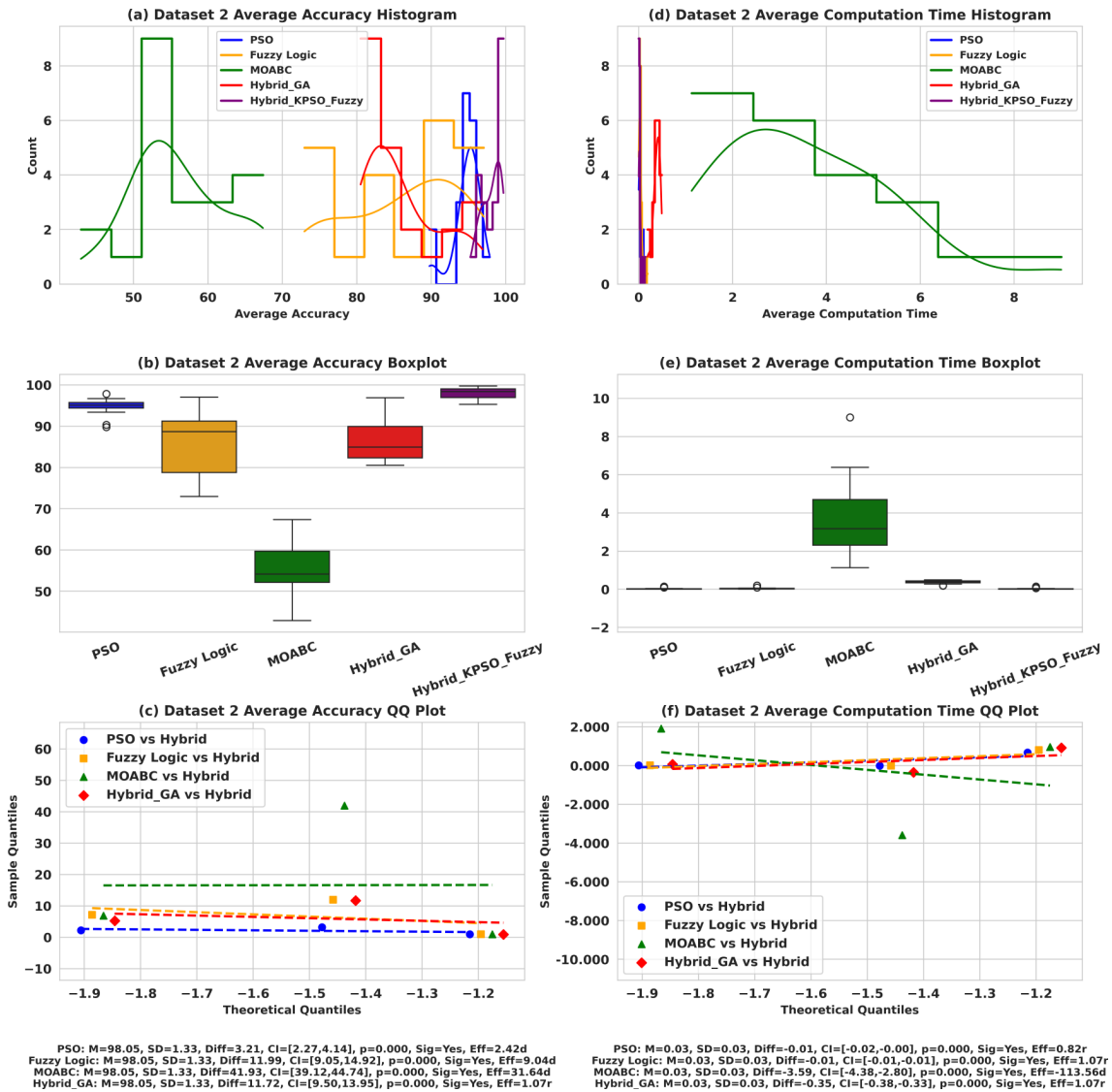


Figure 7. Statistical Analysis of Hybrid KPSO-Fuzzy versus baseline Techniques for Average Accuracy and Computation Time on Dataset 2

well for small datasets because of its simplicity and speed; however, accuracy decreases significantly for large-scale problems (down to 68.61%), indicating limited scalability. MOABC achieves acceptable results for small and medium datasets but exhibits accuracy stagnation near 82% and increased computation cost at larger scales. HGA shows predictable evolutionary behavior but records the lowest accuracy (below 65%) and the highest computation time, making it unsuitable for large requirement sets.

Overall, Hybrid KPSO-Fuzzy offers the best balance between accuracy, efficiency, and scalability. Compared with existing approaches, which often suffer from instability or computational overhead caused by randomization [19–21, 36, 38, 40, 41, 51], the proposed framework provides a robust and scalable solution for large-scale requirement prioritization.

Threats to validity

Several factors may influence the validity of the study outcomes.

Construct validity is limited by the scope of the research questions; this was mitigated through systematic categorization and analysis of fuzzy logic-, optimization-, and machine learning-based prioritization methods.

Data collection validity may be affected by incomplete retrieval of studies; therefore, multiple repositories and carefully selected keywords were used, although results may vary with larger datasets or alternative techniques.

Internal validity may be affected by constructing the benchmark priority list from the same input factors – technical dependency (TD) and user preference (UP) – used by the proposed model, introducing potential circularity in evaluation. However, the benchmark is used as a structured reference baseline rather than an absolute ground truth. This limitation is mitigated through evaluation across multiple datasets and by the stochastic, hybrid nature of the approach. Comparative analysis with independent techniques further supports the robustness of the results.

Dataset scaling within the range 1–5 may influence algorithm behavior, where increased variability could reduce fuzzy accuracy while improving clustering performance.

Metric variability arises from differences in requirement characteristics, algorithms, and project contexts; consistent evaluation criteria were applied to minimize this effect.

Evaluation scope focuses on objective metrics (accuracy, computation time, and scalability), while stakeholder-oriented measures remain future work.

The study also considers only two input criteria (user preference and technical dependency), adopts a fixed cluster size of three, and evaluates primarily functional requirements, which may limit generalizability. Despite these constraints, controlled experimental design and consistent evaluation ensure reliable comparative insights for AI-based requirement prioritization.

6. Conclusions

This study proposes the **Hybrid KPSO-Fuzzy** technique, a robust requirement prioritization framework that effectively balances *accuracy*, *computational efficiency*, and *scalability* by integrating technical dependency and user preferences. The approach follows three phases: *K*-Means clustering groups requirements, PSO and Fuzzy Logic refine prioritization, and cross-validation evaluates performance across key metrics. Experiments on two datasets demonstrate that Hybrid KPSO-Fuzzy consistently outperforms PSO, Fuzzy Logic, MOABC, and Hybrid-GA. Specifically, compared to PSO and Fuzzy Logic, Hybrid KPSO-Fuzzy improves accuracy by 3–5% and 15–27%, respectively, while reducing computation time by up to 10% and 5%. Against MOABC, it achieves up to 99.57% precision with average accuracy gains of 11–12% and faster computation across small, medium, and large datasets. Compared to Hybrid-GA, the technique delivers dramatic accuracy improvements of 41–42% and reduces computation time by 2.95–3.59 seconds on average. Overall, Hybrid KPSO-Fuzzy provides a stable, scalable, and efficient solution for large-scale software requirement prioritization, overcoming the limitations of existing AI-based approaches.

Future works

- Evaluate the Hybrid KPSO-Fuzzy technique using additional factors such as cost, redundancy, stakeholder preferences, and diverse requirement types, with dynamic parameter adaptation for real-time environments.
- Explore machine learning-based prioritization and adaptive weight optimization, including integration with PSO, MOABC, or deep learning for large-scale datasets.
- Address broader dependency types, including contribution and business dependencies, and improve scalability, efficiency, and redundancy reduction in optimization-based methods.
- Enhance fuzzy logic systems through automated rule generation and alternative membership functions (e.g., triangular, Gaussian) to improve accuracy and adaptability.
- Validate the approach using expert-ranked priorities and industrial case studies, with adaptive benchmark weight selection for project-specific optimization.

Overall, future research should focus on improving accuracy, scalability, efficiency, redundancy reduction, consistency ratio, self-adaptability, and the identification of optimal prioritization solutions for real-world applications.

CRedit authorship contribution statement

Rahila Anwar: Data curation, Methodology, Investigation, Writing – original draft, Writing – review and editing, Visualization. Muhammad Bilal Bashir: Conceptualization, Funding acquisition, Methodology, Investigation, Writing – review and editing, supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Funding

This research received no external funding.

References

- [1] M. Gen and L. Lin, “Genetic algorithms,” in *Wiley Encyclopedia of Computer Science and Engineering*. American Cancer Society, 2008, pp. 1–15.
- [2] K.S. Ahmad, N. Ahmad, H. Tahir, and S. Khan, “Fuzzy_MoSCoW: A fuzzy based MoSCoW method for the prioritization of software requirements,” in *International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT)*. Kerala State, Kannur, India: IEEE, 2017, pp. 433–437.
- [3] P. Achimugu, A. Selamat, R. Ibrahim, and M.N. Mahrin, “A systematic literature review of software requirements prioritization research,” *Information and Software Technology*, Vol. 56, No. 6, 2014, pp. 568–585.
- [4] F. Hujainah, R.B.A. Bakar, M.A. Abdulgaber, and K.Z. Zamli, “Software requirements prioritisation: A systematic literature review on significance, stakeholders, techniques and challenges,” *IEEE Access*, Vol. 6, 2018, pp. 71 497–71 523.

- [5] B. Jawale and A. Bhole, "Adaptive fuzzy hierarchical cumulative voting: A novel approach toward requirement prioritization," *International Journal of Research in Engineering and Technology*, Vol. 04, No. 05, 2015, pp. 365–370.
- [6] A. Hudaib, M.H. Qasem, and N. Obeid, "FIPA-based semi-centralized protocol for negotiation," in *Cybernetics Approaches in Intelligent Systems*, Advances in Intelligent Systems and Computing, R. Silhavy, P. Silhavy, and Z. Prokopova, Eds. Cham: Springer International Publishing, 2018, pp. 135–149.
- [7] J. Dąbrowski, E. Letier, A. Perini, and A. Susi, "Analysing app reviews for software engineering: A systematic literature review," *Empirical Software Engineering*, Vol. 27, No. 2, 2022, p. 43. [Online]. <https://link.springer.com/10.1007/s10664-021-10065-7>
- [8] L. Karlsson, T. Thelin, B. Regnell, P. Berander, and C. Wohlin, "Pair-wise comparisons versus planning game partitioning—experiments on requirements prioritisation techniques," *Empirical Software Engineering*, Vol. 12, No. 1, 2007, pp. 3–33.
- [9] S. Valsala and D.A.R. Nair, "Requirement prioritization and scheduling in software release planning using hybrid enriched genetic revamped integer linear programming model," *Research Journal of Applied Sciences, Engineering and Technology*, Vol. 12, No. 3, 2016, pp. 347–354.
- [10] A. Gupta and C. Gupta, "CDBR a semi-automated collaborative execute-before-after dependency-based requirement prioritization approach," *Journal of King Saud University – Computer and Information Sciences*, 2018, p. S1319157818304518.
- [11] P. Tonella, A. Susi, and F. Palma, "Interactive requirements prioritization using a genetic algorithm," *Information and Software Technology*, Vol. 55, No. 1, 2013, pp. 173–187.
- [12] M.I. Babar, M. Ghazali, D.N. Jawawi, S.M. Shamsuddin, and N. Ibrahim, "PHandler: An expert system for a scalable software requirements prioritization process," *Knowledge-Based Systems*, Vol. 84, 2015, pp. 179–202. [Online]. <https://linkinghub.elsevier.com/retrieve/pii/S0950705115001483>
- [13] K. Gulzar, J. Sang, M. Ramzan, and M. Kashif, "Fuzzy approach to prioritize usability requirements conflicts: An experimental evaluation," *IEEE Access*, Vol. 5, 2017, pp. 13 570–13 577.
- [14] H. Ahuja, Sujata, and U. Batra, "Performance enhancement in requirement prioritization by using least-squares-based random genetic algorithm," in *Innovations in Computational Intelligence : Best Selected Papers of the Third International Conference on REDSET 2016*, Studies in Computational Intelligence, B. Panda, S. Sharma, and U. Batra, Eds. Singapore: Springer, 2018, pp. 251–263.
- [15] R. Qaddoura, A. Abu-Srhan, M.H. Qasem, and A. Hudaib, "Requirements prioritization techniques review and analysis," in *International Conference on New Trends in Computing Sciences (ICTCS)*. Amman: IEEE, 2017, pp. 258–263.
- [16] M. Pergher and B. Rossi, "Requirements prioritization in software engineering: A systematic mapping study," in *3rd International Workshop on Empirical Requirements Engineering (EmpiRE)*, 2013, pp. 40–44.
- [17] Y.V. Singh, B. Kumar, S. Chand, and D. Sharma, "A hybrid approach for requirements prioritization using logarithmic fuzzy trapezoidal approach (LFTA) and artificial neural network (ANN)," in *Futuristic Trends in Network and Communication Technologies*, Communications in Computer and Information Science, P.K. Singh, M. Paprzycki, B. Bhargava, J.K. Chhabra, N.C. Kaushal et al., Eds. Singapore: Springer, 2019, pp. 350–364.
- [18] M. Ramzan, M. Jaffar, and A. Shahid, "Value based intelligent requirement prioritization (VIRP): Expert driven fuzzy logic based prioritization technique," *International Journal Of Innovative Computing, Information And Control*, Vol. 7(3), 2011, pp. 1017–1038.
- [19] J.C. Quiroz, S.J. Louis, A. Shankar, and S.M. Dascalu, "Interactive genetic algorithms for user interface design," in *Congress on Evolutionary Computation*, 2007, pp. 1366–1373.
- [20] Y. Zhang, M. Harman, and S.A. Mansouri, "The multi-objective next release problem," in *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, GECCO '07. London, England: Association for Computing Machinery, 2007, pp. 1129–1137.
- [21] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, Vol. 6, No. 2, 2002, pp. 182–197.

- [22] M.H. Marghny, H.M. El-Hawary, and W. Dukhan, "An effective method of systems requirement optimization based on genetic algorithms," *Information Sciences Letters*, Vol. 6, No. 1, 2017, pp. 15–28.
- [23] "A Machine Learning Approach to Workflow Prioritization," in *2019 Systems and Information Engineering Design Symposium (SIEDS)*. Charlottesville, VA, USA: IEEE, 2019, pp. 1–5. [Online]. <https://ieeexplore.ieee.org/document/8735589/>
- [24] J.T. de Souza, C.L.B. Maia, T.d.N. Ferreira, R.A.F.d. Carmo, and M.M.A. Brasil, "An ant colony optimization approach to the software release planning with dependent requirements," in *Search Based Software Engineering*, Lecture Notes in Computer Science, M.B. Cohen and M. Ó Cinnéide, Eds. Berlin, Heidelberg: Springer, 2011, pp. 142–157.
- [25] M. Nazir, A. Mehmood, W. Aslam, Y. Park, G.S. Choi et al., "A multi-goal particle swarm optimizer for test case prioritization," *IEEE Access*, Vol. 11, 2023, pp. 90 683–90 697. [Online]. <https://ieeexplore.ieee.org/document/10223041/>
- [26] P. Avesani, S. Ferrari, and A. Susi, "Case-based ranking for decision support systems," in *International Conference on Case Based Reasoning*. Springer, Berlin, Heidelberg: Springer, 2003, pp. 35–49.
- [27] A. Perini, A. Susi, and P. Avesani, "A machine learning approach to software requirements prioritization," *IEEE Transactions on Software Engineering*, Vol. 39, No. 4, 2013, pp. 445–461.
- [28] P. Avesani, C. Bazzanella, A. Perini, and A. Susi, "Facing scalability issues in requirements prioritization with machine learning techniques," in *13th IEEE International Conference on Requirements Engineering (RE'05)*, 2005, pp. 297–305.
- [29] C. Duan, P. Laurent, J. Cleland-Huang, and C. Kwiakowski, "Towards automated requirements prioritization and triage," *Requirements Engineering*, Vol. 14, No. 2, 2009, pp. 73–89.
- [30] M. Sadiq and S.K. Jain, "Applying fuzzy preference relation for requirements prioritization in goal oriented requirements elicitation process," *International Journal of System Assurance Engineering and Management*, Vol. 5, No. 4, 2014, pp. 711–723.
- [31] L. Zadeh, "Fuzzy sets," *Information and Control*, Vol. 8, No. 3, 1965, pp. 338–353. [Online]. <https://linkinghub.elsevier.com/retrieve/pii/S001999586590241X>
- [32] H. Sadia, S. Abbas, and M. Faisal, "Volatile requirement prioritization: A fuzzy based approach," *International Journal of Engineering and Advanced Technology*, Vol. 8, No. 5, 2019.
- [33] M. Dabbagh and S.P. Lee, "An approach for prioritizing NFRs according to their relationship with FRs," *Lecture Notes on Software Engineering*, Vol. 3, No. 1, 2015, pp. 1–5.
- [34] D. Mougouei, D.M. Powers, and E. Mougouei, "A fuzzy framework for prioritization and partial selection of security requirements in software projects," *Journal of Intelligent and Fuzzy Systems*, Vol. 37, No. 2, 2019, pp. 2671–2686.
- [35] A. Ejnoui, C.E. Otero, and A.A. Qureshi, "Software requirement prioritization using fuzzy multi-attribute decision making," in *Conference on Open Systems*, 2012, pp. 1–6.
- [36] A. Bagnall, V. Rayward-Smith, and I. Whittle, "The next release problem," *Information and Software Technology*, Vol. 43, No. 14, 2001, pp. 883–890.
- [37] J. del Sagrado, I.M. del Águila, and F.J. Orellana, "Ant colony optimization for the next release problem: A comparative study," in *2nd International Symposium on Search Based Software Engineering*, 2010, pp. 67–76.
- [38] D. Greer and G. Ruhe, "Software release planning: an evolutionary and iterative approach," *Information and Software Technology*, Vol. 46, No. 4, 2004, pp. 243–253.
- [39] M. Harman, A. Skaliotis, K. Steinhöfel, and P. Baker, "Search-based approaches to the component selection and prioritization problem," in *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, GECCO '06*. Seattle, Washington, USA: Association for Computing Machinery, 2006, pp. 1951–1952.
- [40] M. Feather and T. Menzies, "Converging on the optimal attainment of requirements," in *Proceedings IEEE Joint International Conference on Requirements Engineering*. Essen, Germany: IEEE Comput. Soc, 2002, pp. 263–270.
- [41] M.O. Saliu and G. Ruhe, "Bi-objective release planning for evolving software systems," in *Proceedings of the 6th Joint Meeting of the European Software Engineering Conference and the*

- ACM SIGSOFT Symposium on the Foundations of Software Engineering*. Dubrovnik, Croatia: ACM Press, 2007, p. 105.
- [42] A. Finkelstein, M. Harman, S.A. Mansouri, J. Ren, and Y. Zhang, “A search based approach to fairness analysis in requirement assignments to aid negotiation, mediation and decision making,” *Requirements Engineering*, Vol. 14, No. 4, 2009, pp. 231–245.
- [43] P. Tonella, A. Susi, and F. Palma, “Using interactive GA for requirements prioritization,” in *2nd International Symposium on Search Based Software Engineering*, 2010, pp. 57–66.
- [44] F. Shao, R. Peng, H. Lai, and B. Wang, “DRank: A semi-automated requirements prioritization method based on preferences and dependencies,” *Journal of Systems and Software*, Vol. 126, 2017, pp. 141–156.
- [45] D. Ameller, M. Galster, P. Avgeriou, and X. Franch, “A survey on quality attributes in service-based systems,” *Software Quality Journal*, Vol. 24, No. 2, 2016, pp. 271–299. [Online]. <http://link.springer.com/10.1007/s11219-015-9268-4>
- [46] A. Al-Adwan and A. Aladwan, “Using Interdependencies for the Prioritization and Reprioritization of Requirements in Incremental Development,” *International Journal of Advanced Computer Science and Applications*, Vol. 11, No. 11, 2020. [Online]. <http://thesai.org/Publications/ViewPaper?Volume=11&Issue=11&Code=IJACSA&SerialNo=29>
- [47] A. Bisht and M. Kushwaha, “Parameter optimization of software requirement by using fuzzy algebra,” 2020.
- [48] S. Abu Saeed, S.U.R. Khan, and A. Mashkoo, “A fuzzy AHP-based approach for prioritization of cost overhead factors in agile software development,” *SSRN Electronic Journal*, 2022. [Online]. <https://www.ssrn.com/abstract=4237372>
- [49] F. Hujainah, R. Binti Abu Bakar, A.B. Nasser, B. Al-haimi, and K.Z. Zamli, “SRPTackle: A semi-automated requirements prioritisation technique for scalable requirements of software system projects,” *Information and Software Technology*, Vol. 131, 2021, p. 106501. [Online]. <https://linkinghub.elsevier.com/retrieve/pii/S0950584920302433>
- [50] X. Dang, Y. Li, M. Papadakis, J. Klein, T.F. Bissyandé et al., “Test input prioritization for machine learning classifiers,” *IEEE Transactions on Software Engineering*, Vol. 50, No. 3, 2024, pp. 413–442. [Online]. <https://ieeexplore.ieee.org/document/10382258/>
- [51] H. Alrezaamiri, A. Ebrahimnejad, and H. Motameni, “Parallel multi-objective artificial bee colony algorithm for software requirement optimization,” *Requirements Engineering*, Vol. 25, No. 3, 2020, pp. 363–380.
- [52] A. Maghawry, R. Hodhod, Y. Omar, and M. Kholief, “An approach for optimizing multi-objective problems using hybrid genetic algorithms,” *Soft Computing*, Vol. 25, No. 1, 2021, pp. 389–405. [Online]. <https://link.springer.com/10.1007/s00500-020-05149-3>
- [53] M. Marghny, E.A. Zanaty, W.H. Dukhan, and O. Reyad, “A hybrid multi-objective optimization algorithm for software requirement problem,” *Alexandria Engineering Journal*, Vol. 61, No. 9, 2022, pp. 6991–7005. [Online]. <https://linkinghub.elsevier.com/retrieve/pii/S111001682100853X>
- [54] N. Tasneem, H.B. Zulzalil, and S. Hassan, “Enhancing agile software development: A systematic literature review of requirement prioritization and reprioritization techniques,” *IEEE Access*, Vol. 13, 2025, pp. 32 993–33 034. [Online]. <https://ieeexplore.ieee.org/document/10876147/>
- [55] R. Anwar and M.B. Bashir, “A systematic literature review of ai-based software requirements prioritization techniques,” *IEEE Access*, Vol. 11, 2023, pp. 143 815–143 860.
- [56] T.A. O’Donoghue, *Planning your qualitative research project: An introduction to interpretivist research in education*. London; New York: Routledge, 2007.
- [57] G.M. James, D. Witten, T.J. Hastie, and R. Tibshirani, *An introduction to statistical learning: with applications in R*, 6th ed., Springer texts in statistics. New York: Springer Science+Business Media, 2013, No. 103.
- [58] J. Kennedy and R.C. Eberhart, “Particle swarm optimization,” in *Proceedings of the IEEE International Conference on Neural Networks*. Perth, Australia: IEEE, 1995, pp. 1942–1948.
- [59] Y. Shi and R.C. Eberhart, “A modified particle swarm optimizer,” in *Proceedings of the IEEE Congress on Evolutionary Computation*. Anchorage, AK, USA: IEEE Press, 1998, pp. 69–73.

- [60] P. Achimugu, A. Selamat, and R. Ibrahim, "Using the fuzzy multi-criteria decision making approach for software requirements prioritization," *Jurnal Teknologi*, Vol. 77, No. 13, 2015. [Online]. <https://journals.utm.my/index.php/jurnalteknologi/article/view/6321>
- [61] J.J. Durillo, Y. Zhang, E. Alba, and A.J. Nebro, "A study of the multi-objective next release problem," in *1st International Symposium on Search Based Software Engineering*, 2009, pp. 49–58.
- [62] P. Achimugu and A. Selamat, "A hybridized approach for prioritizing software requirements based on K-Means and evolutionary algorithms," in *Computational Intelligence Applications in Modeling and Control*, A.T. Azar and S. Vaidyanathan, Eds. Cham: Springer International Publishing, 2015, Vol. 575, pp. 73–93. [Online]. http://link.springer.com/10.1007/978-3-319-11017-2_4
- [63] J.M. Chaves-González and M.A. Pérez-Toledano, "Differential evolution with Pareto tournament for the multi-objective next release problem," *Applied Mathematics and Computation*, Vol. 252, 2015, pp. 1–13.
- [64] B. Kumar, U.K. Tiwari, D.C. Dobhal, and H.S. Negi, "User story clustering using K-Means algorithm in agile requirement engineering," in *International Conference on Computational Intelligence and Sustainable Engineering Solutions (CISES)*. Greater Noida, India: IEEE, 2022, pp. 1–5. [Online]. <https://ieeexplore.ieee.org/document/9844390/>
- [65] K. Lachhwani, "A comprehensive review analysis on PSO and GA techniques for mathematical programming problems," in *Proceedings of International Conference on Computational Intelligence*, R. Tiwari, M.F. Pavone, and R. Ravindranathan Nair, Eds. Singapore: Springer Nature, 2023, pp. 461–476.
- [66] E. Mamdani, "Application of fuzzy algorithms for control of simple dynamic plant," *Proceedings of the Institution of Electrical Engineers*, Vol. 121, No. 12, 1974, p. 1585. [Online]. <https://digital-library.theiet.org/content/journals/10.1049/piee.1974.0328>
- [67] N.R. Pal and S.K. Pal, "A review on image segmentation techniques," *Pattern Recognition*, Vol. 26, No. 9, 1993, pp. 1277–1294. [Online]. <https://linkinghub.elsevier.com/retrieve/pii/003132039390135J>

Authors and affiliations

Rahila Anwar
e-mail: raheelaanwar27864@iqraisb.edu.pk
ORCID: <https://orcid.org/0009-0002-5095-9474>
Computing & Technology Department, IQRA
University, Islamabad, 44000 Pakistan

Muhammad Bilal Bashir
e-mail: bilal.bashir@iqraisb.edu.pk
ORCID: <https://orcid.org/0000-0002-5938-6361>
Computing & Technology Department, IQRA
University, Islamabad, 44000 Pakistan