Received 29 October 2023, accepted 5 December 2023, date of publication 14 December 2023, date of current version 22 December 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3343252

SURVEY

A Systematic Literature Review of AI-Based Software Requirements Prioritization Techniques

RAHILA ANWAR AND MUHAMMAD BILAL BASHIR
Department of Computing and Technology, Iqra University, Islamabad 44000, Pakistan
Corresponding author: Rahila Anwar (rahilaanwar7@gmail.com)

ABSTRACT Software requirements show what the customer desires his software to do. They are the first stepping stone towards a successful software development project. With the increasing complexity of the software due to its size and feature base, it is vital to prioritize the requirements for efficient utilization of development resources. To achieve this, industrial organizations are devising new strategies and improved solutions even with the help of artificial intelligence (AI) tool set. Existing requirements prioritization techniques are human-intensive and suffer from several limitations like overlapping outcomes, scalability problems, time consumption, inaccuracy, and so on. Some of the problems can be solved by including artificial intelligence algorithms and strategies. Several AI-based requirements prioritization techniques have been proposed by applying Genetic Algorithms, Fuzzy Logic, Ant Colony Optimization, and Machine Learning. Literature witnesses some good review studies and surveys on conventional prioritization techniques but there exists none for AI-based techniques that identify not only their strengths but also their weaknesses, advantages of machine learning techniques over other AI-based requirements prioritization techniques, and limitations of applying AI-based techniques in requirements prioritization. This study presents a systematic literature review (SLR) of AI-based requirements prioritization approaches covering 46 papers published from 2000 to 2021. We have given this literature review a new dimension by conducting a parametric analysis of AI-based requirements prioritization techniques and we have identified these parameters after a thorough literature study. Some of the chosen parameters are generic (related to the prioritization process) and some are specific (related to AI techniques). This study has greatly helped us draw a clear line among AI-based techniques to show their domain of application to gain maximum advantage. Our findings will assist researchers, requirement analysts, and other stakeholders in making a wise decision to select the best requirements prioritization technique to gain optimal results.

INDEX TERMS Artificial intelligence, ant colony, fuzzy logic, genetic algorithm, generic parameters, machine learning, optimization, requirement engineering, requirement prioritization, requirement analysis, specific parameters.

I. INTRODUCTION

The Requirement Engineering (RE) is a vital part of software engineering that is characterized as the procedure by which the requirements are determined considering the various needs of clients, understanding the contexts in which the framework would be created, displaying, investigating, arranging, recording stakeholder requirements, ensuring that recorded requirements align with negotiated requirements and dealing with the evolution of changing requirements [14], [32], [51], [57].

Requirement Engineering includes a significant part named Requirement Prioritization that can be used in analysis and negotiation activities [37], [53]. Fig. 1 shows the process flow of the prioritization including inputs, processing, and output. A project's most significant aspects must be prioritized beforehand when it has inadequate resources, an intense execution plan, and high client expectations. At this stage prioritization of requirements becomes necessary. Decision makers face the challenge of surpassing several requirements

The associate editor coordinating the review of this manuscript and approving it for publication was Kostas Kolomvatsos.

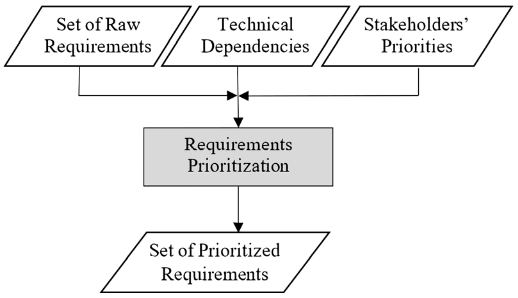


FIGURE 1. Software requirements prioritization process.

than their competency to acknowledge different allocated constraints such as resources, time, and cost [12], [57]. It is necessary to distinguish between key and less significant factors in such situations [37].

Requirement prioritization facilitates the development of project plans so that on-time release of new software product versions is possible. Without a solid strategy, the project may surpass the expected cost [43] as well as the release timetable. The order in which requirements are accomplished influences the end-user and customer's experience, but it also depends on technological restrictions and resource availability. Prioritizing requirements results in an absolute list of requirements that best addresses the various sorts of dependencies [54]. Some decisions regarding the relative priority of requirements or the feasibility of a certain execution sequence must be made by requirement analysts who have prioritizing experience [27], [35], [57].

The usability of software and its commercial value are vastly improved by the volatility of requirements. However, this requirement unpredictability can occasionally lead to misunderstandings and ineffective handling of changing requirements [60]. On the other hand, requirement volatility cannot be ignored because requirements tend to change due to various reasons and if they are not accommodated in the prioritization and development process, the client may refuse to accept the software product.

Many conventional software requirement prioritization techniques have been proposed by authors [12], [14], [29], [32], [37], [43], [47], [51], [55], [57]. These methods are quantitative as well as subjective in nature. Analytical Hierarchy Process (AHP), Cumulative Voting, Numerical Assignment, and others are examples of prominent requirements prioritizing processes. If a few procedures like cumulative voting or ranking are effective for cost and time, their outcomes are not accurate. On the other hand, techniques like AHP are computationally complex, and require expert personnel and a great deal of computations to prioritize the requirements [37].

Existing requirements prioritization techniques suffer from several problems such as complexity of application, expensive in computations, lack of automation, scalability, ambiguity, and uncertainty [12], [37], [47], [53], [59]. Software initiatives might come with several risks, assumptions, and requirements that are incompatible. Unfortunately, conventional prioritization methods are not capable of handling projects with these problems. Most of the aforementioned problems can be resolved with the inclusion of AI-based techniques [24]. For example, fuzzy logic can help in situations where a project has risky and conflicting requirements [44]. Other AI-based techniques that have been applied in prioritizing the requirements include genetic algorithm, simulated annealing, ant colony optimization and machine learning [4], [5], [15], [16], [23], [50], [55], [58].

Literature shows many conventional literature reviews [41], [46] and systematic literature reviews [13], [32], [56], [57], [62] have been carried out in the domain of software requirement engineering and specifically software requirements prioritization. However, to the best of the authors' knowledge, there is no comprehensive literature survey to evaluate AI-based requirements prioritization techniques has been undertaken to date. We conducted a thorough literature review, to sum up the knowledge related to 46 AI-based techniques proposed between the years 2000 and 2021. We have added a new dimension to this systematic review by adding an in-depth parametric analysis of all the proposed approaches in the light of a comprehensive benchmark. We have identified these parameters for the evaluation after an extensive literature survey. These parameters cover every possible dimension of AI-based requirements prioritization techniques in the sense that some parameters cover the inherent properties of the conventional prioritization process and some are specific to the AI-based approaches. This study will help the stakeholders to make the best possible selection of the most appropriate technique to prioritize the requirements of their software. The key contributions of this study include the following;

Literature shows many conventional literature reviews [41], [46] and systematic literature reviews [13], [32], [56], [57], [62] have been carried out in the domain of software requirement engineering and specifically software requirements prioritization. However, no comprehensive literature survey to evaluate AI-based requirements prioritization techniques using generic as well as specific evaluation criteria has been undertaken to date. We conducted a thorough literature review, to sum up the knowledge related to 46 AI-based techniques proposed between the years 2000 and 2021. We have added a new dimension to this systematic review by adding an in-depth parametric analysis of all the proposed approaches in the light of a comprehensive benchmark. We have identified these parameters for the evaluation after an extensive literature survey. These parameters cover every possible dimension of AI-based requirements prioritization techniques in the sense that some parameters cover the inherent properties of the conventional prioritization process and some are specific to the AI-based approaches. This study will help the stakeholders to make the best possible selection of the most appropriate technique to prioritize the requirements of their software. The key contributions of this study include the following;

- It provides a complete list of research publications on AI-based requirements prioritization from the year 2000 to 2021.
- It classifies all the techniques into three major groups and presents a critical review for each of them.
- It analyzes the AI-based requirements prioritization techniques with the help of a well-defined set of parameters.
- It highlights common strengths and weaknesses of all the groups of AI-based techniques collectively as well as individually with concrete justifications.
- It lists all the problem areas that still need further research as future directions for the researchers.
- It presents key advantages of Machine Learning techniques over other AI-based techniques used for software requirements prioritization.
- It analyzes and presents major limitations of applying AI-based techniques in software requirements prioritization.
- This is a unique and "new" research work that did not appear in previous SLR which is a major contribution.

The remaining part of the paper is organized as follows: Section II presents the research method that has been applied to conduct this systematic literature review. Section III presents a detailed survey of Fuzzy Logic, Optimization, and Machine Learning techniques. Section IV and Section V demonstrate the evaluation criteria and Results Analysis and Comparison respectively. Section VI and Section VII present the Discussion and related work of this study respectively. Section VIII concludes this research work on AI-based requirements prioritization techniques, limitations, and future directions.

II. RESEARCH METHOD

This study has been undertaken as an SLR following the standard guidelines of Kitchenham and Charters [13]. To execute the SLR, a review protocol is developed to control the researcher's bias. SLR has been conducted by constituting a team comprising an author and a co-author. After several discussions, we designed research objectives that helped us to articulate three research questions. We selected keywords for searching relevant research papers from online repositories to answer our research questions. The term relevant means that these papers have content related to our domain of study i.e. Requirements prioritization and AI techniques. We designed inclusion, and exclusion criteria, and quality assessment to ensure the quality of research. We extracted all parameters discussed in the literature to evaluate AI-based requirements prioritization techniques and then shortlisted key parameters to design benchmark evaluation parameters for the evaluation of AI techniques used for requirement prioritization.

It consists of research questions, search strategy, study selection, data extraction and synthesis, and threats to validity. The details of the research method are illustrated next with the help of a model presented in Fig. 2.

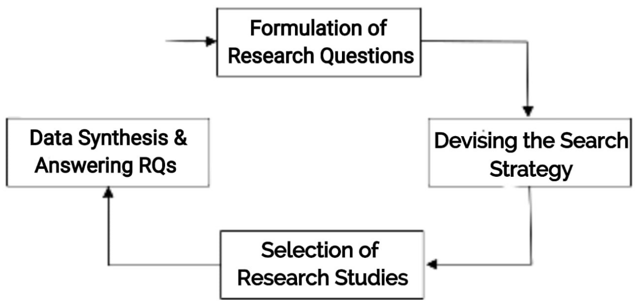

Formulation of Research Questions
Devising the Search Strategy
Selection of Research Studies
Data Synthesis & Answering RQs

FIGURE 2. Research method.

We followed a research method that consisted of four major steps. Firstly, we described the research motivation and questions we wanted to answer. Secondly, we developed a strategy to search for relevant research studies by identifying the most appropriate keywords and preparing search strings to retrieve the necessary material from online sources and databases. We then selected the most relevant research studies to answer the research questions. We set inclusion and exclusion criteria to filter out irrelevant articles and defined quality assurance criteria to obtain high-quality research articles. Finally, we synthesized the data and answered the research questions. To achieve this, we created a parametric benchmark (Section IV) and analyzed all the surveyed approaches in light of that benchmark.

A. RESEARCH MOTIVATION

This SLR aims to analyze AI-based software requirements prioritization techniques that are fuzzy logic, optimization, and machine Learning in the light of parametric benchmark for the core purpose of identifying their key strengths and weaknesses, advantages of machine learning techniques over other surveyed AI-based requirements prioritization techniques and the limitations of applying AI techniques to software requirements prioritization. Various artificial intelligence (AI) techniques are proposed in the literature for prioritizing requirements. However, the trend is towards using machine learning, fuzzy logic, and optimization techniques for this purpose, as per [62]. The primary aim of using AI-based techniques for requirements prioritization is to reduce decision-making effort, thus minimizing errors and accelerating requirement execution, as [43] suggests. Another significant reason for preferring AI-based techniques is that they align with the identified problems of accuracy, scalability, efficiency, redundancy, and lack of optimal solutions.

B. RESEARCH QUESTIONS

We formulated three Research Questions to achieve our research objectives to figure out how important artificial intelligence approaches are in the domain of software requirements prioritization, What has been achieved so far, and what still needs to be done to fully exploit the strengths of AI algorithms. The population of our three Research Questions are published studies from 2000 to 2021. Three Research

TABLE 1. Research questions.

No.	Research Questions	Motivation
RQ1	What are the key strengths and weaknesses of using AI-based techniques for prioritizing software requirements?	To identify state-of-the-art AI-based techniques used for software requirements prioritization and their strengths and weaknesses
RQ2	What are the benefits of using machine learning in comparison to other AI algorithms?	To identify the advantages of Machine Learning software requirement prioritization techniques over other AI-based requirements prioritization techniques. We chose to analyze the difference between Machine Learning and other techniques because ML is state of a state-of-the-art AI technique and the trend of using ML for requirement prioritization is rapidly increasing. There is a dire need to know what edge ML has over other AI techniques when leveraging for requirement prioritization.
RQ3	What are the limitations of applying AI algorithms to prioritize requirements?	To find out limitations of AI-based requirements prioritization techniques to identify future research directions

Questions (RQs) along with their motivation are presented below in Table 1.

C. SEARCH STRATEGY

We devised a search strategy to collect all the research articles related to our domain of study from the online resources. We prepared search strings by merging key terms, then we picked the most famous online resources & databases, and finally, the search was carried out to identify all the relevant articles. Next, we present the details of all the activities performed under this step.

1) SEARCH STRINGS

We followed the instructions provided by Kitchenham and Charters [13] to form the search strings. First, we identified key terms from the research questions. Then we collected synonyms and similar terms that are used in the domain of requirements prioritization. We also used different names for the algorithms that come under the umbrella of artificial intelligence. Then we identified keywords from the relevant research articles [32], [38], [57], [58], [62], [63], [64], [65].

Finally, we used Boolean OR and AND to create the search strings by combining key terms.

The following are the search keywords that resulted.: "Soft-ware Requirements" AND ("Prioritization" OR "Ranking") AND ("Genetic Algorithm" OR "Search based" OR "Meta-Heuristic" OR "Metaheuristic" OR "Fuzzy Logic" OR "Ma-chine Learning" OR "Deep Learning" OR "Artificial Intelligence" OR "Neural Network" OR "Optimization").

The search strings are employed in a variety of ways to locate all work-related research. Guidelines are used to generate search strings. Finally, additional search algorithms and Boolean operators are employed to make the search procedure more robust and to acquire all relevant data. The keywords listed below are used to find similar publications: Requirements prioritization, Requirements Ranking, Optimization Techniques, Genetic Algorithms, Fuzzy Logic, Machine Learning, Metaheuristic, Search-based Techniques, and Artificial Intelligence.

2) RESEARCH REPOSITORIES

According to the study procedure, related research studies were thoroughly retrieved using the selection and rejection criteria. These research articles were gathered from Elsevier, ACM, IEEE, Springer, and Google Scholar.

3) SEARCH OUTCOME

A systematic literature review entails a thorough search of all relevant materials on a particular topic. The search techniques used in this study, however, consist of the phases listed below and are represented in Fig. 3. Stage 1 of the search: A comprehensive search of the five electronic database sources was conducted and the returning results (papers) were compiled into sets of potential papers.

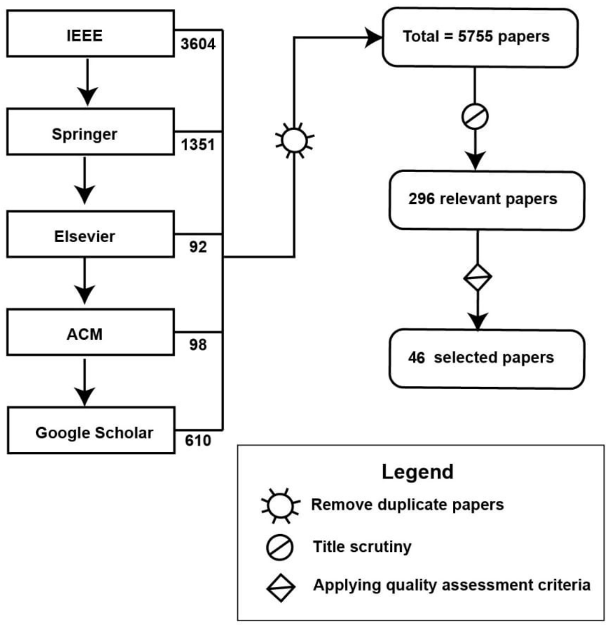


FIGURE 3. Search and selection process.

D. STUDY SELECTION

During the first phase of the search, 5755 prospective studies were discovered. Next, the titles and the abstracts of these studies were scrutinized. This task was critical to eliminate duplicate (same paper published by more than one publisher), grey literature (unpublished work or work in progress), or irrelevant research publications that do not answer our research questions. As a consequence, 296 out of 5755 studies were determined to be relevant and 5459 studies were rejected. Thereafter, using the snowballing process, references of each selected study were searched to identify important studies that might have been missed out during the initial search process but we did not find any recent paper for a new technique. Finally, the quality assessment criteria (QAC) were applied to the selected studies and 46 most relevant studies were chosen to answer our research questions.

1) SCRUTINY

From Fig. 3, during the preliminary search, 5755 prospective studies were discovered. As a result, careful examination was required to narrow down the studies to those that were most relevant. Each study's title was considered first, followed by a brief examination of the contents. As a result, any articles that were not relevant to the discussion subject or were unable to address any of the articulated research questions were deleted from the list of studies. When there are many versions of the same document, the search uses the most thorough, current, and upgraded version, whereas the others are discarded. We conducted a thorough review of requirements prioritization approaches covering papers published between 2000 and 2021. The inclusion and exclusion criteria used for the evaluation of studies are summarized in Table 2.

TABLE 2. Inclusion criteria and exclusion criteria.

No	Inclusion Criteria	Exclusion Criteria
1	The Papers on AI-based software requirements prioritization techniques	The Papers not related to AI-based software requirements prioritization techniques
2	The Papers that discuss parameters for evaluation of AI-based software requirements prioritization techniques	The Papers without evaluation metrics for AI-based software requirements prioritization techniques
3	The papers that are written in English language	Papers that are presented in a language other than English
4	The papers on AI-based requirement prioritization techniques published from 2000 till 2021	The Duplicate papers (Same papers published by different publishers) and grey literature (unpublished work or work in progress) are excluded

2) QUALITY ASSESSMENT OF SELECTED STUDIES

The objective of the critical evaluation of our gathered literature is to check the quality of the studies that were obtained. For study selection, we formulated a three-point quality assessment score (QAS) checklist following the guidelines of Kitchenham et al [13]. Each question has only three optional answers: "Yes", "partly" or "No". These three answers are scored as follows: "Yes" = 1, "Partly" = 0.5, and "No" = 0. Consequently, the quality score for a particular study is computed by finding the sum of all the scores of the answers to the Quality assessment questions. The minimum quality acceptance criteria With a quality score is 2.5. All 46 selected studies qualified for the minimum quality assessment criteria. The Quality assessment questions are presented in Table 3. (For detailed quality scores see Table 33).

TABLE 3. Quality assessment checklist.

No	QA Questions
QAC1	Does the study explain AI-based requirements prioritization technique?
QAC2	Does the proposed study concentrate on the related domain of RQs?
QAC3	Are the evaluation parameters to evaluate AI-based requirements prioritization techniques mentioned and explained?
QAC4	Are the results of the study clearly stated?
QAC5	Is the evaluation of the proposed technique performed on the case study?

E. DATA EXTRACTION AND SYNTHESIS

The data extraction and data synthesis for the selected research articles are shown in Tables 4 and 5 respectively below to discover the answers to three research questions. Table 6. shows evidence from 66 research papers that have been published in different journals, articles, conferences, web pages, and reports. Out of these 66 papers, 46 papers on the topic were chosen to answer the formulated research questions. The overview of 46 selected studies is presented in Tables 26 and 27. The data for RQ1 is arranged into three tables: 28, 29, and 30. In terms of software requirement prioritization, RQ2 discusses the advantages of machine learning approaches over AI techniques. RQ3 examines the limitations of applying AI-based techniques in software requirements prioritization that are presented in Table 31.

TABLE 4. Data extraction and synthesis detail.

No.	Description	Detail
1	Info of name of authors	bibliography, title, publisher detail, publication year, and type of research
2	Overview	An outline of the research
3	Result	Result of Research

TABLE 5. Data synthesis detail.

No.	Description	Detail
1	Strengths and Weaknesses of AI-based Techniques	This SLR also identifies the major strengths of surveyed AI techniques (Section V, Table XXVIII, XXIX, and XXX)
2	The advantages of Machine Learning Techniques	This SLR presents the advantages of Machine Learning techniques over AI techniques relating to the software requirements prioritization (Section V)
3	The limitations of AI-Based Techniques	Software requirements prioritization techniques that are identified from 46 selected papers also have some limitations that are described in Section V (Table XXXI)

F. THREATS TO VALIDITY

In this section, we identified a few known threats to the validity of this study's results.

TABLE 6. Research paper databases and references.

Scientific DB	Type	Selected Research	Total
IEEE	Journal	[27] [43] [50] [57] [62]	5
IEEE	Conference Paper	[47] [29] [3] [10] [16] [21] [11] [8] [58] [56]	10
IEEE	Book	Nil	Nil
Springer	Journal	[63] [24] [31] [60]	4
Springer	Conference Paper	[53] [59] [15] [20] [23] [18] [6] [46] [5] [41]	10
Springer	Book	[55] [34]	2
Elsevier	Journal	[65] [37] [33] [52] [54]	5
Elsevier	Conference Paper	Nil	Nil
ACM	Journal	[17]	1
ACM	Conference Paper	[9]	1
ACM	Book	[14]	1
Google Scholar	Journal	[64] [1] [30] [39] [49] [61] [32] [22] [40] [4] [19] [26] [38] [28] [42] [48] [35] [36] [7] [49] [12] [66]	22
Google Scholar	Conference Paper	[25] [44] [51]	3
Google Scholar	Book	[45]	1
Google Scholar	Web Page	Nil	Nil
Google Scholar	Report	[13]	1

- Our articulated research questions may not include all aspects of AI-based requirements prioritization techniques. This is a construct validity threat and we addressed it by grouping all existing AI-based requirements prioritization techniques. We focused on three groups of fuzzy logic, optimization, and machine learning techniques concerning their methods and building blocks so up to our knowledge and analysis, none of the aspects are missed.
- Incomplete data collection is another threat to this SLR. To avoid this threat, we carefully selected our search keywords to fetch more relevant studies from the research repositories.
- The major threats militating against this review were considered to be publication bias and faulty data extraction. The papers were selected using the previously mentioned search approach which included (a) a variety of literature databases, (b) selection criteria, and (c) quality assessment criteria. To counter this threat, a thorough manual review of all of the retrieved studies' references was conducted to determine any papers that were missed during the original search. In addition, to avoid improper exclusion of desirable studies, a thorough specification of the selection criteria that met the research goals was enforced.
- Inconsistency of data extraction or evaluation of study relevancy is also a threat that occurs when a study's title indicates relevance but the contents do not give answers to any of the research questions. To mitigate the inconsistency of extracted data, the authors completed the independent assessment of the selected studies using quality criteria.
- Performance of existing strategies may be overestimated by researchers who claim that their method is better than others. To avoid this danger, papers comparing several existing procedures were searched out and added to the research to offer an objective evaluation outcome across a variety of methodologies. This is because comparative studies, in the vast majority of situations, provide unbiased reports.
- Inconsistency of data extraction or evaluation of study relevancy is also a threat that occurs when a study's title indicates relevance but the contents do not give answers to any of the research questions. To mitigate the inconsistency of extracted data, the authors completed the independent assessment of the selected studies using quality criteria.
- All metrics are prone to vulnerability. If the type of requirements, type of fuzzy logic approaches, optimization, machine learning techniques, research method, or project nature vary, the outcomes of our research study may change. To mitigate this threat, we examined the nature of the approaches and their core building blocks that remain the same even if the nature of the project changes.

III. SURVEYED TECHNIQUES

In this section, we provide a critical review of all the surveyed techniques. Software organizations are moving towards AI-based techniques for requirements prioritization to resolve problems of conventional techniques. We have classified the techniques into three groups; Fuzzy logic-based techniques, Optimization-based techniques, and Machine learning-based techniques. Next, we present a detailed review of all of the techniques of the aforementioned groups individually.

A. FUZZY LOGIC-BASED TECHNIQUES

It is difficult to quantify requirements into specific numbers. Fuzzy logic uses mathematics to deal with imprecise and linguistic problems. Low-cost, high-quality, and high-progress terms may be specified in a very specific way using fuzzy logic [30]. Lotfi Zadeh [1] proposed the fuzzy sets and fuzzy logic hypothesis in 1965. This is a scientific tool for dealing with uncertainty. It explains unclear methods using human language and assertions that are neither correct nor improper. Following that, we present all of the proposed techniques for ranking software requirements using fuzzy logic.

1) RAMZAN, JAFFAR, AND SHAHID, (2011)

Ramzan, Jaffar, and Shahid [24] propose a novel multi-level technique that is based on the Fuzzy C mean (FCM) method to allocate requirements to each group using a fuzzy membership function. The cost function is minimized when requirements with data near the centroid of their clusters are allocated high membership values and it is maximized when requirements with data far from the centroid are assigned low membership values. The authors conducted a comparative study based on experimental findings from a few university, business, and government-funded initiatives. Due to the involvement of stakeholders and specialists in first and second-level prioritization, this technique is more expensive than some other approaches.

2) MOMENI et al. [31]

Momeni et al. in [31] provide a novel technique based on a neuro-fuzzy system that can give an ideal sequence of quality features essential for developing items that satisfy consumers. In a neuro-fuzzy system, the fuzzy membership function is applied. This sort of matrix determines the link between client needs and quality features. The presented technique demonstrates quality factors like correctness, interoperability, security, understandability, traceability, speed, and time of job completion. According to the findings of this study, the network computation and output production speed is approximately 8% higher than the QFD ranking method. Researchers believe that the neurofuzzy system can give better outcomes than previous studies.

3) EJNIOUI et al. [25]

Ejnioui et al. in [25] prioritize requirements as a fuzzy multi-attribute decision problem in which the anticipated value operator is used to order the options in the issue plan. The suggested approach can deal with inaccurate data by showing distinct quality properties as grey numbers. This approach is simple to use and may be used to investigate what-if scenarios with different weights for different attributes. Because of its flexibility, minimal cost, time efficiency, and ease of use, the proposed approach is particularly useful. However, we do not even observe any experimental based validation of the proposed technique in this research work. The strategy will be validated in the future with the aid of a case study, according to the researchers.

4) SADIQ AND JAIN [30]

Sadiq and Jain in [30] introduce a fuzzy-based methodology for prioritizing requirements in the PRFGORE process by integrating a-level weighted F-preference relation, extent fuzzy AHP for pairwise comparisons of functional and non-functional requirements (FR and NFR), and binary sort tree method to obtain a prioritized list of requirements. The authors claim that previous research does not support organizing requirements when stakeholders' preferences are not expressed in linguistic variables and the MCDM technique is applied in the requirements elicitation process. To solve this problem, a unique fuzzy-based approach for ranking software requirements is presented.

5) JAWALE AND BHOLE [37])

Jawale and Bhole in [37] advocate the use of fuzzy logic with adaptive mechanisms to target scenarios where complicated project behavior might depart from customer expectations. Each object at the highest level of abstraction is then decomposed into more specific levels. The technique of normalization is used to determine the final priorities. The adaptation module monitors and analyzes the results of prioritized requirements, determining whether they are accurate or not. If the requirements are not prioritized appropriately, Fuzzy HCV is applied again. Finally, using the defuzzification method, priorities are defuzzified, yielding final priorities. The authors claim that the Adaptive Fuzzy Hierarchical Cumulative Voting approach overcomes complexity, ambiguity, and uncertainty, resulting in a better decision. However, this technique is not assessed in real-world scenarios, and no tool has been deployed in this study.

6) DABBAGH AND LEE [39]

Dabbagh and Lee [39] propose Fuzzy logic and an alpha cut technique for prioritizing non-functional requirements (NFR) are used to create a *mxn* decision matrix. The relevance degree of each NFR about each FR is determined. This method, according to the researchers, may help practitioners focus on the most essential non-functional requirements early in the life cycle, rather than later, when changes in requirements are often difficult and expensive to implement. This enables soft-ware architects to focus on the most critical non-functional requirements as the major driver of system software architecture design, as well as simplify the selection of appropriate instructions for attaining a software system's required quality attributes.

7) BABAR et al. [35]

The research paper [35] introduces the Priority Handler (PHandler), a requirements prioritization expert system. PHandler uses a value-based intelligent requirements prioritization approach, a neural network, and an analytical hierarchical procedure to make the requirements prioritization process scalable. To remove expert biases and make the PHandler more efficient, a back-propagation neural network is employed to predict the value of a requirement. In addition, to improve the scalability of the requirements prioritization process, the analytical hierarchy approach is used to prioritize sets of requirements.

8) ACHIMUGU et al. [36]

The purpose of this research [36] is to identify the short-comings of present prioritizing systems. Existing approaches are ultimately revealed to have several flaws. The FMADM methodology is used as the strategy for improving prioritization in the world's major projects and infrastructure projects.

Overall, the suggested strategy outperforms the other techniques in terms of computational complexities, accuracy, and large disparities between final ranks among others. This will help to avoid trust, contract, or agreement breaches during the software development process. Based on the reported findings, this research should be regarded as a step forward in the field of multi-attribute decision-making.

9) MISHRA et al. [44]

Mishra, Khanum, and Agrawal [44] have developed a unique multi-level quality-based insightful requirement elicitation technique. The prioritization of requirements is modeled using fuzzy rules in this technique. Five input factors are created, including cost, design, performance, response time, and the number of stakeholders, as well as two output variables, completeness, and understandability, as well as low, medium, and high rankings. According to the authors of this article, changing the input values changes the degree of completeness and understanding capacity. This work may be expanded to include the automatic classification of prioritized requirements into important, essential, and peripheral categories.

10) AHMAD et al. [51]

The MoSCoW method [51] is extended in a fuzzy environment by the authors. They claim that the software requirements engineering community has paid less attention to how to implement the MoSCoW technique in a fuzzy environment for software requirement prioritization (SRP). The suggested technique is used to prioritize the Library Management System's (LMS) requirements utilizing a goal-oriented requirements elicitation method and ranking values determined using an equation (GOREP). Future work might involve prioritizing the LMS when various stakeholders engage in the requirements prioritization process.

11) GULZAR et al. [49]

This study [49] shows a new framework for effectively quantifying conflicts among usability requirements for prioritization. The Mamdani tool is used to compare the efficiency and learnability of member functions. The suggested system is tested using an Electronic Healthcare System as a case study. To assess the suggested technique, a statistical analysis is carried out. Project managers, software engineers, requirement engineers, team leaders, and requirement analysts are among the 20 specialists who make up the team. Their feedback is recorded and the final findings are graphed.

12) MOUGOUEI et al. [61]

Mougouei et al. [61] present a fuzzy system to alleviate the adverse effect of ignoring security requirements and upgrade the precision of prioritization and selection. The proposed system (PAPS) contains two significant procedures. Pre Pre-prioritization and Selection (Pre-PAS) incorporates demonstrating, portrayal, and investigation of security requirements. They use fuzzy control language (FCL) to create the fuzzy inference rules. PAPS' $O(n)$ linear complexity is a good indicator of its scalability. Bubble Sort $0(n2)$ and Binary Search Tree $0(nlogn)$ [32]. Similarly, any prioritizing method based on pairwise comparisons is more challenging than PAPS, requiring at least $O(n2)$ in complexity. Other prioritizing and selection approaches, such as Cumulative Voting, EVOLVE, and Planning Game [45], have not been quantitatively compared to PAPS since their complexity has not been accounted for in the literature.

13) SADIA AND FAISAL [60]

Sadia and Faisal [60] suggest a fuzzy logic-based volatile requirements prioritization approach. Fuzzy logic is powerful for dealing with the elusiveness and granularity of information. Volatile Requirement Priority Ranking is the result of the fuzzy inference algorithm (VRPR). The greater the VRPR score, the more important the volatile requirement is. Fuzzifying inputs, implementing fuzzy operators, applying implication methods, aggregating outputs, and defuzzifying results are the five processes in the fuzzy inference system. Three linguistic variables make up the input phase of the proposed model. Volatile Requirement Priority Ranking is the only linguistic variable in the output phase (VRPR). Using suitable membership functions, the input and output variables are translated into fuzzy sets. Volatile Requirement Priority Ranking is the result of the fuzzy inference algorithm (VRPR). The greater the VRPR score, the more important the volatile requirement is. Because requirements change during the development life cycle, prioritizing these changing requirements will undoubtedly aid project managers in providing a better approach to dealing with these changing requirements.

14) SINGH et al. [59]

Singh et al. in [59] propose a hybrid strategy that combines logarithmic fuzzy preference programming (LFPP) and artificial neural networks (ANN) to prioritize a customer's requirements based on numerous factors at a reasonable cost. Multiple criteria are entered as part of the information. The data is entered into a pre-assembled MATLAB programming. Experts working with the database decide the appropriate hardware and assembly equipment and save it in the database. In addition, the constructed Neural network assesses the appropriateness or inappropriateness of the decision-maker's actions (DM). This research model is developed in MATLAB to determine the priority weights of alternatives and it is then tested using a case study to identify the best washing machine provider. The suggested model using LFPP and ANN, according to the authors, addresses these current constraints by utilizing priority weights for primary qualities and 14 sub-attributes to increase the result's accuracy in the selection of the best provider.

15) BISHT AND KUSHWAHA [64]

In [64], the authors have provided multi-level value-based intelligent requirements prioritizing approach applying fuzzy logic as a facilitating process. They redefine the term "value" software to better achieve companies' goals. Prioritization is attained from the standpoint of stakeholders and specialists in the first and second levels. In the third level of prioritization, fuzzy logic is employed to enhance prioritizing outcomes and reduce the manual nature of the results.

The system's different variables (both input and output) are defined in the first phase. Two input variables, requirement value, and stakeholder priority, and one output variable, requirement priority are used for the proposed problem. Variable sets are defined for all of these inputs and their consequences. Using the proper membership function, the values in these sets are fuzzified. For the proposed method, the trapezoidal fuzzy membership function is used. The fact that the maximum or minimum point in a specific problem circumstance is not simply one number. This is one of the main reasons to employ this particular function. In addition, a comparison analysis based on experimental data from different projects is given. This research indicates that intelligent requirement prioritizing may produce better and more spectacular results in all situations.

B. OPTIMIZATION BASED TECHNIQUES

This section covers Optimization based requirements prioritization techniques proposed by various authors till 2021. Optimization algorithms are well suited for system requirements considered as the next release problem (NRP) to get the best attainable set of requirements that produce an optimal solution [2]. An optimization algorithm is a mechanism for comparing multiple solutions iteratively until an optimal or most suitable one is found [44]. To improve system requirements, researchers use three distinct meta-heuristic search techniques: Genetic Algorithm, Simulated Annealing, and Ant Colony Optimization (ACO) [21].

1) BAGNALL et al. [2]

Bagnall et al., [2] develop Exact, Greedy, and Local Search Techniques to tackle the next release problem (NRP) experimented with MoCell and PAES [22]. Firstly, they apply an exact approach to solve a linear programming relaxation of the problem to derive upper limits. After that, the model is strengthened into a full integer programming model, and a generic branch and bound method is used. Second, they create a collection of simple greedy algorithms for generating lower limits quickly. To improve the value obtained, the best of these procedures are then augmented along GRASP lines. Finally, they use a basic neighborhood structure to apply a set of conventional local search strategies, such as the hill climbing algorithm and the simulated annealing algorithm, to provide a high-quality, near-optimal solution in a short amount of time. The findings of the comparison analysis show that the simulated annealing method outperforms both greedy algorithms and hill climbers and that exact techniques are better for the smallest problems, whereas simulated annealing algorithms are best for the rest. Furthermore, for larger problems, simulated annealing may fail to produce an optimum solution in an acceptable amount of time. Despite these findings, more research into both the heuristic and exact methodologies is needed.

2) FEATHER AND MENZIES [3]

Feather and Menzies [3] use Simulated Annealing and an iterative model (DDP) for the selection of requirements and the optimization problem known as Defect Detection and Prevention. In large-scale requirements models, this unique methodology provides a mechanism to converge to near-optimal solutions. It also indicates crucial decision points throughout the approach, allowing specialists to infuse extra information and help. To randomly sample the space of solutions, a requirement interaction model is used. This generates a vast quantity of data, which is subsequently analyzed using a tool. As a consequence, a short list of key decisions that lead to the desired outcome. The requirement interaction modeling framework proposed in this research is based on NASA's Defect Detection and Prevention (DDP) strategy and tool for risk assessment, planning, and management. DDP is concerned with requirements, risks, and risk management. The results of the pilot research, as well as arguments for its wider application, indicate that this technique requires further research.

3) DEB et al. [4]

Deb et al. in [4] suggest the non-dominated sorting genetic algorithm II (NSGA-II), a multi-objective EA (MOEA) based on non-dominated sorting that addresses three problems. In particular, a fast non-dominated sorting technique with high computing complexity is presented. A selection operator is also available, which creates a mating pool by combining parent and offspring populations and selecting the best (fitness and spread) solutions. The authors of the proposed NSGA-II substitute a crowded-comparison method for the sharing function technique, which mitigates both of the above problems to some extent. The novel approach does not require any user-defined parameters to maintain diversity across population members. To determine the density of solutions surrounding a specific solution in the population, the average distance between two sites on either side of this point along each of the goals is computed. Simulation results of the limited NSGA-II are compared to another restricted multi-objective optimizer on a variety of test problems, including a five-objective seven-restricted non-linear problem and NSGA-II is found to perform substantially better. In addition, the suggested method has a lower computing complexity.

4) GREER AND RUHE [7]

Among all the methods applying genetic algorithms to rank software requirements, the EVOLVE technique encourages continual software development planning [7]. This method uses an iterative optimization method with a genetic algorithm as a backbone. To demonstrate the feasibility of the suggested technique, the authors utilize a sample software project with 20 requirements. In this study, Palisade's

Risk Optimizer tool is used for numerical analysis. Authors claim that the most suitable method is the 'order' method as they are concerned with requirements' ranking. This approach generates different stages of permutations of an initial outcome and is intended to improve the rankings of objects. Only solutions situated at the convex hull in the goal space are identified in the case of non-convex problems, which is a restriction of this technique. However, our primary goal is to generate a (limited) selection of viable ideas from which decision-makers might ultimately select.

5) HARMAN et al. [9]

Harman et al. in [9] explain that the Component Selection Problem is an optimization problem in which the managers prioritize the components under consideration. When planning the software development process, this problem assists the management in determining which component combinations will create logical product choices, whereas the Component Prioritization Problem assists in determining which components should be developed first. This problem, like the majority of real-world optimization problems, requires the optimization of many objective functions. The component selection problem may be framed as an optimization 0-1 knapsack problem with a certain bound of total cost, according to the researchers. As a result, they recommend using search-based heuristics to solve individual knapsack problems to sample the Pareto curve as nearly as feasible.

6) BAKER et al. [10]

This paper [10] presents the application of automated approaches in the domain of requirements engineering to rank software requirements. It introduces greedy and simulated annealing techniques to identify optimum solutions and it formulates the problem as a sequence of feature subset selection problems. This research study evaluates the technique for selecting and ranking forty potential software components for mobile telecommunication devices using real-world data from a large global telecommunications corporation. Real data has been anonymized and domain-specific information has been deleted for security concerns. However, data values have not been changed so all reported results are authentic and reproducible. Following the studies, the authors compare the outcomes of greedy and simulated annealing methods with expert judgment prioritization results. The simulated annealing algorithm's findings are shown to be consistent over many distinct executions with identical parameter values, as well as across numerous runs with variable parameter settings. The results demonstrate that the simulated annealing method outperforms the greedy method. The full set of experiments to achieve a ranking using the greedy technique took virtually no time at all on conventional computer equipment, but the simulated annealing procedure took around 30 seconds on average.

7) SALIU AND RUHE [11]

Saliu and Ruhi [11] propose a decision-making tool for the RP of developing software systems that consider both business and implementation factors. The RP problem was reformulated as a bi-objective optimization problem with trade-offs between the two points of view. Pareto-optimal solutions provide alternate release plans rather than a single response, which improves decision-making. This is partly because no formal model can fully capture all of the choice parameters in a human-centric decision problem. Release Planning (RP) includes selecting what new features and change requests should be executed in which release of a software system, as well as when they should be implemented.

The proposed method for identifying SD-coupling is independent of the granularity with which components are defined, ensuring the technique's scalability. The only restriction is that components must be described at the same granularity level across all software projects. Some criteria may have structural, technical, or functional correlations that must be met simultaneously or individually in specific settings. Requirements Interaction is the study and management of inter-dependencies between requirements.

8) QUIROZ et al. [15]

Quiroz et al. [15] use a collaborative and interactive genetic algorithm (IGA) to design user interfaces in the XUL interface defining language by integrating some heuristic GUI design metrics with input from the user. Incorporating user input into IGAs and employing evolutionary approaches for UI design were two problems that researchers tackled. Individuals in the population are represented by interface specifications and fitness is calculated using a weighted combination of user input and design guidelines. Users can successfully lead evolution toward user interface designs that reflect user preferences as well as calculated guideline metrics, according to the findings of a pilot research involving three users. Furthermore, the authors of this study claim that their method lowers user fatigue by requiring users to select just two people from a group of nine for each t generation of the IGA.

9) ZHANG et al. [16]

In [16], the authors examine the appropriateness of weighted and Pareto optimum evolutionary algorithms for the Multi-Objective Next Release Problem (MONRP) in Requirements Engineering. They also provide a non-dominated sorting genetic algorithm (NSGA-II) and attempt to demonstrate that it is suitable for solving the MONRP. These two strategies perform well to solve the scalability concerns that techniques like AHP have, but they do not generate a full requirements' ordering. Rather, requirements are categorized for the future release's preparation. In terms of paired assessments, their suggested technique employs IGA to restrict the amount of knowledge that must be extracted from the user. As a result, the technique may be used for requirement sets of any size. The findings of an experimental study into the appropriateness of weighted and Pareto optimum genetic algorithms, as well as the NSGA-II algorithm are presented in the article, demonstrating proof to strengthen the argument that NSGA-II is appropriate for the MONRP. The research also includes benchmark data to show when the MONRP becomes non-trivial at different sizes. The authors, on the other hand, do not provide any information regarding tool support.

10) FINKELSTEIN et al. [17]

Finkelstein et al. in [17] provide a multi-objective optimization technique to help in the analysis of trade-offs in many clients' diverse ideas of fairness. The framework suggests that each idea of fairness should include a goal in a multi-objective, Pareto optimal Search-Based Software Engineering (SBSE) context. Pareto optimality is well-suited to this scenario since it makes no assumptions about which goal takes precedence. The method presented in the paper may be used to see if a solution is fair under all definitions of fairness. Multi-objective search methods are based on the dominance concept to solve MOOPs. In the algorithms, two solutions are compared to see if one of them outperforms the other. The NSGA-II method is applied to the Fairness in Requirement Assignments Problem to identify the Pareto front in various circumstances. The Two-Archive method uses the Convergence Archive (CA) and the Diversity Archive (DA) to gather and record candidate solutions in the population. This study presents the findings of a comparison between Random Search and two more sophisticated, evolutionary multi-objective search methods. The findings support the overall approach, indicating that more sophisticated techniques outperform Random Search. The results also demonstrate that the outcomes of the more sophisticated techniques are encouragingly stable in terms of performance and slightly complementary in terms of diverse solutions.

11) DURILLO AND ZHANG [18]

Durillo and Zhang in [18] examine the NRP problem from a multi-objective perspective, concentrating on the number of solutions identified, the variety of solutions covered by these fronts, and the number of optimum solutions found. They also compare the performance of two state-of-the-art multi-objective metaheuristics for solving NRP, NSGA-II, and MOCell. The NSGA-II algorithm uses a ranking approach to extract non-dominated solutions from a population and give them a one-star grade. The rank of the next set of non-dominated solutions is 2 and so on.

The authors utilize two indicators in this paper: one that measures solution diversity and the other that assesses both convergence and diversity. To evaluate the algorithms' search skills, each experiment is repeated 100 times to determine the mean and standard deviation, which are used to measure central tendency (or location) and statistical variance, respectively. They use the Wilcoxon test to compare samples in a variety of ways. MOCell outperforms NSGA-II in terms of the number of solutions it can generate, although NSGA-II can provide better solutions in large instances than MOCell. Furthermore, the authors discover that the best solutions are made up of a significant percentage of low-cost criteria as well as the requirements that satisfy customer needs most.

12) TONELLA et al. [20])

Tonella et al. in [20] present an Interactive Genetic Algorithm (IGA) for intelligent requirements prioritizing that integrates incremental information securing and consolidation with previously existing limitations of dependencies and priority. The authors compared IGA against state-of-the-art interactive prioritizing techniques and the Incomplete Analytic Hierarchy Process on a real software system as part of the ACube (Ambient Aware Assistance) project (IAHP). According to them, IGA outperforms other state-of-the-art interactive prioritizing methods.

13) SAGRADO et al. [21]

In [21], the Next Release Problem (NRP), as an optimization problem, is described. In various case studies, the study uses three distinct methods to solve NRP: simulated annealing, genetic algorithms, and ant colony optimization (tailored to the NRP problem). The greedy randomized adaptive search method (GRASP), genetic algorithm (GA), and Ant colony system (ACS) are evaluated using two data sets and parameter configurations. The authors ran 100 separate runs for all data sets to test each method. In the first run, ACS and GRASP found the same solution, but when more requirements were added, the GA algorithm found the worst solution in terms of satisfaction when compared to ACS and GRASP. This, according to the authors, was due to the design effect. GRASP discovered the best solution in terms of satisfaction on the second dataset, followed by ACS with somewhat worse solutions. It was projected that if parameter *beta* was set to a greater value, the outcomes would be similar to GRASP, strengthening ACS' greedy behavior. The findings of the paper show that the solutions discovered by the proposed approaches contain a significant number of user-defined criteria for a certain system release.

14) DE SOUZA et al. [23]

De Souza et al. [23] use an Ant Colony Optimization (ACO) technique to address the next release problem in the existence of dependent requirements. The authors tested their suggested method on 72 synthetic datasets and compared it to simulated annealing and evolutionary algorithms. The findings show that the proposed ACO algorithm may give more accurate solutions to the Software Release Planning problem when compared to the other two techniques. In the genetic algorithm, a basic heuristic method is utilized to provide significant solutions for the initial population, whereas single-point crossover and mutation operators are employed to generate genuine solutions. A binary tournament is utilized as a selection method. A legitimate neighborhood operator is also used for the simulated annealing. The Wilcoxon Rank

Sum Test is used to evaluate the statistical significance of the outcome differences generated by each pair of algorithms, with significance levels of 90%, 95%, and 99%. The authors assert that under these conditions, the outcomes generated by ACO are essentially superior to those generated by a genetic algorithm in all situations.

15) TONELLA et al. [27]

Tonella et al. in [27] propose an Interactive Genetic Algorithm (IGA) for requirements prioritization. It belongs to the a posteriori approaches category and makes use of the paired preference elicitation method as a successful way to obtain relevant data from the user. Different limits and ranking criteria are represented in the requirement document for creating such user information (in the form of requirement qualities). At an increasing user rate, the difference in variance between IGA and IAHP is compared. The authors used statistical tests (ANOVA) to test their claim. The proposed method is compared to the state-of-the-art interactive prioritization approach Incomplete Analytic Hierarchy Process (IAHP) in a real case study to validate it. They claim that the proposed algorithm aims to deliver an exact requirement sequence while keeping requirement analysts' decision-making effort linked to pairwise comparison affordable.

16) CHAVES-GONZÁLEZ AND PÉREZ-TOLEDANO [40]

The authors employ an updated multi-objective version of the differential evolution (DE) evolutionary algorithm to cope with many real-world examples of the software requirements selection problem in this research paper. A series of experiments with case studies on software requirements selection are conducted to demonstrate the effectiveness of the multi-objective proposal and the results show that the proposed algorithm outperforms other applicable algorithms previously presented in the literature on a set of publicly available datasets. Because the results are insufficient in terms of efficacy, efficiency, and robustness, the authors of the paper [40] offer a multi-objective evolutionary algorithm to address these difficulties. The DEPT algorithm's findings are compared to the multi-objective standard NSGA-II (Fast Non-dominated Sorting Genetic Algorithm) and other techniques presented in previous works published in the field to assess the correctness of the proposed methodology. For numerous cases of the software requirements selection problem, the DEPT method has delivered high-quality results that outperform those achieved in prior research investigations.

17) VALSALA AND NAIR [43]

Valsala and Nair [43] propose a model that focuses not only on requirements prioritization but also on the scheduling of prioritized requirements. Scheduling the prioritized requirements is very important as this leads to minimizing the cost and development time of the project. To solve premature convergence concerns, the Enriched Genetic Algorithm (EGA) is used to prioritize requirements, and a heuristic Revamped Integer Linear Programming (RILP) model is used for scheduling. It is critical to figure out how to limit the number of requirements subsets in the most efficient method possible. The authors employ the Expectation Maximization (EM) technique for this. Requirement dependencies and the project's period are two sorts of metrics utilized in RILP. The authors compare simulations of both models and find that the Hybrid EGRILP model obtains a very optimal solution when compared to the enriched genetic algorithm and the Revamped ILP algorithm for software requirement prioritization and scheduling. The proposed software release planning method is more efficient in terms of project span with minimum delay and maximum profit.

18) KUMARI AND SRINIVAS [42]

In this paper [42], the software requirements selection problem is empirically evaluated using the Quantum-inspired Elitist Multi-objective Evolutionary Algorithm (QEMEA), Quantum-inspired Multi-objective Differential Evolution Algorithm (QMDEA) and Multi-objective Quantum-inspired Hybrid Differential Evolution (MQHDE). All of the algorithms are created on a Windows 7 platform. The basic model is investigated using six data sets, whereas the MONRP RIM model is investigated with four data sets. The MONRP problem is examined using QMDEA, QEMEA, and MQHDE. These algorithms' results have also been compared to the NSGA-II and other publicly available techniques. The comparison is made using the obtained Pareto fronts, performance metrics like Hypervolume (metric for both convergence and diversity), Spread (diversity metric), Generational Distance (convergence metric), Capacity (size of Pareto front - representing the number of alternative solutions provided by the algorithms) and extreme Pareto optimal solutions. With a 95% confidence level, the KruskalWallis non-parametric hypothesis test is used to statistically assess all of the measures.

19) MARGHNY et al. [50]

Marghny et al. in [50] provide a non-dominated sorting genetic algorithm with a Pareto tournament for multi-objective optimization (NSGA-IIPT) approach for ranking software requirements after taking into account their interaction and cost. The proposed technique adjusts the clients' preferences and cost criteria to find high-quality solution sets within a particular implementation cost constraint. To demonstrate the effectiveness of the suggested technique, the authors use eight different scenarios drawn from two real-world datasets. These data sets both contain a varied amount of requirements, requirement interactions, and client preferences, and both have recently been used in prior published works, allowing researchers to contrast the results of the current research to those of previous research. The proposed technique, NSGA-IIPT, is compared to the outcomes of prior studies (DEPT, ACO, NSGA-II, and GRASP). According to the authors, NSGA-IIPT has proven to be capable of outperforming its peers in terms of best exploring the search space.

20) RAO et al. [48]

The aim of this work [48] is to optimize process parameters to attain these goals. The multi-objective Jaya (MO-Jaya) approach is a revolutionary posteriori multi-objective optimization technique that can handle several goals simultaneously and generate many optimal solutions in a single simulation session. The fitness function of the MO-Jaya algorithm is based on regression models created by earlier researchers for the machining operations. The MO-Jaya algorithm applies a heuristic strategy called as constrained-dominance concept to adequately handle the restrictions. Simulations utilizing NSGA, GA, NSGA-II, NSTLBO, BBO, PSO, SQP, and Monte Carlo are compared to the MO-Jaya algorithm's findings. According to the findings, the recommended technique outperforms the others. In MATLAB R2009a, a computer code for the MO-Jaya method is created. The program is run on a computer system with a 2.93 GHz processor and 4 GB of RAM. The MO-Jaya algorithm's results are compared to the NSTLBO, SQP, and MC simulations. Prior research has indicated that the results achieved with the MO-Jaya approach are superior to those obtained with other techniques such as NSGA, GA, iterative search, and BBO.

21) AHUJA et al. [55]

Ahuja et al. [55] provide a novel strategy that, when compared to the previous interactive genetic approach, improves performance by employing a least-squares-based random genetic algorithm. The goal of this study is to help engineers prioritize requirements by saving time and thereby lowering decision-making effort. As the number of pairings rises in the proposed task, the distance increases marginally. This mathematical method is implemented to investigate the performance gap between pairs. The authors argue that, based on performance comparison findings of all nine ways listed, VOP (value-oriented prioritizing) is the most effective methodology as it produces non-erroneous results and uses simple procedures. Furthermore, this can assist us in judiciously handling additional requirements.

22) ALREZAAMIRI et al. [63]

To increase the quality of the results, the authors develop a parallel method in [63] based on the main–secondary concept. The algorithm is divided into three basic phases, each of which is linked to the activity of a certain species of bee. There are three categories of bees in this algorithm: employed, spectator, and scout bees. Employed bees seek the surrounding area for new food sources and go back to the hive if they find any food location. They use dancing displays to inform spectator bees about the position of the food supply. Following the information obtained from the hired bees, each observation bee in the hive chooses a food source at random to conduct additional searches. The more abundant the food supply, the more likely observing bees will choose it. The scout bee is the third kind of bee. Scout bees are hired bees who already have abandoned their food supply. In pursuit of a new food source, they move about the territory at random. The proposed approach is implemented in MATLAB version R2014b. According to the findings, the suggested approach greatly improves the quality of solutions in the first case. In addition, in the second scenario, the technique reduces execution time by enhancing the quality of the output. Other comparable research, such as NSGA II, GRASP, and ACS, have found results that are superior to those achieved by the ABC algorithm. For the most challenging data set managed, the proposed technique can obtain an HV of above 60%, whereas other existing methods can only reach an HV of 40% for the same data set.

C. MACHINE LEARNING BASED TECHNIQUES

This section covers machine learning-based requirements prioritization techniques proposed by various authors. Machine learning is a subfield of artificial intelligence (AI) that enables computers to learn and evolve without being explicitly programmed. In comparison to other AI-based technologies, machine learning has not been widely researched in the domain of requirements prioritization. A recent systematic literature review on requirement prioritization techniques and their empirical evaluation [62] reports only 4 recent publications out of 102 evaluated studies that exploit machine learning techniques for requirements prioritization but this trend of leveraging machine learning techniques is increasing rapidly [62], [66] due to their potential advantages that are discussed in analysis and discussion section.

1) AVESANI et al. [5]

Avesani et al. in [5] design a case-based elicitation process to help with the rank evaluation planning problem. The authors employ a paired method in which preference elicitation is based on a comparative analysis of two options. They propose a simple iterative methodology based on two automated stages: case selection strategy development and approximation of case preferences. The former seeks to make a pairwise comparison technique easier, whereas the latter seeks to reduce elicitation time. Boosting is a machine learning method that improves the threshold on the size of the case base while reducing the cognitive load on the end user. This method seeks a balance between elicitation effort and accuracy. The authors used boosting methodology to combine multiple weak rules that could be somewhat accurate to build extremely accurate prediction rules.

The case-based preference elicitation model is evaluated by discriminating between online and offline studies. The off-line tests are created to evaluate the properties of the process as well as the offered solutions. The online tests are being carried out as a case study on a core issue with real users in the civil defense area. The results of the case study show that when the learning phase is with end-user elicitation, a case-based strategy can be sustained over a large number of examples and that the rank boost algorithm is efficient in achieving a good trade-off between final rank precision and elicitation effort.

2) AVESANI et al. [8]

Avesani et al. in [8] propose a case-based ranking framework for requirements prioritization that employs machine learning algorithms to solve the scalability problem. The authors compared their findings to those of AHP in terms of expert elicitation effort and requirement prioritization accuracy. The preference elicitation technique is based on the examination of two choices, using a paired approach. This paper proposes a simple iterative technique based on two automated processes for developing a case selection method and approximating case preferences. Experiments with simulated data and a real-world situation in the field of civil defense show that a reasonable trade-off between the accuracy of predicted preferences and the end user's elicitation effort may be achieved. The aim of this research is on the ex-post method, in which genuine situations (i.e. alternatives) play an important part in preference elicitation. The authors give experimental proof that a rankboost method is successful in achieving a favorable trade-off between final rank accuracy and elicitation effort.

3) DUAN et al. [19]

Duan et al. in [19] propose a technique for prioritizing requirements based on stakeholders' business goals, interests, and cross-cutting issues like security and performance requirements, utilizing data mining and machine learning methods. Two case studies based on the Ice Breaker System's requirements, as well as a collection of stakeholders' raw feature requests gathered from the SugarCRM discussion forum, are used to demonstrate and assess the approach's success. The Spherical Kmeans (SPK) clustering method, which outperforms hierarchical alternatives, is used in the implementation phase.

The limitations in the method are directly related to the limitations of the underlying classification, traceability, and clustering algorithms, which are all based on probabilistic data mining and information retrieval techniques and hence could not produce perfect precision or recall in the findings. The NFR classifier is a data mining tool that detects and categorizes a variety of non-functional requirements (NFR) in the areas of performance, security, and performance. The classifier is based on the concept of weighted indicator phrases, which weighted each potential phrase based on how effectively it indicated the existence of a certain NFR type.

4) PERINI et al. [26]

Perini et al. [26] portray the Case-Based Ranking (CBRank) method that joins stakeholders' inclinations with requirement ranking approximations. CBRank features are empirically evaluated on simulated data and compared to a state-of-the-art prioritizing approach, demonstrating the method's capacity to enable the management of the trade-off between elicitation effort and ranking precision and utilization of domain knowledge. These experimental observations are supplemented by a case study on a real software project. The authors conducted a controlled experiment with 23 participants to compare two tool-supported versions of CBRank and AHP in terms of ease of use, time consumption for completing the job, and accuracy of final ranking. SCORE, a web-based application that implements CBRank's three-step prioritizing approach, is used to carry out the prioritization process. In terms of trade-offs between expert elicitation effort and requirement prioritization accuracy, the authors claim that their approach outperforms AHP in the worst-case scenario.

5) ACHIMUGU AND SELAMAT [34]

In this paper [34] create a hybridized algorithm that employs requirement preference weights derived from stakeholder linguistic evaluations. The RALIC dataset, which contains requirements with relative weights of stakeholders, is used to verify this technique. The goal of this study is to create a better-prioritized strategy to address scalability, rank reversals, and computing complexity. Clustering/evolutionary-based methods are used in this study. By calculating the maximum, minimum, and mean scores, the suggested approach may also classify large amounts of requirements efficiently by reducing conflicts between prioritized requirements. This will help software developers choose the most important and least important requirements, which will help with software release planning and reduce contract, trust, and agreement violations. This research can be considered a step forward in the area of computational intelligence based on the findings. To cluster requirements and assess their relative importance, a hybrid technique based on differential evolution and the k-means algorithm is used. It tries to combine the advantages of two methods by combining k-means to build the initial solution and differential evolution to improve it.

6) SHAO et al. [52]

In [52], for prioritization, DRank, a more realistic technique, is presented. To make the method of selecting ranking criteria easier and more practical, a prioritizing evaluation characteristics tree is created. The subjective requirements prioritizing according to stakeholder preferences is calculated using RankBoost, which makes assessing the prioritization easier. A weighted PageRank algorithm is proposed to assess requirement dependencies, allowing objective dependencies to be automatically transformed into partial order relations and an integrated requirements prioritization method is developed to amend stakeholders' subjective preferences with objective requirements dependencies, resulting in a more efficient prioritization process. To address the differentiating properties of contribution dependencies, a PageRank-Req method is presented.

A controlled experiment is conducted to validate the performance of DRank based on comparisons with Case-Based Ranking, Analytical Hierarchy Process, and EVOLVE.

The authors suggest a variety of tools such as TAOM4E, OpenOME, RE-Tools, and GR-Tool. The experiment is being carried out at a Software Company on two projects: a Book Trading System (BTS) and a Library Management System (LMS). According to statistical analyses using a Shapiro–Wilk W test, DRank outperforms CBRank, EVOLVE, and AHP (top matching rate for requirement prioritization method) and IR (inverse rate of requirement prioritization method) in terms of TMR accuracy, though the difference is not significant in some cases. DRank takes less time and is more reliable than other methods, according to the data. The accuracy of the final priority sequence can be improved by taking requirement dependencies into account, according to a simulated experiment.

7) GUPTA AND GUPTA [54]

The major purpose of this study [54] is to introduce a semi-automated dependency-based collaborative requirement prioritization approach (CDBR), which employs an execute-before-after (EBA) link between requirements, linguistic values, and a machine learning algorithm to reduce stakeholder and developer disagreements and improve final priority approximation. The proposed method focuses on three major problems that have received little attention in earlier research: scalability problem, requirements dependencies, and stake-holder and developer communication. To produce final acceptable implementation priorities, CDBR uses the Particle Swarm Optimization (PSO) technique to reduce disagreements between stakeholders and developers' ranking.

In the first example, nine distinct requirement sets (with various requirements) are utilized to evaluate the suggested approach's performance in terms of managing scalability. The priority of stakeholders and developers for each of these nine sets is determined at random. The developer's priority is calculated using a dependency matrix that is likewise created randomly while keeping the density of the matrix in mind. The higher the density, the more dependencies there are in the system, and hence the more complicated it is. In the second scenario, a set of requirements for cargo booking management in a warehouse (CBMW) is chosen as a case study and CDBR, interactive genetic algorithm (IGA), and analytical hierarchical process are used. The precision of the findings is determined by comparing the difference in disagreement between the CDBR-AHP and CDBR-IGA priority lists using the Analysis of Variance (ANOVA) test technique. The fraction of disagreement between CDBR-AHP and CDBRIGA determines the accuracy of the findings. The results are accurate and equivalent in terms of accuracy, scalability, and stakeholder and developer disagreement levels. In terms of efficiency and processing time, CDBR exceeds AHP and IGA.

8) BOLLUMPALLY et al. [58]

This model presented in [58] employs the XGBoost method, which is a gradient-boosted tree implementation. To reduce over-fitting and maintain high performance, tree boosting groups a large number of slow learners together. Tree boosting uses loss function and optimization techniques to identify the parameters that minimize the loss function given the data, similar to other statistical learning problems. To solve this problem, a sequence of trees is created, each one fitting the residuals of the previous one before being added to the function. Each subsequent tree attempts to enhance the fitted function in locations where an error has already been introduced.

The first technique (A) includes performing binary classification at the task level and assigning a priority score to each process based on the maximum prediction probability. The alternative technique (B) entails aggregating the features after they have been processed to establish a one-to-one correspondence between characteristics and processes. Following the data preparation, the classification capacity of several modeling methodologies is investigated. A ROC-AUC and recall for the positive outcome class are used to measure classification accuracy. The model is built utilizing data from the client's financial domain from 2018, to apply results to other domains in the future. The authors compare early findings on task-level predictions to perform model selection. The chosen model forecasts the probability of completing tasks using gradient decision tree-boosting with a logistic output.

9) HUJAINAH et al. [65]

Hujainah et al. in [65] propose a new semi-automated RP technique (SRPTackle) and automation implementation tool (SRPTackle-Tool) to address problems with existing RP techniques such as time consumption, scalability, excessive reliance on expert intervention, automation and the lack of a SQP process for evaluating stakeholder impact in prioritizing requirements. The proposed SRPTackle provides a semi-automated process for prioritizing a large number of requirements without requiring manual intervention, as well as reducing the need for expert intervention in assigning priority values to requirements, classifying requirements, running the SQP process and generating a ranked list of requirements. The SRPTackle approach employs the WSM technique for RPV formulation, clustering algorithms (K-means and K-means++), and the BST. In seven trials, the performance of SRPTackle is assessed using medium and large sets of requirements from the RALIC benchmark dataset. In comparison to other current RP approaches, the findings show that SRPTackle can manage a broad range of requirements and generate more accurate results in less time, as well as being more successful in resolving the defined RP limits. Catering to the requirements Independence might be a future trend for increasing SRPTackle performance.

There are several requirements prioritizing approaches and selecting the most appropriate one is a difficult task. This study is proposed for the evaluation of AI-based techniques based on evaluation criteria. To mitigate the negative impacts of traditional prioritization techniques such as accuracy, scalability, cost, optimal solutions, and many others mentioned in the literature, software companies are turning to AI-based techniques for requirements prioritization. Many requirements prioritizing techniques have been offered in this critical field, however, there is a lack of evidence that can be used to determine what key parameters are used to evaluate AI-based techniques. To address this key problem we have proposed this study.

IV. EVALUATION CRITERIA

Requirements prioritization is a process of ranking requirements that has some inputs, processing, and outputs. Since three major types of AI-based approaches have been used to rank the requirements including fuzzy logic, optimization algorithms, and machine learning, so we need evaluation criteria that can asses them separately. This is because all three types possess different characteristics and operating methods. So besides generic criteria to evaluate prioritization techniques we have also defined specific criteria to evaluate fuzzy logic, optimization algorithms, and machine learning techniques.

We identified nine parameters that are common evaluation criteria for all surveyed Fuzzy Logic, Optimization, and Machine Learning based requirements prioritization techniques such as accuracy, scalability, efficiency, affordability, Optimal set of solutions, ease of use, tool support, case study and statistical analysis with different values of 'Yes', 'Partial' and 'No' whereas some parameters are associated with specific AI techniques such as consistency ratio, self-adaptive and positive membership function are three evaluation criteria used only for Fuzzy Logic-based prioritization techniques, optimization algorithm, fitness function, and interactivity are used for evaluation of Optimization-based prioritization techniques whereas classifier, classification model and feature set are used as evaluation criteria of Machine Learning-based prioritization techniques. The parameter of Redundancy problem handling is another evaluation criterion being used for both Machine Learning and Optimization-based prioritization techniques. This parameter is not considered for the evaluation of surveyed Fuzzy Logic techniques.

We designed benchmark parameters and selected their values of Yes, No, and partial based on related research studies [55], the domain of our study, and experience. The values chosen for the criteria have a real relation to the research questions. The criteria of evaluation such as accuracy, scalability, efficiency, and others mentioned in our study are potential strengths of AI-based requirements prioritization techniques and if the values of these criteria are No or partial, it becomes a weakness of that technique which is in other words an open area of research. The range of values of these parameters is defined on real projects. The value of the criteria is 'Yes' if a technique produces results greater or equal to 80%. The value of criteria is 'Partial' if a technique's results range between 50% and 80%. The value of the criteria is 'No' if a technique's results are less than or equal to 50%. This range is set for parameters of accuracy, scalability, and efficiency only because these parameters are quantitatively measured. We design general criteria for the evaluation of AI techniques that may fit most of the projects and also specific criteria associated with each group of AI-based requirements prioritization techniques.

A. GENERIC CRITERIA

In this section, we discuss generic evaluation criteria that have been designed to assess the techniques based on their abilities to rank the requirements and possess all the desired characteristics in them.

1) ACCURACY (ACCU)

A requirements prioritization technique is effective if it generates accurate results that are according to stakeholders' desires. In general, the prioritization results should be error-free. Table 7 presents the evaluation criteria for accuracy.

TABLE 7. Evaluation criteria for accuracy.

Value	Criteria
Yes	This means that this technique generates accurate results if accuracy is greater than or equal to 80 percent.
No	This technique does not generate accurate results if the accuracy is less than or equal to 50 percent.
Partial	This means that this technique is partially accurate if accuracy is between 50 and 80 percent.

2) SCALABILITY (SCALE)

The majority of requirements prioritizing approaches work well when there are a few requirements, but many strategies have limitations when several requirements range from medium to large. A suitable method works well for a wide range of requirements. The scalability evaluation criteria are listed in Table 8.

TABLE 8. Evaluation criteria for scalability.

Value	Criteria
Yes	This means that this technique is scalable if several requirements are greater than or equal to 80 percent.
No	This technique is not scalable if several requirements are less than or equal to 50 percent.
Partial	This means that this technique is partially scalable if the number of requirements is between 50 and 80 percent.

3) EFFICIENCY (EFFI)

An efficient technique should make less number of comparisons that will lead it to find the solution quickly. An efficient technique utilizes all required resources well without wasting or exceeding given limits such as cost, time, and other system resources. Criteria for efficiency are presented in Table 9.

TABLE 9. Evaluation criteria for efficiency.

Value	Criteria
Yes	This means that this technique is efficient.
No	This technique is not efficient.
Partial	This means that this technique is partially efficient.

4) AFFORDABILITY (COST)
An affordable requirements prioritization technique is cost-effective. The value of each requirement is greater than the cost of implementing it. It may make pairwise comparisons to determine the relative cost of implementing each candidate requirement. It may give the project more weight than it can afford to accomplish. Finding an ideal set of requirements early on in the software development process is the most cost-effective way. An efficient technique reduces the total cost of development within the limits of resources (cost). We present evaluation criteria for affordability in Table 10.

TABLE 10. **Evaluation criteria for affordability.**

Value	Criteria
Yes	This value represents the technique's affordability for cost saving.
No	This value means that the technique is not affordable and may not save cost.

5) SET OF SOLUTIONS (OPTIM)
The effective technique produces multiple possible sets of ranked requirements and the best one can be chosen from them. More than one solution allows the requirement engineers to pick the most suited set of ranked requirements. The criteria for this parameter are presented in Table 11.

TABLE 11. **Evaluation criteria for set of solutions.**

Value	Criteria
Yes	This value means that the technique produces multiple possible solutions.
No	This value means that the technique does not produce multiple solutions.

6) EASE OF USE (EOUSE)
Prioritization approaches should be simple to use, understand, and gain the user's trust and attention. For both small and large number of requirements, the techniques should be simple to use. An efficient automated prioritization approach uses less sophisticated calculations to produce the final result and the results are simple to comprehend. This parameter's assessment criteria are listed in Table 12.

TABLE 12. **Evaluation criteria for ease of use.**

Value	Criteria
Yes	This value indicates how simple it is to understand and implement the approach to rank requirements.
No	This value means that the technique is difficult to learn and use.

7) REDUNDANT REQUIREMENTS (REDN)
Repeating groups of requirements are called redundant requirements. The redundancy problem is a major problem that needs to be addressed well at multiple levels as it leads to exceeding the system budget and time. An automated prioritization technique that resolves redundancy problems whereas prioritization of requirements is considered a reliable technique. Requirements prioritization techniques should resolve redundancy problems to increase system reliability. Two objectives are optimized by handling the redundancy problem. The first objective is maximizing the reliability of the system and the second objective is minimizing system cost. Table 13 shows the evaluation criteria for this parameter.

TABLE 13. **Evaluation criteria for redundant requirements.**

Value	Criteria
Yes	This value shows that the approach overcomes the problem of redundancy.
No	This value shows that the approach does not address the problem of redundancy.

8) TOOL SUPPORT (TOOL)
Automated tools can be used to enhance the performance of requirements prioritization. A tool is an excellent way to assess the viability of a technique since it serves as evidence of concept and experimental results can demonstrate the method's success. The assessment criteria for this metric are listed in Table 14 below.

TABLE 14. **Evaluation criteria for tool support.**

Value	Criteria
Yes	This value indicates that the approach completely automates the requirement prioritizing process and offers tool assistance.
No	This value means that the technique does not offer automation for the requirement prioritization process.

9) CASE STUDY (CASE)
A case study is a qualitative research method to analyze the proposed approach. It provides empirical evidence of the effectiveness of a technique used for prioritization. This parameter specifies whether the case study has been conducted in real by researchers or not. Table 15 below demonstrates the evaluation criteria for this parameter.

TABLE 15. **Evaluation criteria for case study.**

Value	Criteria
Yes	This value tells that one or more case studies have been conducted to assess the technique.
No	This value tells that no case study has been used to evaluate the proposed technique.

10) STATISTICAL ANALYSIS (STAT)
This is a tool to analyze the results of experiments statistically. Many authors have conducted comparative analyses on the results of different prioritization techniques using different statistical tools. Table 16 below describes the evaluation criteria for statistical analysis.

TABLE 16. Evaluation criteria for statistical analysis.

Value	Criteria
Yes	This value means that statistical analysis has been conducted to evaluate the prioritization technique.
No	This means that statistical analysis has not been conducted to evaluate the prioritization technique.

B. CRITERIA FOR FUZZY LOGIC BASED TECHNIQUES

In this section, we discuss evaluation criteria for fuzzy logic-based techniques. It has been designed to assess the techniques specifically in the light of the properties and attributes that a good fuzzy logic-based requirements ranking technique should have. A good fuzzy logic method should provide consistency ratio that should be always between 0 & 0.1 [59], membership function should always be positive [59], [64] and self-adaptive property [24], [30], [31], [37], [60].

1) CONSISTENCY RATIO (CONSI)

Consistency arises as the conflicts among requirements reduce. A prioritization approach is selected that prioritizes requirements in such a way that there are no conflicts among the prioritized list of criteria. To remove inconsistency, an approach computes an exact consistency ratio that is always between 0 and 0.1 and offers a normalized most favorable unique priority vector for each fuzzy judgment. Table 17 shows the assessment criteria for this parameter.

TABLE 17. Evaluation criteria for consistency ratio.

Value	Criteria
Yes	This value means that the technique computes an accurate consistency ratio.
No	The value No means that the technique does not compute an accurate consistency ratio.

2) SELF ADAPTIVE (ADAPT)

Self-adaptive features of fuzzy logic systems include self-optimizing, self-healing, self-configuring, and self-protection. It may classify requirements as crucial, essential, or peripheral automatically. Prioritized requirements can be classified as non-negotiable and negotiable requirements or sufficient and appropriate requirements. A technique will be considered good if it possesses self-adaptive characteristics. The evaluation criteria for this parameter are presented in Table 18.

TABLE 18. Evaluation criteria for self adaptive.

Value	Criteria
Yes	This value indicates that the approach is self-adaptive.
No	The value indicates that the technique does not have self-adaptive properties.

3) MEMBERSHIP FUNCTION (FUNC)

In terms of input/output variable selection, membership functions, and methods for determining completeness and understandability of requirements, a fuzzy system is built. An effective fuzzy logic system always provides a positive degree of membership function. Its value never gets negative. Table 19 presents the evaluation criteria for this parameter.

TABLE 19. Evaluation criteria for membership function.

Value	Criteria
Yes	This value means that the technique provides a membership function.
No	This value means that the technique does not provide a membership function.

C. CRITERIA FOR OPTIMIZATION ALGORITHMS BASED TECHNIQUES

In this section, we discuss evaluation criteria for the techniques that use one of the optimization algorithms. It has been designed to assess the techniques specifically in light of the properties and attributes that a good optimization algorithm-based requirements prioritization technique should have.

1) OPTIMIZATION ALGORITHM (ALGO)

The researchers use a variety of optimization algorithms to prioritize software requirements. There exist some variations of genetic algorithm and besides those ant colony optimization has also been applied for the same. The type of algorithm used in an approach gives an insight into the process of ranking and optimizing a set of solutions. Table 20 presents the evaluation criteria for this parameter.

TABLE 20. Evaluation criteria for optimization algorithm.

Value	Criteria
XGA	The value XGA means that authors have applied one of the variations of Genetic Algorithm (GA) for requirements prioritization.
ANT	This value indicates that the authors employed the Ant Colony Optimization technique to prioritize the requirements in the proposed approach.

2) FITNESS FUNCTION (FITF)

The fitness function is the core part of an optimization algorithm. We use the disagreement measure as an indicator of fitness to identify the finest individuals. The fitness function should be computed precisely by the prioritization technique. When the level of disagreement goes down, the fitness of a solution improves. A good fitness function assures improved optimization and quick convergence to the optimal solution. Table 21 presents the evaluation criteria for this parameter.

TABLE 21. Evaluation criteria for fitness function.

Value	Criteria
Yes	The value indicates that the technique proposes and presents the fitness function.
No	The value indicates that the technique does not include a fitness function.

3) INTERACTIVITY (INTER)

The main objective of using an optimization algorithm is to automate the process but some researchers have proposed an interactive optimization algorithm that requires the user's input to proceed in certain situations. This defeats the purpose of automation. A good prioritization technique should be fully automated to minimize human effort and reduce the chances of human errors. Table 22 presents the evaluation criteria for this parameter.

TABLE 22. Evaluation criteria for interactivity.

Value	Criteria
Yes	This indicates that the technique requires user interaction to rank the requirements.
No	This indicates that the technique requires no user interaction to rank the requirements.

D. CRITERIA FOR MACHINE LEARNING BASED TECHNIQUES

In this section, we discuss evaluation criteria for machine learning-based techniques. It has been designed to assess the techniques specifically in light of the properties and attributes that a good machine learning-based requirements prioritization technique should have.

1) CLASSIFIER (CLASS)

In the domain of requirements prioritization, it is important to see how the authors perceive the problem of ranking the requirements. Classifier classifies either problem belongs to binary class, multiple class, or regression problem. This will show how comprehensive is the coverage of the problem domain. We present the evaluation criteria for this parameter in Table 23.

TABLE 23. Evaluation criteria for classifier.

Value	Criteria
<Type>	This value means that authors have specified the classifier type they have used in the research including BIN for the binary classifier, MUL for the multi-class classifier, and REG for the regression-based classifier.
No	This means that authors have not specified the classifier they have used for classification.

2) CLASSIFICATION MODEL (MODEL)

A classification model is a model that is built from an input data set provided to the classification algorithm. The term model usually implies some mechanistic insight or logic into how the input variables are related to each other. An accurate classification model facilitates a classifier for accurate class detection of input variables. Table 24 presents the evaluation criteria for this parameter.

TABLE 24. Evaluation criteria for classification model.

Value	Criteria
<Name>	This signifies that the authors have described the classification model that will be used to classify the requirements. The number XLM indicates that the authors used different Loop Models, such as the Iterated Single-User Loop Model (ISULM), Basic Single-User Loop Model (BSULM), Basic Multi-User Loop Model (BMULM), and Self Multi-User Loop Model (SMULM). The value MNM denotes the adoption of the Probabilistic Network Model by the authors.
No	This means that authors have not specified the classification model used for classification in the proposed work.

3) FEATURE SET (FSET)

A feature is an individual measurable property or characteristic of a phenomenon being observed. An effective algorithm chooses an informative, discriminating, and independent feature set. Feature selection is crucial when software prioritization is performed through machine learning. It should be done carefully as the feature set helps classify the requirements as per their importance and constraints. Table 25 presents the evaluation criteria for this parameter.

TABLE 25. Evaluation criteria for feature set.

Value	Criteria
Yes	The value indicates that the authors have specified the features set that they have used for ranking the requirements with machine learning.
No	The value indicates that the authors have not specified the features set that they have used for ranking the requirements with machine learning.

V. RESULTS ANALYSIS AND COMPARISON

This section presents the demography of selected papers and examines Fuzzy Logic, Optimization, and Machine Learning Techniques in depth to determine their strengths and weaknesses (RQ1). Moreover, this section presents the advantages of Machine Learning techniques over AI-based techniques (RQ2) and the limitations of applying AI-based techniques for software requirements prioritization (RQ3).

A. DEMOGRAPHICS OF OUR STUDY

Table 26 details the overview of 28 research studies per year, including a breakdown of publication type of Journal and Table 27 gives an overview of 16 conference papers and 2 books. The publication date of primary studies ranges from 2000 to 2021. We extracted total of 46 papers out of which 15 papers on fuzzy logic-based requirements prioritization techniques were published 4 in Springer, 1 in Elsevier, 10 in Google Scholar, 22 studies on optimization-based requirements prioritization techniques published 8 in IEEE, 6 in Springer, 2 in Elsevier, 2 in ACM, 4 in Google Scholar and 9 studies on machine learning based requirements prioritization techniques amongst 2 published in IEEE, 2 in Springer, 3 in Elsevier and 2 in Google Scholar.

Fig. 4. shows the distribution by venue type of selected studies after applying inclusion, exclusion criteria, and quality assessment: 61% of papers are published in journals, 35% in conferences, and 4% as book chapters.

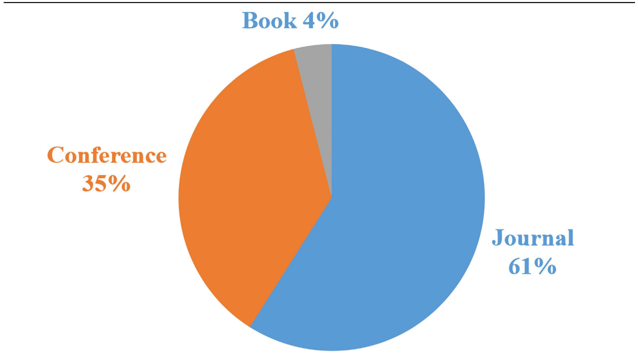


FIGURE 4. The distribution of research papers per venue type in the period from 2000 to 2021.

We also noticed that the trend of using AI techniques in the area of requirements prioritization increased in the year 2015 and onward as shown in Fig. 5.

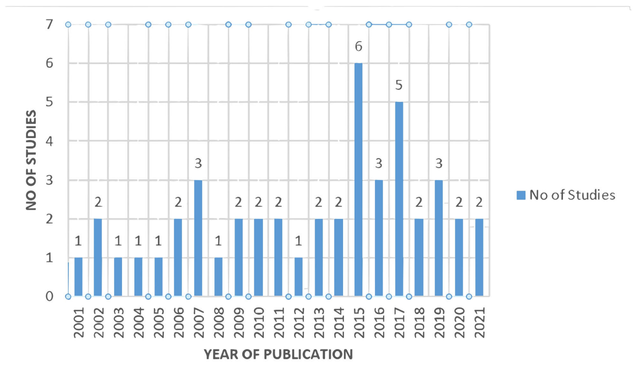


FIGURE 5. Our selected studies concerning year of publication.

B. STRENGTHS AND WEAKNESSES OF THE SURVEYED TECHNIQUES (RQ1)

We have identified key strengths as well as weaknesses of three surveyed techniques that are discussed below in detail:

1) STRENGTHS AND WEAKNESSES OF FUZZY LOGIC-BASED TECHNIQUES

Our survey results depicted in Table 28 show that Fuzzy logic techniques need to generate more accurate prioritization results. Because data is frequently inaccurate, Fuzzy inference must compute data to find correct values. By decoupling dependable variables, dependencies between variables in the system can be handled via fuzzy inference. As a result, fuzzy logic improves the ordering process by making it more correct and reliable. A good Fuzzy logic System sets a threshold value for the level of disagreement and keeps the disagreement level below this threshold value. The level of disagreement should be between 0-1. It should not be negative for effective and desired results. Results show that out of 15 surveyed techniques, only two techniques [24], [35] show 90 % and 93.89% accuracy respectively due to multi-level prioritization, 9 techniques [30], [31], [44], [49], [51], [61], [36], [59], [64] are partially accurate whereas other studies do not mention neither provide any computation of level of disagreement. So this is evidence that the accuracy of fuzzy logic systems needs to be computed and improved.

The framework of fuzzy logic is scalable for a large number of requirements [35]. Fuzzy logic techniques can be simply extended to cover various types of requirements, other than proposed techniques by Dabbagh and Lee [39], Mishra et al. [44], Sadia and Faisal [60], Ahmad et al., [51], [64] where researchers do not mention either their proposed technique is scalable or not. Achimugu et al. [36] propose a multi-criteria decision-making technique that is implemented on twenty requirements only.

In terms of both time utilization and capability to learn, the Fuzzy Logic technique is more efficient, compared to related approaches other than techniques proposed in [30], [44], [51], [59], [60], [61], [64], and [36] where researchers have not provided any detail about this parameter. We analyze that a certain level of automation leads to a more efficient technique in the end. Time is an imperative component in any project and it must be considered and managed well. In general, a time restriction assists in identifying which requirements can be implemented in a shorter period. In comparison to traditional methodologies, our survey findings suggest that fuzzy systems require less time to compute.

Results shown in Table 28 say that Fuzzy Logic system offers cost-effective solutions except proposed methods in [35], [51], [60], [61], [64], and [36] considering cost factor while prioritizing requirements keeps project within budgetary plan. Although researchers claim that a fuzzy logic system is cost cost-effective solution cost is not calculated explicitly in these surveyed papers. Similarly, we find no evidence that proposed fuzzy logic technique generates optimal set of solutions in [25], [30], [37], [39], [44], [49], [51], [60], [61], and [64] so this area needs research attention.

The researcher does not evaluate their proposed techniques considering ease of use parameter in Ramzan et al. [24], Momeni et al. [31], Mishra et al. [44], Ahmad et al. [51], Singh et al. [59], Sadia, Faisal [60], Bisht and Kushwaha [64] and [35]. Hence no detail is given about the ease of use parameter. The prioritization and selection method's complexity proposed by Mishra et al., is O(n) which is less complex than conventional methods specifically AHP which requires O(n2) operations so in practically every situation, fuzzy logic requirements prioritization produces better and more remarkable results.

TABLE 26. Overview of selected studies (publication type journal).

Paper ID	Reference	Technique	Category	Year	Type	Publisher
P2	Bagnall et al [2]	Exact, Greedy and GRASP Techniques	Optimization	2001	Journal	Elsevier
P4	Deb at al [4]	NSGA11	Multi-objective Genetic Algorithm	2002	Journal	Science Direct
P7	Greer et al. [7]	EVOLVE Method	Optimization	2004	Journal	Elsevier
P17	Finkelstein et al [17]	Multi-objective Optimization Technique	Optimization	2008	Journal	ACM
P19	Duan, et al. [19]	Machine Learning Technique	Machine Learning Technique	2009	Journal	www.dsdm.org
P24	Ramzan et al [24]	Expert Driven Fuzzy Logic Based Prioritization Technique	Fuzzy Logic	2011	Journal	Springer
P26	Perini et al [26]	Case-Based Ranking (CB Rank)	Machine Learning	2013	Journal	International Journal of Computer Science and Communication Networks
P27	Tonella et al [27]	IGA	Genetic Algorithm	2013	Journal	IEEE
P30	Sadiq And Jain [30]	Requirements prioritization in goal-oriented requirements elicitation process	Fuzzy Logic	2014	Journal	IARIA
P31	Momeni et al [31]	Neuro-Fuzzy based Approach	Fuzzy Logic	2014	Journal	Springer
P35	Babar et al [35]	An expert system for a scalable software RP process	Fuzzy Logic	2015	Journal	Knowledge-Based Systems
P36	Achimugu et al [36]	fuzzy multi-criteria decision-making (FMCDM) approach	Fuzzy Logic	2015	Journal	Journal Technology
P37	Jawale and Bhole [37]	Adaptive Fuzzy Hierarchical cumulative Voting	Fuzzy Logic	2015	journal	Elsevier
P39	Dabbagh and Lee [39]	An Approach for Prioritizing NFRs According to Their Relationship with FRs	Fuzzy Logic	2015	Journal	Int. J. Res. Eng
P40	Gongalez and Toledano [40]	Differential Evolution algorithm	Optimization	2015	Journal	International Journal of Modern Education and Computer Science
P42	Kumari and Srinivas [42]	quantum-inspired evolutionary algorithms	Evolutionary algorithms	2017	Journal	Information and Software Technology Elsevier
P43	Valsala and Nair [43]	Hybrid Enriched Genetic Revamped Integer Linear Programming Model (EGRILP)	Genetic Algorithm	2016	Journal	IEEE
P44	Mishra et al [44]	Approach to Prioritize the Requirements Using Fuzzy Logic	Fuzzy Logic	2016	Journal	Research Gate
P48	Rao et al [48]	JAYA Algorithm	Multi- objective optimization Algorithm	2017	Journal	Engineering Applications of Artificial Intelligence
P49	Gulzar et al [49]	Fuzzy Approach to Prioritize Usability Requirements Conflicts	Fuzzy Logic	2017	Journal	MATEC Web of Conferences
P50	Marghny et al [50]	Non-dominated sorting genetic algorithm with Pareto tournament for a multi-objective optimization approach (NSGA-IIPT)	Optimization	2016	Journal	IEEE
P52	Shao et al [52]	DRank	Semi automated Machine Learning	2017	Journal	elsevier
P54	Gupta et al [54]	CDBR	Machine Learning	2018	Journal	elsevier
P60	Sadia and Faisal [60]	Volatile Requirement Prioritization	Fuzzy Logic	2019	Journal	Springer
P61	Mougouei et al [61]	Partial selection of security requirements in software projects	Fuzzy Logic	2019	Journal	Research Gate
P63	Alrezaamiri et al [63]	Parallel multi- objective artificial bee colony algorithm	multi-objective Optimization	2020	Journal	Springer
P64	Bisht and Kushwaha [64]	Value intelligent RP technique	Fuzzy Logic	2020	Journal	IJRDASE
P65	Hujainah et al [65]	SRPTackle	Machine Learning	2021	Journal	Information and Software Technology Elsevier

TABLE 27. Overview of selected studies (publication type: conference paper and book).

Paper ID	Reference	Technique	Category	Year	Type	Publisher
P3	Feather and Menzies [3]	Simulated annealing	Optimization	2002	Conference Paper	IEEE
P5	Avesani et al. [5]	Case-Based Ranking for Decision Support Systems	Machine Learning	2003	Conference Paper	Springer
P8	Avesani et al. [8]	Scalability problem with Machine Learning	Machine Learning	2005	Conference Paper	IEEE
P9	Harman et al. [9]	ACO	Optimization	2006	Conference Paper	ACM
P10	Baker et al. [10]	Greedy and simulated annealing algorithms	Optimization	2006	Conference Paper	IEEE
P11	Saliu and Ruhe [11]	Bi objective release planning	Optimization	2007	Conference Paper	IEEE
P15	Quiroz et al [15]	Interactive genetic algorithm	Genetic Algorithm	2007	Conference Paper	Springer
P16	Zhang et al. [16]	Specific Multi-Objective Next Release Problem (MONRP)	Optimization	2007	Conference Paper	IEEE
P18	J. J. Durillo and Zhang [18]	NSGA 11 for multi-objective next release problem	Optimization	2009	Conference Paper	Springer
P20	Tonella et al. [20]	IGA	Genetic Algorithm	2010	Conference Paper	Springer
P21	Sagrado et al. [21]	Meta heuristic Technique	Optimization	2010	Conference Paper	IEEE
P23	De Souza et al. [23]	Ant Colony Optimization (ACO)	Optimization	2011	Conference Paper	Springer
P25	Ejnioui et al. [25]	Fuzzy multi-attribute decision making	Fuzzy Logic	2012	Conference Paper	International Journal Of Innovative Computing, Information And Control
P34	Achimugu and Selamat [34]	K-means Algorithm	Machine Learning	2015	Book	Springer
P51	Ahmad et al. [51]	A fuzzy based MoSCoW Method	Fuzzy Logic	2017	Conference Paper	Research Gate
P55	Ahuja et al. [55]	Least-Squares-Based Random Genetic Algorithm	Genetic Algo	2018	Book	Springer
P58	Bollumpally et al. [58]	A Machine Learning Approach to Workflow Prioritization	Machine Learning	2021	Conference Paper	IEEE
P59	Singh et al. [59]	Hybrid approach for RP using logarithmic fuzzy preference programming (LFPP) and artificial neural network (ANN)	Fuzzy Logic	2019	Conference Paper	Springer

A good fuzzy logic method should provide a consistency ratio that should be always between 0 & 0.1. Results show that out of all techniques being surveyed, only two techniques compute consistency ratio in [59] and [35] whereas other research studies do not give any detail regarding consistency ratio neither researchers compute consistency ratio in these studies.

The adaptation mechanism of Fuzzy logic targets uncertain and ambiguous situations where composite service behavior can be altered by client expectations and preferences. A good Fuzzy logic system should have self-adaptation properties. To broaden the scope of events that may happen during run time, the suggested Adaptive Fuzzy Hierarchical Cumulative Voting [37] analyzes different self-adaptive features such as self-protection, self-healing, self-optimizing, and self-configuring. Prioritizing requirements at run time may be beneficial. Fuzzy Logic-based techniques proposed in [24], [30], [31], [37], and [60] offer self-adaptive properties that make the prioritization process more efficient and effective whereas techniques proposed in [25], [35], [39], [44], [49], [51], [59], [61], [64], and [36] do not provide any evidence of self-adaptive properties.

The effectiveness of a Fuzzy logic system may depend on how well the membership function of input/output is defined but that needs a level of skill that is not always present in software projects so better provision of expertise to well well-defined membership function may make the fuzzy logic System more effective. The degree of membership function should never be negative. To assist decision-makers, the fuzzy system is a more effective tool in requirements analysis if the degree of membership function is always positive. Our survey results demonstrate that in eleven studies, researchers have provided details of positive membership function parameters whereas they have not provided any detail regarding membership function in [37], [39], [51] [35], and [36].

Results show that four research studies [31], [44], [59], [60] have tested proposed prioritization techniques practicality using MATLAB Fuzzy logic toolbox for automating software requirement prioritization whereas an intelligent tool is used for technique proposed in [24] and fuzzy inference is implemented using fuzzy control language in [61]. For the Fuzzy logic technique proposed by Dabbagh and Lee [39], the tool is created in Microsoft Visual Studio 2008 and.NET Framework 3.5 utilizing the C# programming language. Additionally, Microsoft SQL Server 2008 R2 is used to back up the data created by TIPA.

Authors have not provided any tool support information or online links for techniques proposed in [25], [30], [36], [37], [49], [51], [64], and [35], The main reason researchers use MATLAB fuzzy logic toolbox for implementation purposes to design and edit fuzzy inference systems, or one can use Simulink to incorporate Fuzzy algorithms may be included into simulations or standalone C programs that rely on MATLAB-created fuzzy systems. Even though one may operate entirely from the command line, this toolkit heavily depends on graphical user interface (GUI) elements to help us complete our tasks.

Researchers use Fuzzy c means (FCM) in [24] for comparative analysis of requirements but they do not mention specifically which tool is used for requirements prioritization technique implementation. The approach proposed in [25] is rooted in credibility theory but again no tool support detail has been mentioned. The adaptation module monitors and evaluates the outcomes of prioritized requirements [37], determining whether they are accurate or not based on the analysis. Although the authors mention that they use an adaptation module they do not specifically mention the name of the supporting tool or any other additional details. Similarly, in [61] researchers implement the rules of fuzzy inference utilizing fuzzy control language (FCL) but they do not mention any additional details of the tool.

Results show that 11 out of 15 surveyed fuzzy logic-based prioritization techniques [24], [30], [31], [35], [44], [49], [51], [59], [60], [61], and [36] mentioned in table 28 are implemented using real-world cases whereas four techniques mentioned in [25], [37], [39], and [64] are not being tested in real case. Similarly, a theoretical framework is presented in [25] but this study lacks real case evidence. Moreover, detail related to software tool is also not provided. Real-world experiments are needed for further verification of results. As part of future work, Dabbagh and Lee [39] intend to research to evaluate the presented technique in a real-world industrial context to determine its efficacy and limitations in practical solutions.

2) STRENGTHS AND WEAKNESSES OF OPTIMIZATION BASED TECHNIQUES (RQ1)

The results of the literature survey illustrated in Table 29 prove that Optimization techniques generate accurate results. All surveyed techniques provide a minimum degree of disagreement which leads to the best desired and accurate solution. For the best-fit solution, disagreement should be at a minimum level. However, the final disagreement generated by IGA in [20] and [27] is minimal but it is not zero. Similarly, the Least-Squares-Based Random Genetic Algorithm [55] offers the least amount of conflict between the overall prioritized requirements and the different limitations that are either encoded with the requirements or encoded repeatedly by the user throughout the prioritization procedure.

Although optimization techniques are accurate, they are often less scalable, as demonstrated by sources such as [2], [3], [4], [7], [9], [15], [21], [43], and [64]. Most optimization techniques are designed for small requirements, which raises questions about their effectiveness as the number of requirements increases. J.J Durilio and Zhang have claimed in [18] that NSGA-11 outperforms Mocell for large requirements, but further investigation is necessary to determine how these approaches scale for even larger sets of requirements and/or customers. Unlike iterative GA techniques, the NSGA-11 methodology can expand the distance between pairings as the number increases, resulting in better outcomes. Increasing the number of requirements can lead to more accurate results overall. However, IGA is only partially automated and requires human input for fitness function, making it more prone to errors and less accurate solutions.

Many researchers have not explicitly discussed the efficiency parameter in optimization techniques such as those mentioned in references [2], [3], [7], [11], [15], [18], [20], [21], [23], and [40]. However, time is a crucial evaluation criterion when it comes to prioritization techniques. Optimization techniques can help reduce the consumption of time by minimizing the number of comparisons or random selection of population, unlike conventional techniques. For instance, the Least-Squares-Based Random Genetic Algorithm [55] can reduce processing time by selecting population randomly, while keeping it less than execution time.

According to Table 29, optimization-based requirements prioritization methods are cost-effective. All surveyed techniques' authors claim that optimization algorithms provide affordable solutions in terms of cost except for those in references [15] and [21]. Non-dominated sorting genetic algorithm with Pareto tournament (NSGA-IIPT) [50] is an updated genetic algorithm that can generate high-quality solutions within a set development budget by combining customer goals and budget constraints. Other techniques like IGA and EVOLVE do not explicitly calculate cost.

A key parameter for evaluating optimization techniques is the multiple optimal set of quality solutions. All optimization techniques provide optimal sets of solutions with minimum disagreement and high fitness functions, except for those presented in references [3], [20], [21], and [27]. Sagrado et al. in [21] did not address the quality of the solutions parameter in their proposed study, but they plan to look into it in the future. However, researchers in this study faced difficulties with classic operators during the adaptation of GA, so they introduced a repair operator to find a valid solution in a greedy manner. Feather and Menzies [3] propose an optimization technique that offers human-based selection, which takes more time and may be unable to produce a solution closer to an optimal solution.

TABLE 28. **Analysis of fuzzy logic-based requirements prioritization techniques.**

Techniques	Evaluation Parameters											
	ACCU	SCALE	EFFIC	AFFORD	OPTIM	EOUSE	TOOL	CASE	STAT	CONSI	ADAPT	FUNC
Ramzan et al. [24]	Yes	Yes	Yes	Yes	Yes	No	No	Yes	Yes	No	Yes	Yes
Ejnioui et al. [25]	No	Yes	Yes	Yes	No	Yes	Yes	No	No	No	No	Yes
Momeni et al. [31]	Partial	Yes	Yes	Yes	Partial	No	Yes	Yes	Yes	No	Yes	Yes
Sadiq and Jain [30]	Partial	Yes	No	Yes	No	Yes	No	Yes	Yes	No	Yes	Yes
Jawale and Bhole [37]	No	Yes	Yes	Yes	No	Yes	No	No	No	No	Yes	No
Dabbagh and Lee [39]	No	No	Yes	Yes	No	Yes	Yes	No	No	No	No	No
Babar et al. [35]	Yes	Yes	Yes	No	Yes	No	No	Yes	Yes	Yes	No	Yes
Achimugu et al. [36]	Partial	No	No	No	Yes	Yes	No	Yes	Yes	No	No	Yes
Mishra et al. [44]	Partial	No	Yes	Yes	No	No	Yes	Yes	No	No	No	Yes
Ahmad et al. [51]	Partial	No	No	No	No	No	No	Yes	No	No	No	No
Gulzar et al. [49]	Partial	Yes	Yes	Yes	No	Yes	No	Yes	Yes	No	No	Yes
Singh et al. [59]	Partial	Yes	No	Yes	Partial	No	Yes	Yes	Yes	Yes	No	Yes
Mougouei et al. [61]	Partial	Yes	No	No	No	Yes	Yes	Yes	No	No	No	Yes
Sadia and Faisal [60]	No	No	No	No	No	No	Yes	Yes	No	No	Yes	Yes
Bisht and Kushwaha [64]	Partial	No	No	No	No	No	No	No	No	No	No	Yes

In several research studies such as [9], [10], [16], [27], [50], [18], [20], [21], [43], and [64], the parameter for easy usage is not clearly explained. Additionally, some optimization algorithms produce a set of unnecessary solutions, as noted in [2], [3], [9], [10], [11], [15], [17], [20], [21], [27], and [43]. It is concerning that many of these optimization techniques do not address the redundancy problem or perform any computations to resolve it. This research gap indicates that further attention is needed.

Several optimization algorithms, including Order GA, Weighted + Pareto GA + NSGA11 Algorithm, IGA, ACO, Hybrid Enriched Genetic Revamped Integer Linear Programming Model, Non-dominated sorting genetic algorithm with Pareto tournament for a multi-objective optimization approach (NSGA-IIPT), Least-Squares-Based Random Genetic Algorithm, pseudo-polynomial algorithm using dynamic programming, MOABC, Multi-objective JAYA (MO-JAYA), Quantum-inspired Elitist Multi-objective Evolutionary Algorithm and Differential Evolution with Pareto Tournaments (DEPT), are used for requirements prioritization. Bagnall et al. use Genetic branch and bound algorithm, Greedy algorithm, and GRASP to achieve near-optimal solutions in less time. Quiroz et al. use IGA and evolutionary methodologies for UI design evolution, with an interpolation technique to minimize selections. Computational efficiency is enhanced by eliminating low fitness requirements.

Various optimization techniques utilize different fitness functions to achieve their objectives. For instance, some studies, such as the work of [7], employ Benefit and Penalty functions, while others, such as [16], use Maximize Score and Minimize Cost functions. Objective and Restriction functions are utilized in [23], whereas Benefit Maximization and Cost Minimization are applied in [2], [17], and [18]. In [10], the Cost Objective function is utilized, while Development Cost and Overall Client Satisfaction are two Objective functions used in [11], [40], and [42]. On the other hand, the fitness function in [20] and [27] is computed by finding the minimum disagreement level.

In the context of Interactive Genetic Algorithms (IGA), the fitness function uses a straightforward disagreement calculation to evaluate the quality of prioritization. This calculation is slightly influenced by erroneous values to help maintain the degree of disagreement below a specific threshold. Once the degree of disagreement is sufficiently low or the elicitation budget has been exhausted, the prioritization process comes to a conclusion. However, the current fitness function utilized in IGA may not lead to the best prioritization, as user inputs, in the form of pairwise comparisons to score individuals, are error-prone.

The NSGA-IIPT algorithm, as presented in [50], employs the Pareto tournament approach and a crowding distance measure as a fitness function to ensure diversity among solutions. In contrast, the Hybrid Enriched Genetic Revamped Integer Linear Programming Model, discussed in [43], calculates the fitness function by subtracting the penalty from one (fitness = 1-penalty). To normalize both objective function values between 0 and 1, researchers suggest that this normalization function is more resilient. Quiroz et al. [15] determine the objective fitness component by evaluating how well UI individuals in the population follow defined style constraints. Gongalez and Toledano [40] must normalize the values of fitness functions before measuring hyper-volume (HV) because HV is not separate from arbitrary objective scaling, and the value of these measurements may be impacted if the range of each objective function varies.

In ten studies, interactive genetic algorithms have been implemented to take user feedback during the prioritization process whenever required. Quiroz et al. [15] employ an interactive tool that prompts the user to make a decision once every generation. Tonella et al. [20], [27] employ an interactive genetic algorithm. However, partial automation and user involvement can lead to increased time consumption and erroneous results.

The surveyed optimization-based requirements prioritization methods utilized real case studies, with the exception of [9]. In addressing the Multi-Objective Next Release Problem (MONRP) [16], four algorithms were employed to simultaneously maximize both value and cost. The study revealed that weighted and Pareto optimum genetic algorithms, as well as the NSGA-II algorithm, demonstrated evidence to support the notion that NSGA-II is well-suited for solving the MONRP. The IGA algorithm [20], [27] was also employed in real software systems. In a real-world case, the proposed method was compared against the state-of-the-art interactive prioritization approach known as Incomplete Analytic Hierarchy Process (IAHP). The authors of this study claim that IGA outperforms IAHP. However, the primary concern with IGA is redundancy problem handling. Furthermore, another optimization algorithm called ACO [23] was implemented via a real case study, which produced more valid and accurate results by utilizing binary tournament as a selection method. There were no notable variations in ACO's results before and after the time constraints were applied. Finally, the non-dominated sorting genetic algorithm with Pareto tournament for a multi-objective optimization approach was utilized to test the efficacy of the proposed approach (NSGA-IIPT) [50]. This was implemented utilizing various examples selected from two real-world datasets.

Feather and Menzies conducted a study, referenced in [3], which used TAR2 to create 100,000 model variants. These variants were analyzed using a computer program to identify the values that caused the most significant deviation from expert predictions. In another study, Quiroz et al. [15]) collected data from three users, presenting nine individuals for user review using a probabilistic tournament selection approach. It is important to conduct further user studies to evaluate the tool's practicality and user interface evolution. Gongalez and Toledano [40]) used two real datasets to assess the effectiveness of their proposed technique, applying four development effort constraints to each dataset. This led to the evaluation of their proposal using four distinct datasets.

In a total of nine surveyed studies, explicit information was provided regarding the tools employed for implementing optimization techniques. For instance, the EVOLVE method [7] utilized Palisade's Risk optimizer tool, which offers a range of methods for modifying variables. However, in one study [50], researchers claimed to have used a software tool to implement the Non-dominated sorting genetic algorithm with Pareto tournament for a multi-objective optimization approach (NSGA-IIPT), without specifying the name of the tool or providing a web link for additional information. They only mentioned system requirements for experimentation. Similarly, Gongalez and Toledano [40] only provided hardware and software requirements, without specifying the software tool employed. On the hardware side, they used an Intel Xeon 2.33 GHz CPU with 4 GB RAM and the gcc 4.4.5 compiler on a Linux kernel 3.13 64-bit OS. In contrast, Alrezaamiri et al. [48], [63], and [42] evaluated the effectiveness of their proposed techniques using the Matlab R2014b tool.

Feather and Menzies [3] used the Tar2 summarizing tool to analyze their data set and identify larger treatment sets. Deb et al. [4] implemented NSGA-11 with an optimizer tool, while Saliu and Ruhe [11] utilized ILOG-CPLEX Optimizer for their bi-objective evolving algorithm. Baker et al. [10] employed greedy and simulated annealing techniques in a C++ environment, while Quiroz et al. [15] utilized evolutionary methods for UI design evolution in XML User-interface Language (XML UIL). XUL was chosen due to its adaptability and ease of widget modification. The syntax and structure of XUL also allow for the creation of a variety of programs, from simple two-button interfaces to full-fledged applications with menu bars, toolbars, and other conventional widget capabilities. The resulting programs appear to be created by the user, with no hint of AI-powered assistance.

After analyzing and comparing various optimization methods, the techniques of Harman et al. were excluded from the evaluation, as per [9] and [11]. Statistical analysis of the results indicated that optimization-based prioritization methods outperformed existing techniques. Specifically, NSGA-II, a Pareto GA, and a Single-Objective GA were found to perform better than Random Search. Additionally, a comparison of IGA, GA, and RAN using the ANOVA test revealed that both GA and IGA outperformed RAN significantly. Furthermore, the Wilcoxon Rank Sum Test showed that ACO performed significantly better than GA. Nonetheless, ACO's major limitation is that it only accounts for synthetic data and may overlook certain aspects of real-world events.

In their study, Finkelstein et al. [17] conducted a performance comparison of 'intelligent' search-based optimization algorithms against Random Search. They found that the former outperformed the latter significantly. Their evaluation included comparisons of the Hybrid EGRILP model [43] with other prioritizing and scheduling models. Results showed that EGRILP produced the most optimal solution. In addition, the NSGA-IIPT algorithm outperformed other algorithms in an independent samples t-test. The Least-Squares-Based Random Genetic Algorithm was also evaluated against other methods, and VOP was found to be more efficient in producing error-free results, consistency, and ease of usage. However, the statistical analysis tool used in their performance comparison tests was not specified.

Gongalez and Toledano [40] utilized the Shapiro-Wilk test to determine that the arithmetic mean was an acceptable statistical measurement for their data. They conducted a comparative analysis using three quality factors to evaluate their study against other comparable papers that have been published. The first factor was the number of non-dominant solutions discovered, referred to as NDS. The favor was given to Pareto fronts with the most non-dominated solutions. The second factor evaluated was the hyper-volume (HV) indicator. This measure was used to calculate the volume in the target space covered by members of a non-dominated set of solutions Q. Algorithms with greater HV values were deemed favorable. The third quality indicator, D-Spread, evaluated the degree of dispersion achieved by collecting obtained solutions. This indicator calculated the Euclidean distances between successive solutions in the Pareto front to evaluate solution diversity. Pareto fronts with a lower spread value were deemed better.

In summary, optimization-based requirements prioritization techniques have been shown to produce accurate, cost-effective, and optimal solutions. However, these techniques face challenges in terms of scalability, efficiency, and redundancy that require further attention. Parameters used for the evaluation of optimization-based requirements prioritization techniques are given in Table 29.

3) STRENGTHS AND WEAKNESSES OF MACHINE LEARNING-BASED TECHNIQUES (RQ1)

Following a thorough analysis of machine learning algorithm-based techniques, we have determined that the techniques presented in Table 30 produce partially accurate requirement ordering. While machine learning techniques are less manual and therefore less prone to human error, there is still room for improvement in the accuracy of the results. Some techniques, such as those presented in [8], are partially automated, which increases the probability of error. Additionally, the user's cognitive burden required to offer a ranked preference relation, as described in [5], has no impact on the approximation inaccuracy.

Researchers in [52] and [54] have claimed that less dependency between requirements improves accuracy. Gupta et al. in [54] use minimum disagreement between stakeholders' and developers' priority as a measure of accuracy.

All of the surveyed machine learning techniques proposed in [5], [8], [19], [26], [34], [52], [54], [65], and [58] offer scalable solutions, as they can prioritize a large number of software requirements and can be used in a variety of scenarios. Table 30 shows that all techniques can be easily extended and overcome redundancy problems by removing repeating groups of requirements, which leads to efficiency in terms of cost and time.

The results indicate that the case-based ranking technique proposed in [5] and [34] is efficient in terms of time, but not cost-effective. On the other hand, the technique proposed in [26] is cost-effective, but less efficient. There is no evidence of efficiency and cost-benefit in [8]. However, the techniques proposed in [19], [52], [54], [65], and [58] are both efficient and cost-effective. Analysts' judgment effort connected with pairwise comparison is kept within reach in these techniques.

According to surveys, most Machine Learning prioritization techniques provide solutions that are less redundant than those presented in [65] and [58]. Additionally, only the techniques discussed in [52], [54], and [65] are evaluated for ease of use. All of the techniques were tested with real, synthetic, and artificial datasets as well as sample software project data using various classification models, classifier feature sets, and software tools.

In the field of data analysis, a tool called a classification tool is utilized to identify and organize various non-functional requirements such as security, performance, and usability. This is discussed in [19]. Meanwhile, the nearest neighbor classifier is employed for case-based ranking in decision support systems as stated in [5], and the K-means classifier is utilized in [65] and [34]. Binary learning weak classifier is utilized in [26] and [58] to partially order requirements, while binary classifier is used to classify requirements in [8]. Finally, Multiple base learners are utilized in the rank boost algorithm in [52], while the swarm optimization algorithm is used in CDBR discussed in [54] to generate final priority.

Various classification models are utilized in scientific research, including the self-multi-user loop model, basic single-user loop model, basic multi-user loop model, and iterated single-user loop model for iterative process and binary classifier. These models are discussed in sources like [5], [8], [19], [26], [34], [52], [54], [58], and [65].

During the prioritization process, it is crucial to carefully identify the feature set. Researchers evaluate their techniques based on feature sets such as accuracy and elicitation effort in [8], security, performance, or usability in [19], number of requirements and disagreement level in [26], stakeholder preferences, time consumption, and accuracy in [65], accuracy in [58], requirements dependency in [52], and minimum disagreement in [54].

Multiple studies were analyzed to test the effectiveness of various techniques for prioritizing requirements based on machine learning. The studies that were analyzed include [5], [8], [19], [26], [52], [54], [58], [65], and [34]. However, some of the studies did not provide details on the tools used. For example, [5], [34], and [54] did not mention the tools used. In [19], an automated tool was mentioned but not named. On the other hand, the study mentioned in [26] implemented a three-step prioritization approach using a web-based tool called Score, and the Mann-Whitney-Wilcoxon test was used. Other tests used in the studies include the Shapiro-Wilk W test in [52], ANNOVA test in [54], arithmetic mean and standard deviation in [34], SRPTackle-Tool in [65], and a Python tool in [58]. A table showing the parameters used for evaluation is included in the studies.

C. COMPARISON OF AI-BASED REQUIREMENTS PRIORITIZATION TECHNIQUES

Figures 6 and 7 compare Fuzzy logic, Optimization, and Machine Learning software prioritization techniques based on the total number of 'Yes' responses for each parameter. The results suggest that optimization-based RP techniques are claimed to be the most accurate (100%), affordable (91%), and easy to use (54.54%) in the literature, compared to fuzzy logic-based RP techniques and machine learning-based RP techniques. However, optimization-based RP techniques are

TABLE 29. **Analysis of optimization algorithms-based requirement prioritization techniques.**

Techniques	Evaluation Parameters												
	ACCU	SCALE	EFFIC	AFFORD	OPTIM	EOUSE	REDN	TOOL	CASE	STAT	ALGO	FITF	INTER
Bagnall et al. [2]	Yes	No	No	Yes	Yes	Yes	No	No	Yes	Yes	XGA	Yes	No
Feather and Menzies [3]	Yes	No	No	Yes	No	Yes	No	Yes	Yes	Yes	XGA	Yes	No
Deb et al. [4]	Yes	No	Partial	Yes	Yes	Yes	Partia	Yes	Yes	Yes	NSGA 11	Yes	Yes
Greer et al. [7]	Yes	No	No	Yes	Yes	Yes	No	Yes	Yes	Yes	XGA	Yes	Yes
Harman et al. [9]	Yes	No	Partial	Yes	Yes	No	No	No	No	No	POLY	Yes	No
Baker et al. [10]	Yes	Partial	Partial	Yes	Yes	No	No	Yes	Yes	Yes	GREEDY	Yes	No
Saliu and Ruhe [11]	Yes	Partial	No	Yes	Yes	Yes	No	Yes	Yes	No	BORPES	Yes	Yes
Quiroz et al. [15]	Yes	No	No	No	Yes	Yes	No	Yes	Yes	Yes	IGA	Yes	Yes
Zhang et al. [16]	Yes	Partial	Partial	Yes	Yes	No	Partial	No	Yes	Yes	XGA	Yes	Yes
Finkelstein et al. [17]	Yes	Partial	Partial	Yes	Yes	Yes	No	No	Yes	Yes	MOEA	Yes	Yes
J.J. Durilio and Zhang [18]	Yes	Partial	No	Yes	Yes	No	Partial	No	Yes	Yes	MOCell	Yes	No
Tonella et al. [20]	Yes	Partial	No	Yes	No	No	No	No	Yes	Yes	XGA	Yes	Yes
Sagrado et al. [21]	Yes	No	No	No	No	No	No	No	Yes	Yes	XGA	Yes	No
De Souza et al. [23]	Yes	Partial	No	Yes	Yes	Yes	Partial	No	Yes	Yes	ANT	Yes	No
Tonella et al. [27]	Yes	Partial	Partial	Yes	No	No	No	No	Yes	Yes	XGA	Yes	Yes
Gongalez and Toledano [40]	Yes	Partial	No	Yes	Yes	Yes	Partial	No	Yes	Yes	DEPT	Yes	No
Valsala and Nair [43]	Yes	No	Partial	Yes	Yes	No	No	No	Yes	Yes	XGA	Yes	Yes
Kumari and Srinivas [42]	Yes	Yes	Partial	Yes	Yes	Yes	Yes	Yes	Yes	Yes	QEMEA	Yes	No
Marghny et al. [50]	Yes	Partial	Partial	Yes	Yes	No	Partial	No	Yes	Yes	XGA	Yes	No
Rao et al. [48]	Yes	Partial	Yes	Yes	Yes	Yes	Partial	Yes	Yes	Yes	MO-JAVA	Yes	No
Ahuja et al. [55]	Yes	Partial	Partial	Yes	Yes	Yes	Partial	No	Yes	Yes	XGA	Yes	Yes
Alrezaamiri et al. [63]	Yes	No	Partial	Yes	Yes	No	Partial	Yes	Yes	Yes	MOABC	Yes	No

TABLE 30. **Analysis of machine learning-based requirements prioritization techniques.**

Techniques	Evaluation Parameters												
	ACCU	SCALE	EFFIC	AFFORD	OPTIM	EOUSE	REDN	TOOL	CASE	STAT	CLASS	MODEL	FSET
Avesani et al. [5]	Partial	Yes	Yes	No	Yes	No	Yes	No	Yes	Yes	NNC	XLM	Yes
Avesani et al. [8]	Yes	Yes	No	No	Yes	No	Yes	No	Yes	Yes	BIN	IM	Yes
Duan, et al. [19]	No	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	NFRC	PNM	Yes
Perini et al. [26]	Partial	Yes	No	Yes	Yes	No	Yes	Yes	Yes	Yes	BIN	CBIM	Yes
Achimugu & Selamat [34]	Partial	Yes	Yes	No	Yes	Yes	Partial	Yes	Yes	Yes	KMEANS	HYBRID	Yes
Shao et al. [52]	Partial	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	RANK-B	DRank-IM	Yes
Gupta and Gupta [54]	Partial	Yes	Yes	No	Yes	Yes	Yes	No	Yes	Yes	CDBR	IM	Yes
Bollumpally et al. [58]	Yes	Yes	Yes	Yes	No	No	No	Yes	Yes	Yes	BWLC	XGBoost	Yes
Hujainah et al. [65]	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	KMEANS	SRPTackle	Yes

also found to be the least scalable (4.54%), least efficient, and least effective in handling redundancy (4.54%), as compared to the surveyed fuzzy logic and machine learning-based RP techniques. These findings highlight research gaps that need to be addressed in future work.

According to research, only 41% of optimization-based RP techniques provide tool-related information, while 95.45% use case studies as their research methods and 91% are statistically analyzed. The authors of these studies are encouraged to provide information about their tools to help other researchers in the research process.

On the other hand, 60% of studies on Fuzzy logic-based RP techniques report that their methods are scalable and affordable. However, only 73.33% of these studies have been tested using case studies. Interestingly, accuracy (13.33%) and optimal solution (20%) are not commonly discussed or evaluated. It is also noteworthy that only 46.66% of fuzzy logic techniques are analyzed statistically, in contrast to optimization and machine learning-based RP techniques.

Machine learning-based RP techniques have shown exceptional results in scalability (100%), optimal solution (88.88%), efficiency (77.77%), and redundancy handling (66.66%). However, only 33.33% of these studies claim that machine learning-based RP techniques produce more than 80% accurate results. It is worth noting that all surveyed machine learning-based techniques provide tool support information, and case studies are conducted with statistical analysis.

The overall results of this study to highlight the identified limitations and strengths of three surveyed AI-based RP techniques are demonstrated below in Figure 7:

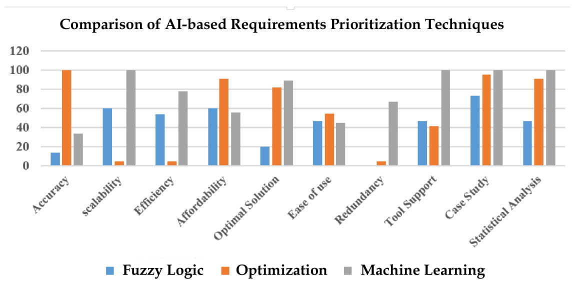
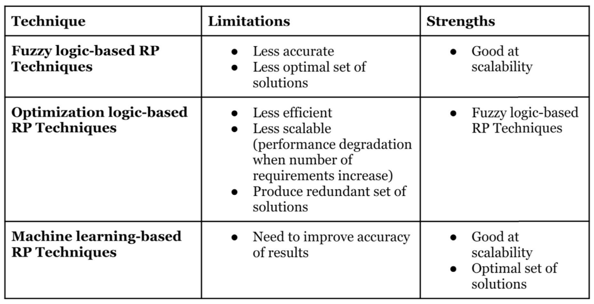


FIGURE 6. Comparison of AI-based requirements prioritization techniques.



Technique	Limitations	Strengths
Fuzzy logic-based RP Techniques	• Less accurate • Less optimal set of solutions	• Good at scalability
Optimization logic-based RP Techniques	• Less efficient • Less scalable (performance degradation when number of requirements increase) • Produce redundant set of solutions	• Fuzzy logic-based RP Techniques
Machine learning-based RP Techniques	• Need to improve accuracy of results	• Good at scalability • Optimal set of solutions

FIGURE 7. Limitations and strengths of AI-based requirements prioritization techniques.

D. ADVANTAGES OF MACHINE LEARNING ALGORITHMS OVER AI TECHNIQUES (RQ2)

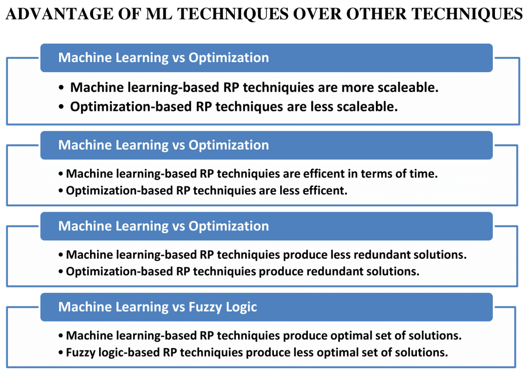
Machine learning techniques are a highly effective and scalable approach for dealing with large sets of requirements. They have been found to outperform expert elicitation effort and requirement priority trade-offs. To achieve better results, it is recommended to use Pareto fronts with a lower spread value, as suggested by various sources, including Avesani et al. [5], Shao et al. [52], Gupta and Gupta [54], and others.

However, when the number of requirements increases, optimization algorithms tend to slow down, as noted in sources such as [2], [4], [7], [11], [15], [16], and [63]. On the other hand, machine learning techniques are generally time-efficient, as indicated by sources such as Avesani et al. [5], Shao et al. [52], Gupta and Gupta [54], and others.

In terms of producing optimal sets of solutions for prioritization, research conducted by Avesani et al. [5], Shao et al. [52], and Gupta and Gupta [54] demonstrate that machine learning techniques are highly effective, whereas fuzzy logic frameworks presented in [25], [37], [39], [51], [60], [61], and [64] have not been verified for optimal sets of solutions.

It's interesting to note that optimization algorithms may generate redundant solutions, sometimes even repeating identical requirements multiple times due to inherent randomness, as noted in [20].

It has been observed that machine learning techniques are becoming increasingly popular for requirements prioritization. These techniques can be easily scaled and provide



ADVANTAGE OF ML TECHNIQUES OVER OTHER TECHNIQUES

Machine Learning vs Optimization
- Machine learning-based RP techniquies are more scaleable.
- Optimization-based RP techniques are less scaleable.

Machine Learning vs Optimization
- Machine learning-based RP techniques are efficent in terms of time.
- Optimization-based RP techniques are less efficent.

Machine Learning vs Optimization
- Machine learning-based RP techniquies produce less redundant solutions.
- Optimization-based RP techniquies produce redundant solutions.

Machine Learning vs Fuzzy Logic
- Machine learning-based RP techniquies produce optimal set of solutions.
- Fuzzy logic-based RP techniquies produce less optimal set of solutions.

FIGURE 8. Advantages of machine learning-based RP techniques over other AI-based RP techniques.

optimal solutions. However, there is room for improvement in terms of the accuracy of prioritization results [5], [26]. The benefits of using machine learning-based RP techniques are summarized in Figure 8.

E. LIMITATIONS OF APPLYING AI-BASED TECHNIQUES FOR SOFTWARE REQUIREMENTS PRIORITIZATION (RQ3)

There are various AI techniques available to prioritize software requirements, but each of them comes with their own set of advantages and limitations. Fuzzy logic-based techniques, for example, may lack accuracy and may not always provide the best solution. Many of these techniques only offer theoretical or framework models that have not been statistically validated. Ejnioui et al. [25] have introduced the cardinality theory to determine the expected value, and they plan to test their simulation algorithm for accuracy in the future. Momeni et al. [31] have proposed a neuro-fuzzy-based approach that consumes 9% less time than other techniques. Sadiq and Jain [30] have suggested a fuzzy technique for requirements prioritization in a goal-oriented requirements elicitation process, but their tool is yet to be tested in a real setting for accuracy. Jawale and Bhole [37] have developed an adaptation module, but its accuracy is still unknown as it has not been tested on real datasets. Dabbagh and Lee [39] have provided an illustrative example for prioritizing requirements, but more experiments in the real world are needed to evaluate the technique's accuracy [51], [60], [61].

As the number of requirements increases, optimization techniques may slow down and produce redundant sets of solutions. Furthermore, these techniques may repeat the same requirements multiple times within a solution, due to their random nature [[2], [3], [4], [7], [11], [15], [16], [63]].

Simulated annealing, proposed by Bagnall et al. [2], is best suited for smaller problems and may not yield optimal solutions for larger problems. Both heuristic and exact methods have room for improvement.

Feather and Menzies [3] present a defect and prevention model that relies heavily on TAR2. This makes it less efficient and unable to scale unless TAR2 scales as well.

Manual selection can also be time-consuming and may not provide ideal solutions. The EVOLVE method, proposed by Greer and Ruhi, becomes less efficient as the population size increases. While the probability of obtaining better results improves, it comes at the expense of computation time. NSGA, proposed by Zhang et al. [16], is effective for Specific Multi-Objective Next Release Problems (MONRP) but is unable to handle dependency problems.

Quiroz et al. [15] propose a technique that requires testing with a large sample size to determine the usability of the tool and its effectiveness in steering and biasing the evolution of user interfaces. Meanwhile, Sagrado et al. [21] utilize GRASP, ACS, and GA techniques for prioritization and individual repair to transform valid solutions. However, the quality of the solution is yet to be tested, which will be done in the future. Tonella et al. [20] present an interactive genetic algorithm that needs empirical investigation with real users. The convergence of the method is not guaranteed, and it depends on the consistency of the fitness function over time during incremental knowledge acquisition. The main problem with this approach is redundancy.

De Souza et al. [23] offer an ant colony optimization technique for prioritization but with less efficiency in terms of implementation time. When evaluating the same number of solutions, ACO is significantly slower than the other two metaheuristics, potentially due to its more sophisticated constructive process in constructing potential solutions. Finally, Gongalez and Toledano [40] propose a differential evolution method. However, it requires further research as it has only been tested on two datasets.

Machine learning algorithms need to be accurate to meet the desired level of agreement among prioritized requirements. A case-based ranking system proposed by Avesani et al. [5] has limitations, as it puts a heavy cognitive load on users when providing a rated preference relation, leading to approximation errors. This technique must be improved because it only shows a 4% disagreement for 10% of pairs of requirements.

Duan et al. [19] present a technique that offers 74% recall and 16% precision. However, the approach's limitations are tied to the underlying classification, traceability, and clustering algorithms that rely on probabilistic data mining and information extraction techniques. Therefore, the output may not provide perfect accuracy or recall.

Perini et al. [26] propose a case-based ranking system that generates a 12% disagreement between desired and achieved results and a 24% disagreement for bug criteria. However, this technique cannot handle dependencies and cannot update ranks on runtime.

Expanding the application of DRank proposed by Shao et al. [52] by including new dependency types is crucial. The model currently only supports contribution and business dependencies. Table 31 below summarizes the limitations of the AI-based requirements prioritization techniques.

VI. DISCUSSION

This section discusses the results of our study and identifies areas for future research based on literature gaps. Our systematic literature review (SLR) reveals a substantial increase in the use of AI techniques for prioritizing software requirements. Demographic analysis indicates that this area attracts the attention of both researchers and requirements engineers. Fig. 5 shows that most studies in this direction were published in 2015 and 2017. The total number of papers published for fuzzy logic is 15, for optimization techniques it's 22, and for machine learning it's 9. However, the trend of using machine learning techniques is rapidly increasing.

Fig. 4 shows that high-quality journals account for 61% of the publications from 2000 to 2021, followed by conferences with 35% and books with 4%. This indicates an increasing effort to explore this research direction.

After analyzing the data in Tables 28, 29, and 30, we conclude that nine parameters are commonly used to evaluate all surveyed fuzzy logic, optimization, and machine learning-based requirements prioritization techniques. These parameters include accuracy, scalability, efficiency, affordability, the optimal set of solutions, ease of use, tool support, case study, and statistical analysis, with varying levels of 'Yes', 'Partial', and 'No' responses. However, some parameters are specific to certain AI techniques. For instance, consistency ratio, self-adaptive, and positive membership functions are only used for fuzzy logic-based prioritization techniques. Fitness function and interactivity are used to evaluate optimization-based prioritization techniques, while classifier, classification model, and feature set are used for machine learning-based prioritization techniques. The parameter of 'Redundancy problem handling' is another evaluation criterion used for both machine learning and optimization-based prioritization techniques, but not for surveyed fuzzy logic techniques. These specific parameters are related to the functionality of these AI-based prioritization techniques.

The study conducted a systematic literature review (SLR) to evaluate different techniques for prioritizing requirements, including fuzzy logic, optimization, and machine learning. It identified the strengths and weaknesses of these techniques, as well as the advantages of using machine learning over other AI-based approaches. However, the study also highlighted the limitations of applying AI techniques in requirements prioritization.

Based on the results of the systematic literature review (SLR), it has been observed that Fuzzy logic has the potential to handle a vast number of requirements [24], [25], [30], [31], [35], [37], [49], [59], [61]. However, there is a need to enhance its accuracy [25], [30], [31], [36], [37], [39], [44], [49], [51], [59], [60], [61], [64] and develop an optimal set of solutions [25], [30], [31], [37], [39], [44], [49], [51], [59], [60], [61], [64]. This aspect requires further research to be thoroughly explored in the future.

TABLE 31. Limitations of applying AI-based requirements prioritization techniques.

Paper ID	Authors	Technique	Limitations
P2	Bagnall et al. 2001 [2]	Simulated Annealing	The results presented in this paper are best for smaller problems but for larger problems, this technique may fail to yield an optimal solution in a reasonable time. Despite these solutions, there is scope for further development in heuristic and exact techniques.
P3	Feather and Menzies, 2002 [3]	Defect Detection and Prevention Model	Lack of attention on key decision points and lack of chance for specialists to bring their expertise to the process. This model heavily depends upon the TAR2 treatment learner. The proposed method would not scale until TAR2 scales. DDP implementation took a long time on large datasets. Manual selection also takes time and may fail to find optimal solutions.
P5	Avesani et al., 2003 [5]	Case-Based Ranking for Decision Support Systems	Cognitive overload required by the user to provide rated preference relation does not reduce approximation error.
P7	Greer and Ruhe 2004 [7]	EVOLVE	Less efficient. This technique Concentrates on effort estimates and their prioritization by appropriate stakeholders. Although a greater population size increases the likelihood of getting a better solution, it speeds up the optimization process at the expense of computation time.
P15	Quiroz et al. 2007 [15]	IGA	The utility of the tool needs to be evaluated with a larger sample to check its effectiveness.
P16	Zhang et al. 2007 [16]	Specific Multi-Objective Next Release Problem (MONRP) NSGA-11	This study doesn't handle dependency problems.
P19	Duan, et al. 2009 [19]	Machine Learning Technique	The limitations of the approach are closely related to the limitations of the underlying traceability, classification, and Clustering algorithms, all of which are based upon data mining and information retrieval techniques that are probabilistic and that, therefore, do not return Perfect precision or recall in the results.
P20	Tonella et al. 2010 [20]	IGA	To evaluate the real, practical viability and acceptability of the trade-off given by IGA, as opposed to AHP, empirical research is required. The algorithm's ability to converge relies on how consistently the fitness function stables over time as new knowledge is acquired. The primary issue with this strategy is redundancy. The authors intend to carry out empirical research with real (as opposed to artificial) users.
P21	Sagrado et al. 2010 [21]	GRASP, GA, ACS	Individuals are fixed (or turned into legitimate solutions removing individual requirements in increasing order of their profit worth. The authors plan to study the effectiveness of the solutions identified and enhance the requirements selection process in future work.
P23	De Souza et al. 2011 [23]	ACS	Less efficient in terms of execution time. ACO performs complex constructive in the building of each candidate solution.
P25	Ejnioui et al. 2012 [25]	Simulation algorithm for fuzzy multi-attribute decision making	Cardinality theory presented but the proposed theory is not tested. Hence the accuracy is still unanswered.
P26	Perini et al. 2013 [26]	CBRank	Further studies need to be conducted to get more accurate results. Future work is handling dependencies and updating ranks on run time.
P37	Jawale and Bhole, 2015 [37]	Adaptive Fuzzy Hierarchical cumulative Voting	No case study and no statistical testing. The adaptation module is not tested for accuracy.
P39	Dabbagh and Lee,2015 [39]	Fuzzy Logic	The proposed approach is not evaluated in a real setting. The illustrative example is not enough to claim accuracy.
P40	Gongalez and Toledano, 2015 [40]	Differential evolution algorithm	This technique is implemented using only two real datasets. More studies need to be conducted to check the effectiveness and scalability of this technique.
P51	Ahmad et al. 2017 [51]	A fuzzy-based MoSCoW Method	The proposed technique is tested using a small set of requirements. The results are not validated.
P52	Shao et al. 2017 [52]	DRank	DRank only supports contribution dependencies and business dependencies. Other dependency types can exist between requirements. It is important to improve the applicability of DRank by covering more dependency types.
P60	Sadia and Faisal, 2019 [60]	Fuzzy Logic	The results are not verified or tested.
P61	Mougouei et al. 2019 [61]	Fuzzy Logic	Real-world experiments are needed to further verify this in practice.

Although optimization techniques [2], [3], [4], [7], [9], [10], [11], [15], [16], [17], [18], [20], [21], [23], [27], [40], [42], [43], [48], [50], [55], [63] can generate accurate results, there is a need for further research to evaluate the efficiency, scalability, and redundancy handling metrics of optimization-based requirements prioritization techniques [2], [3], [4], [7], [9], [10], [11], [15], [16], [17], [18], [20], [21], [23], [27], [40], [43], [48], [50], [55], [63].

Techniques for prioritizing requirements based on machine learning have proven to be scalable [5], [8], [19], [26],

TABLE 32. Strengths and weaknesses of AI-based RP techniques.

Authors	Techniques	Algorithms	Evaluation Criteria	Strengths *Features*	Weaknesses \Limitations
Ramzan et al. [24]	Multi-level value-based intelligent requirement prioritization technique	The FCM algorithm	Average time consumed for prioritization, Average cost, Average precision, Average consistency	A multilevel prioritization where end users, experts, and intelligent systems perform requirement prioritization at various levels Accurately prioritize more than 90% of the requirements	Costlier than conventional techniques.
Ejnioui et al. [25]	Multi-attribute decision-making method	Fuzzy simulation algorithm	Sclability, ease of use, affordability, Time efficiency	It handles imprecise data by modeling various quality attributes, highly flexible since it can accommodate any number of attributes, low-cost, time-efficiency, and ease of implementation	No case study was performed to compare results in real scenarios.
Momeni et al. [31]	A Neuro-Fuzzy-Based Approach to Software Quality Requirements Prioritization	Neuro-fuzzy Approach	Time consumption, cost, more efficient in terms of both the time consumption and the learning ability compared to related approach	It is tested on a small set of requirements	Accuracy of results is not being tested.
Sadiq and Jain [30]	Fuzzy-based approach in goal-oriented requirements elicitation process in group decision-making process(PRFGORE)	a-level weighted F-preference relation and extent fuzzy analytic hierarchy process	Security, reliability, and performance	It is helpful when information is imprecise and identifies which requirement is the most important in the decision-making process.	An example is illustrated but not tested empirically.
Jawale and Bhole [37]	Adaptive Fuzzy Hierarchical Cumulative Voting (AFHCV)	Fuzzy logic and cumulative voting	Accuracy, Scalability, and Speed,	deal with complexity, ambiguity, and uncertainty, and easily target situations where complex service behavior can deviate from customer expectations.	High time consumption as priorities are assigned to all requirements in the requirement set by every stakeholder manually.

TABLE 32. *(Continued.)* **Strengths and weaknesses of AI-based RP techniques.**

Dabbagh and Lee [39]	Fuzzy Logic and Alpha cut approach	Fuzzy logic	Accuracy, Easy to use,	The main contribution of the proposed approach over existing work is to establish a linkage between FRs and NFRs to perform the process of prioritizing NFRs. Simple, easy to use, and generates an accurate result.	The technique is not evaluated in a real industrial setting to investigate the effectiveness as well as weakness of the approach in practical solutions.
Babar et al. [35]	PHandler value-based intelligent requirement prioritization technique, neural network, and analytical hierarchical process	ANN, AHP, Fuzzy logic	Consistency, Scalability, and Performance, Efficiency	Reduces the extent of expert biases and makes the PHandler efficient. The analytical hierarchy process is applied to prioritized groups of requirements to enhance the scalability	The major limitation is linked with the involvement of highly professional business analysts. Another minor limitation of the proposed system is the small increased effort due to the hybridization of the RV function of the VIRP and AHP. In the RV function, ranking the pRCF and rRCF factors and comparisons in the AHP increase the effort.
Achimugu et al. [36]	Fuzzy multicriteria decision-making (FMCDM) approach	Fuzzy logic	accuracy, Large disparity or disagreement between ranked weights as well as complexities	The proposed approach is accurate, hence completely reliable.	A parallel hybridization of FMADM and evolutionary algorithms to solve requirements prioritization problem.
Mishra et al. [44]	A novel multi-level quality-based insightful prerequisite requirement elicitation procedure	Fuzzy logic	Time, cost, penalty, risk, completeness, understandability	In different environments, the multilevel approach produces better results	Comparative analysis is not made for the validity of results.
Ahmad et al. [51]	SRPusing fuzzy based MoSCoW method	Fuzzy logic and MoSCoW	Triangular fuzzy numbers (TFN) are applied, making this approach simple in representation and computation.	Tested on small datasets. Need to test on large datasets.	The effectiveness of this technique is tested using 10 requirements. A larger number of requirements should be used to empirically evaluate the proposed method.

TABLE 32. *(Continued.)* **Strengths and weaknesses of AI-based RP techniques.**

Gulzar et al. [49]	A novel framework that focuses on mapping usability requirements attributes	Fuzzy logic	efficiency, effectiveness, learnability, satisfaction, and low error rate	identifies correct usability requirement conflicts. More flexibility in identifying conflicts among requirements. Gives correct and optimal results in a considerably short period.	Tested on 12 requirements only. Need to test on a large number of requirements.
Singh et al. [59]	A Hybrid Approach for Requirements Prioritization Using LFPP and ANN	Fuzzy logic	Scalability, Time complexity	Overcomes limitations like negative degree of membership functions having no sense, unique outcomes, scalability problem, time complexity	Need to test a large sample of data for accuracy.
Mougouei et al. [61]	A fuzzy framework for prioritization and partial selection of security requirements	Fuzzy logic	impact, cost, and technical ability	The overall complexity leads to AHP and other techniques	Needs to be implemented in the real world to verify its scalability in practice.
Sadia and Faisal [60]	Volatile Requirement Prioritization using Fuzzy logic	Fuzzy logic	Detectability, Changeability, Dependability	Prioritizes requirements based on volatility	Validation of results is missing.
Bisht and Kushwaha [64]	Parameter Optimization of Software Requirements by Using Fuzzy Algebra	Fuzzy Logic and Bayesian network	Accuracy,	The technique performs well in varying and conflicting environments.	More work needs to be done toward the classification of requirements as critical, essential, peripheral, negotiable non-negotiable, etc.
Bagnall et al. [2]	Exact techniques, Greedy algorithms (GRASP), Local search techniques (Hill climbers, Simulated Annealing)	—	Cost, Profit	Simple to implement and has a simple objective function. Simulated Annealing found the best solutions within less computing time.	GRASP techniques improve results but with an increase in computing time.

TABLE 32. *(Continued.)* **Strengths and weaknesses of AI-based RP techniques.**

Feather and Menzies [3]	A requirement interaction model: DDP	Simulated Annealing	Risk, Optimal solutions, Scalability, Spread of solutions	DDP can represent and reason simultaneously with a multitude of mitigation, and their effectiveness at reducing multiple risks. In actual usage, DDP application sessions have dealt with up to 150 requirements, risks, and mitigation. DDP produces near-optimal solutions.	It takes more time to generate a large number of examples. The model heavily depends upon the TAR2 tool and would not scale unless TAR2 scales.
Deb et al. [4]	A non-dominated and sorting-based multi-objective EA (MOEA), called non-dominated sorting genetic algorithm II (NSGA-II)	Multi-objective evolutionary algorithm	Spread of solutions and Convergence	finds a much better spread of solutions and better convergence near the true Pareto-optimal front.	Researchers in the field should consider such epistatic problems for testing a newly developed algorithm for multi-objective optimization.
Greer et al. [7]	EVOLVE method for Software Release Planning	Genetic Algorithm	Redundancy, Optimal solutions	Generates the most candidate solutions iteratively. There is a great variance in the solutions generated by different runs of the solution algorithm. Considers inherent precedence and coupling constraints. It offers greater flexibility by allowing changes in requirements, constraints, and priorities.	Needs to apply the method in a more complex industrial setting and obtain feedback on its operational aspects and effectiveness.
Harman et al. [9]	Search–Based Approaches to the Component Selection and Prioritization Problem	Search-based Heuristics	Cost, user preference	This study proposes to solve individual knapsack problems by employing search-based heuristics that find optimum or close to optimum solutions to sample the Pareto curve as closely as possible. Simultaneously optimizes multi-objective functions	This paper presents only a problem statement. The proposed idea needs to be executed in real real-world environment.

TABLE 32. *(Continued.)* **Strengths and weaknesses of AI-based RP techniques.**

Baker et al. [10]	Search-Based Approaches to Component Selection and Prioritization for the Next Release Problem	Greedy and simulated annealing algorithms	Cost, customer desirability, development time, and expected revenue	The simulated annealing approach is robust for software selection and ranking problems	The search space for component selection and ranking problem is not well suited to a greedy approach.
Saliu and Ruhe [11]	Bi-Objective Release Planning for Evolving Systems (BORPES)	Constraint algorithm	Level of Satisfaction of SD-coupling	BORPES generates alternative release plans to address decision-making under uncertainty. Independent of the level of granularity at which components are defined, which ensures the scalability of the technique	The technique is evaluated using a single evaluation criterion of the level of satisfaction of SD-coupling. Large-scale validation of the technique using other evaluation criteria is required.
Quiroz et al. [15]	Interactive evolutionary approach	Interactive Genetic Algorithm (IGA)	User preferences and Time	Time efficient technique, Less human fatigue over time	Need to test on a large sample to test the effectiveness and utility of the tool.
Zhang et al. [16]	weighted and Pareto optimal genetic algorithms, together with the NSGA-II algorithm	Pareto optimal genetic algorithms and NSGA-II	Cost, Customer satisfaction	NSGA-II outperform the Pareto GA, both in terms of the diversity of results and in terms of quality of the results. NSGA-II is a computationally efficient algorithm whose complexity is O(mN2). The weighted Single-Objective GA can help find extreme points on the Pareto front, by a suitable choice of weights.	Need to verify the findings by applying a search technique to real-world problems.
Finkelstein et al. [17]	A multi-objective search-based optimization approach to fairness analysis in requirement assignments	NSGA-II, Two-Archive, and Random Search	Cost	the Random data set and Greer data set, NSGA-II and Two-Archive algorithms share almost the same amount as the Pareto front. Indeed, both succeed in covering almost the entire front, showing very similar performance.	The performance of the algorithms could have been improved by individual fine-tuning empirically or through systematic experimentation. The technique needs to be tested by using more criteria such as dependencies between the requirements not addressed in this study.

TABLE 32. *(Continued.)* Strengths and weaknesses of AI-based RP techniques.

J.J. Durilio and Zhang [18]	NSGA-II and MOCell for the Multi-Objective Next-Release Problem	NSGA-II and MOCell	Cost and User satisfaction	MOCell outperforms NSGA-II in terms of the range of solutions covered, while this latter is able of obtaining better solutions than MOCell in large instances.	Need to address dependencies between requirements. Need to verify the results of these techniques by applying search techniques to the real world. The Random search algorithms obtain the worst results in all the instances. The scalability problem of these techniques needs to be resolved.
Tonella et al. [20]	Interactive RP using Genetic Algorithm	Genetic Algorithm	Disagreement and Requirement position Distance	It improves non-interactive optimization. It handles several requirements that are otherwise problematic for state-of-the-art techniques and also minimizes the disagreement between a total order of prioritized requirements and the various constraints	More empirical investigation is necessary to assess the actual, practical viability and acceptability of the trade-off offered by IGA, as compared to AHP. Need to test using other metrics too as other metrics may be more meaningful or more appropriate.
Sagrado et al. [21]	Search Techniques for Next Release Problem	A greedy randomized adaptive search procedure (GRASP), a genetic algorithm (GA), and an ant colony system (ACS)	Satisfaction, Effort, and Time	The solution obtained is repaired greedily to verify all the problem restrictions. GRASP found always the best solution in terms of satisfaction.	GA found slightly worse solutions in terms of satisfaction. The quality of solutions needs to be improved.
De Souza et al. [23]	Ant colony optimization technique	ACS Algorithm	Execution Time, Quality of solutions	ACO algorithm generates more accurate solutions with little computational effort to the Software Release Planning problem when compared to Genetic Algorithm and Simulated Annealing.	ACO operated substantially slower than the Simulated Annealing and Genetic algorithm when evaluating the same number of solutions. The algorithm tested on synthetically generated data. Needs to test on real datasets.

TABLE 32. *(Continued.)* **Strengths and weaknesses of AI-based RP techniques.**

Tonella et al. [27]	Interactive Requirements Prioritization Technique	Interactive Genetic Algorithm	Number of disagreements, the average distance of the position of a requirement in the rank from the position of the same requirement in the GS	It Produces a good approximation of the reference requirements ranking, requiring an acceptable manual effort and tolerating a reasonable human error rate. IGA can converge more quickly to an ordering.	This algorithm is tested on two metrics. Other metrics may be more useful. It is a simple user error model to simulate a user who occasionally makes errors. A more sophisticated model is required to test.
Gongalez and Toledano [40]	Differential Evolution with Pareto Tournaments	Differential Evolution (DE) evolutionary algorithm	Development cost and overall clients' satisfaction	DEPT can explore better the space search for solutions because it can generate more diverse non-dominated solutions, which are the best	Need to use this search technique for larger real-world problems, or even for other Software Engineering problems.
Valsala and Nair [43]	Revamped Integer Linear Programming (RILP) model	Enriched Genetic Algorithm (EGA) Revamped the ILP algorithm	Development cost Period	Overcoming the premature convergence problem. Minimizes the project cost and the project period. Yields an optimal solution.	It is tested on 8 requirements only. Needs to test on large sets of requirements.
Kumari and Srinivas [42]	The hybridization of Differential Evolution with Genetic Algorithms coupled with quantum computing concepts (MQHDE), Quantum-inspired Elitist Multi-objective Evolutionary Algorithm (QEMEA)	QMDEA, MQHDE, NSGA-II	Implementation Cost, Customer satisfaction, Redundancy handling	Effectively balance the two issues of multi-objective optimization - convergence and diversity. QEMEA does not require additional genetic operators (i.e. crossover and mutation operators. QEMEA uses elitism which ensures the quality of solutions. MQHDE can be easily scaled up for more objectives and is parameter-free.	The benchmark taken from literature may not true representation of real-world situations. More real-world applications need to be experimented with.
Marghny et al. [50]	Meta-heuristic Techniques	Non-dominated Sorting Genetic Algorithm with Pareto Tournament for Multi-Objective Optimization (NSGA-IIPT)	system's development cost and maximizes customer satisfaction	The duplicate solutions are bound to be removed to diversify solutions as well as leave the chance for new solutions. It minimizes the total system's development cost and maximizes customer satisfaction totality.	Need to test on larger real-world datasets by taking assistance from system design experts to address dependencies between requirements.

TABLE 32. *(Continued.)* **Strengths and weaknesses of AI-based RP techniques.**

Rao et al. [48]	Heuristic approach for handling constraints known as the constrained-dominance concept	MO-Jaya algorithm	Computational Time	It can provide multiple optimal solutions in a single simulation run. It shows a higher convergence rate as compared to NSGA-II. MO-Jaya requires only common control parameters and does not require any algorithm-specific parameters for its work	The MO-Jaya is applied only to modern machining processes. It needs to be tested on other traditional and modern machining processes to check its effectiveness.
Ahuja et al. [55]	VOP, EVOLVE, AHP, NAT, SERUM	-	Accuracy, Scalability, ease of use, Risk, Time, Effort, Total number of comparisons	Scalability, and accuracy of value-oriented prioritization (VOP) are higher than binary search techniques. It is simple, easy to use, and handles more requirements in less time.	EVOLVE and SERUM both calculate the benefits while implementing the requirements but EVOLVE does not estimate the cost factor. Need to test the techniques in real-world settings to validate the results.
Alrezaamiri et al. [63]	A parallel algorithm based on the master–slave model	Parallel multi-objective artificial bee colony algorithm (MOABC)	Efficiency, effort,	Reduces execution time significantly through improvement in the quality of the solution	Two types of limitations were defined for this problem. The first limitation was the dependencies between the requirements, and the second was the cost of product development. Applying these limitations makes us face a restricted optimization problem.
Avesani et al. [5]	Case-Based Ranking for Decision Support Systems	Boosting Algorithm	Accuracy	Effectively estimates pairwise preferences and reduces the effort of the elicitation process.	When the number of cases increases the rated preference and the related cognitive overload becomes useless. The paper omitted to illustrate the design detail of the acquisition policy.

TABLE 32. *(Continued.)* Strengths and weaknesses of AI-based RP techniques.

Avesani et al. [8]	Machine Learning	Machine learning	Accuracy and efficiency	Exploit machine learning techniques to overcome AHP limits. It maintains the costs of the prioritization process lower than its benefits. Prioritization results are approximately 96% accurate.	The difference between the forward and backward errors is a side effect of the evaluation approach. Need to handle dependencies between requirements.
Duan, et al.[19]	Multiple orthogonal clustering algorithms	NFR classifier, Spherical K-means (SPK)	Accuracy, efficiency	It scales to many thousands of requirements. It is useful in organizing the massive amounts of stakeholders' needs.	Probabilistic in nature, therefore do not return perfect precision or recall of the results.
Perini et al. [26]	CBRank	Rank boosting, Machine learning	Elicitation effort and ranking accuracy, ease of use, time consumption	Reduces the elicitation effort, that is the amount of information required from stakeholders, to achieve rankings of a given quality degree. It is highly adaptable to different domains.	More sophisticated pair-sampling policies need to be addressed. Needs to be tested in more complex real settings. Dependences between requirements need to be considered.
Achimugu and Selamat [34]	Clustering evolutionary-based algorithms	K-Means and Evolutionary Algorithms	Accuracy, scalability	It resolves the issues of scalability, rank reversals, and computational complexities. It minimizes disagreements between prioritized requirements. k-means algorithm needs the least number of evaluations compared to other algorithms.	-
Shao et al. [52]	DRank	RankBoost, PageRank-Req algorithm	Time	Less time-consuming and more effective than EVOLVE, a Case-Based Ranking, Analytical Hierarchy Process.	Considering requirement dependencies can improve the accuracy. It provides an easy-to-use evaluation attributes tree (PEAT) for the ranking criteria selection. DRank should be tested in more industrial projects to improve the external validity.

TABLE 32. *(Continued.)* **Strengths and weaknesses of AI-based RP techniques.**

					DRank only supports contribution dependencies and business dependencies. Other dependency types may exist.
Gupta and Gupta [54]	Dependency-based collaborative requirement prioritization approach (CDBR)	Particle Swarm Optimization (PSO) algorithm	Accuracy, the Processing time, Scalability	In terms of efficiency and processing time, CDBR exceeds AHP and IGA. Converges faster than Genetic Algorithm. It is Scalable.	This approach does not consider execution relation among requirements dependencies.
Bollumpally et al. [58]	A Machine Learning Approach to Workflow Prioritization	XGBoost	Accuracy	Constructs a priority score at a higher level of granularity.	The entire model needs to be re-trained in the case that a single department has substantial modifications.
Hujainah et al. [65]	SRPTackle A semi-automated requirements prioritization technique for scalable requirements	Clustering algorithms (K-means and K-means++) and a binary search tree	Accuracy, Time Consumption	It prioritizes large-scale requirements with reduced time consumption and its effectiveness in addressing the key challenges in comparison with other techniques.	Handling requirement inter-dependencies is another important consideration. Implication to other projects and datasets other than RALIC benchmark datasets is recommended.

[34], [52], [54], [65] and capable of producing optimal solutions [5], [8], [19], [26], [34], [52], [54], [65]. However, the accuracy of their results could be improved [5], [19], [26], [34], [52], [54].

Accuracy of the results claimed in different ML studies is 4% disagreement in [8], 10%, 15%, 20% disagreement in [5], 74% recall and 16% precision in [19], 12% disagreement between desired and achieved results and 24% disagreement for bug criteria in [26] which is strong evidence that machine learning techniques for requirements prioritization need to be well researched focusing accuracy parameter.

In summary, the findings revealed that the limitation of one of the AI-based techniques is the strength of other AI-based RP techniques and vice versa. For instance, both fuzzy logic-based RP techniques and machine learning-based RP need to improve the accuracy of prioritization while optimization-based RP techniques are good at accuracy. Similarly, fuzzy logic-based RP techniques produce less optimal solutions while machine learning-based RP techniques are good at generating optimal solutions. The scalability is a common problem of both fuzzy logic and machine learning-based RP techniques and the strength of optimization-based RP techniques.

The literature survey results conclude that the limitations of AI-based RP techniques such as accuracy, optimal set of solutions, efficiency, scalability, and redundancy handling still exist. If RP techniques are to be pervasively adopted in the field of RE, there is a strong need to adopt the identified current strengths and address their inherent limitations. Thus, the current work presents a small leap forward that can serve as guidelines for the future development of RP techniques.

The findings of this SLR are very helpful for researchers as it gives directions for future work. The identified strengths of AI-based techniques may be merged to propose a novel idea to address the identified weaknesses of AI-based techniques. The identified weaknesses of AI-based requirements prioritization techniques invite research attention which is a major contribution of this SLR.

VII. RELATED WORK

Literature shows many conventional and systematic literature reviews have been conducted so far to analyze the domain

TABLE 33. Results of quality scores of selected studies.

Paper ID	QAC1	QAC2	QAC3	QAC4	QAC5	Overall Score/5
P2	1	1	1	1	1	5
P3	1	1	1	1	1	5
P4	1	1	1	1	1	5
P5	1	1	1	1	1	5
P7	1	1	1	1	1	5
P8	1	0.5	1	1	1	4.5
P9	1	0.5	1	0.5	0	3
P10	1	1	1	1	1	5
P11	1	0.5	1	1	1	4.5
P15	1	1	1	1	1	5
P16	1	0.5	1	0.5	1	4
P17	1	1	1	1	1	5
P18	1	1	1	1	1	5
P19	1	1	1	1	1	5
P20	1	1	1	1	1	5
P21	0.5	0.5	1	0.5	1	3.5
P23	1	1	1	1	1	5
P24	1	1	1	1	1	5
P25	1	0.5	1	0.5	0	3
P26	1	1	1	1	1	5
P27	1	1	1	1	1	5
P30	1	0.5	1	1	1	4.5
P31	1	0.5	1	1	1	4.5
P34	1	1	1	1	1	5
P35	1	1	1	1	1	5
P36	1	1	1	1	1	5
P37	1	0.5	1	1	0	3.5
P39	1	0.5	1	1	0	3.5
P40	1	1	1	1	1	5
P42	1	1	1	1	1	5
P43	1	1	1	1	1	5
P44	1	0.5	1	0.5	1	4
P48	1	1	1	1	1	5
P49	1	0.5	1	1	1	4.5
P50	1	1	1	1	1	5
P51	1	0.5	1	1	1	4.5
P52	1	1	1	1	1	5
P54	1	1	1	1	1	5
P55	1	0.5	1	1	1	4.5
P58	1	0.5	1	1	1	4.5
P59	1	1	1	1	1	5
P60	1	0.5	1	1	1	4.5
P61	1	1	1	1	1	5
P63	1	1	1	1	1	5
P64	1	1	1	1	0	4
P65	1	1	1	1	1	5

of requirements prioritization from different perspectives and angles. Nine review studies related to the RP process are collected.

Kaur and Bawa [28] present an overview of conventional prioritization strategies, followed by an analysis in light of software requirements prioritization. Prioritization techniques are classified as manual vs. algorithmic procedures, simple vs. complicated in terms of complexity, fine vs. coarse in terms of granularity, and ratio vs. ordinal in terms of scale in this study. This study analyzes 7 prioritization techniques including analytic hierarchy process, value-oriented prioritization, cumulative voting, numerical assignment, binary search tree, planning game, and B tree prioritization rather than AI-based requirements prioritization techniques. All three research questions that have been answered in our SLR i.e., RQ1, RQ2, and RQ3 are missing in this study.

Achimugu et al. [32] conduct a systematic literature study on requirements prioritization and select 73 primary studies to discover that current prioritization techniques have several shortcomings, including scalability, ways for dealing with rank changes throughout requirements evolution, stakeholder coordination, and requirements dependency constraints. In addition, the applicability of existing approaches in a complex and real scenario has yet to be addressed. This SLR identifies limitations of existing techniques and the processes involved in prioritization but the main focus is on the conventional techniques while RQ1 and RQ2 are missing in this study.

Pitangueira et al. [38] offers a thorough study and mapping of the search-based software engineering (SBSE) methodologies that have been presented to handle software requirement selection and priority problems as well as presenting quantitative and qualitative assessment whereas our study focuses on strengths and weaknesses of fuzzy logic, optimization, and machine learning techniques to prioritize requirements.

Devulapalli et al. [41] conducted a survey on requirements prioritization to uncover requirements prioritization practices in software development companies and to better understand the relationship between requirements prioritization and software delivery and resource allocation. This study examines the data to determine which areas need attention in terms of requirements prioritization. The surveyed papers were selected subjectively, rather than following a systematic searching procedure. In contrast, our study is an SLR rather than a summary.

Yousaf et al. [46] conduct a thorough literature review of 10 techniques for requirements prioritization. Most of these techniques are conventional that are compared using the following criteria: time, number of comparisons, ease of use, scalability, and accuracy whereas our study focuses on the evaluation of every aspect of AI-based requirements prioritization techniques using both general as well as specific criteria related to surveyed AI-based requirements prioritization techniques.

Sufian et al. [56] perform a systematic literature review of 33 papers to identify various techniques and tools used for requirements prioritization and the limitations of these techniques. This SLR compares the top ten conventional requirements prioritization techniques based on three parameters: scalability, ease of use, and consistency whereas our study evaluates three AI-based requirements prioritization techniques to identify their strengths as well as weaknesses.

Hujainah et al. [57] conduct a thorough literature review of 122 relevant studies to identify strengths, opportunities, and limitations of current techniques. The stakeholders engaged in the requirements prioritization process are identified and new stakeholder categories are offered. The study demonstrates that present methodologies have significant limitations in terms of time utilization, scalability, quantification, and priority of participating stakeholders, as well as requirement interdependence and the requirement for highly skilled human involvement. The scope of this survey is much broader than that of our study as it covers literature analyzing 108 techniques which include most of the conventional requirements prioritization techniques whereas our study analyzes AI-based requirements prioritization techniques using both common as well as specific evaluation criteria to investigate the strengths and limitations of these techniques in greater depths.

Bukhsh et al. [62] compare requirements prioritization techniques proposed in 102 papers between 2007 and 2019 using a variety of criteria such as accuracy, ease of use, scalability, efficiency, attractiveness, time consumption, fault tolerance, reliability, complexity and level of automation. Out of the 15 studies, 13 are compared using specific criteria with the AHP with at least one other RP method whereas in our study focus is on evaluation of AI-based RP techniques to identify their strengths and weaknesses. Authors of this SLR come to the conclusion that the Analytical Hierarchy Process is the most precise and widely used approach for prioritizing requirements in industry. However, scalability is still a major problem. Furthermore, authors claim that machine learning has potential in addressing these problems.

Dabrowski et al. [66] conduct comprehensive SLR covering 182 papers published between 2012 and 2020 to identify various techniques to analyze App reviews for software engineering and classifies App review analysis not only in terms of mined information and applied data mining techniques but most importantly in terms of supported software engineering activities. This study evaluates 22 review mining approaches on basis of evaluation criteria: accuracy, efficiency, usefulness, informativeness and usability whereas our study differs in domain i.e. evaluation of AI-based requirements prioritization techniques. Our study differs from this SLR as we focus on detailed and specific analysis of AI-based requirements prioritization techniques using key parameters to identify their potential benefits and limitations.

This review is a unique work synthesizing knowledge from the literature for evaluation of AI-based requirements prioritization techniques using key parameters to identify their strengths and weaknesses. Our SLR, however, differs substantially from previous studies in scope of the literature surveyed and depth of our analysis. Our study is different from previous works in accordance with the generic as well as specific parameters we considered for the evaluation of AI-based techniques to identify strengths and weaknesses

(RQ1), advantages of machine learning techniques over other AI-based requirements prioritization techniques (RQ2) and limitations of applying AI-based techniques in requirements prioritization (RQ3) to identify new research areas.

VIII. CONCLUSION, LIMITATIONS, AND FUTURE WORKS

Software requirements prioritization techniques are deemed successful if they deliver accurate, scalable, efficient, less redundant, and optimal solutions. The performance of AI-based requirements prioritization techniques depends heavily on both generic and specific parameters. This systematic literature review aims to identify the strengths and weaknesses of AI-based RP techniques, including Fuzzy Logic, Machine Learning, and optimization algorithm-based techniques for requirement prioritization. The evaluation criteria used are of two types: general criteria for evaluation, and specific criteria for fuzzy logic, optimization, and machine learning techniques. These criteria help to identify the pros and cons of AI-based prioritization techniques and enable stakeholders to make informed decisions while prioritizing requirements and selecting prioritization techniques.

Fuzzy logic-based requirements prioritization techniques are known to be scalable but need to improve their accuracy. Optimization-based techniques achieve an accurate prioritized list of requirements but can slow down when there are several requirements, produce a redundant set of solutions, and repeat the same requirements more than once. Machine learning-based techniques produce the best optimal and scalable solutions, but their accuracy needs improvement as they do not achieve a minimum level of disagreement.

There are various AI-based approaches that offer advantages, but they also have limitations due to their nature. Therefore, no technique exists that fits well for all required features in the domain of requirements prioritization. As a result, an improved AI-based software requirements prioritization technique is needed, which integrates the benefits of fuzzy logic, optimization, and machine learning techniques to achieve better results.

Some limitations of existing literature include:

- A limitation of the existing literature is that while using the evaluation criteria, authors did not mention explicitly the reason why they selected the evaluation criteria for their proposed techniques, from where they get these evaluation criteria either from literature or based on their personal experience and what measures they used for the selected evaluation criteria. They have not mentioned that the efficiency parameter is used either in terms of time or computational cost.
- Most of the authors claimed that their proposed techniques are accurate and efficient but they do not statistically analyze the proposed technique so the output values in a number of the selected metrics such as accuracy and efficiency are missing. Therefore, the results produced in these cases are mostly unreliable. The selection of appropriate evaluation metrics is still an open research problem. All the evaluation criteria against each study mentioned in tables 28, 29, 30 with values of 'No' and 'Partial' need rigorous research.
- Most of the studies proposed the techniques to prioritize non-functional requirements while a few studies considered functional requirements. There is a need for coverage of functional aspects in the AI-based RP process as well.
- The existing AI-based requirements prioritization techniques are not aligned with the client's priorities. The client may be bothered by some features and may not for others. Some features may involve profitability and productivity for the client business, and some may not. There is a slogan in software engineering that "Client is always right". However, the client's perspective is missing in existing AI-based RP techniques.
- Most of the existing AI-based RP techniques do not consider technical dependencies while prioritizing requirements. Technical dependencies have great potential in the requirements prioritization process. So considering technical dependencies while prioritization of requirements is still a gray area. There is great potential for further research related to AI-based software requirements prioritization using appropriate parameters.
- A few studies provided 'tool support' information and datasets for results validation. Most of the studies did not use the public data set for validation. Public datasets should be used so that future researchers can conduct empirical studies to reproduce the results and make further improvements.
- The study results show that there are less number of studies on machine learning techniques in Requirements prioritization as compared to fuzzy logic and optimization techniques. Further research is required in the requirements prioritization process by applying machine learning techniques to achieve accuracy and efficiency.
- As shown in Table 28, there are only two fuzzy logic-based techniques that are analyzed using consistency ratio. So there is a need to evaluate fuzzy logic-based techniques based on consistency ratio. Moreover, a self-adaptive feature of fuzzy logic-based RP techniques needs to be well-researched.

In order to achieve reliable and intended results in prioritizing software requirements, it is necessary to shift towards AI-based techniques and carefully select appropriate parameters and criteria for measuring their effectiveness. Although several conventional and AI-based prioritization techniques have been proposed, there is currently no literature survey that focuses on parametric-based evaluation. To address this research gap, further studies are needed to evaluate AI-based techniques using both generic and specific criteria.

To this end, we conducted a thorough literature survey to evaluate AI-based software requirements prioritization techniques using a comprehensive set of parameters. Our analysis revealed that while a variety of prioritization techniques exist, there is still room for improvement in terms of scalability, accuracy, efficiency, redundancy, and the optimal set of solutions. By evaluating these AI-based techniques on various parameters, we hope to provide valuable insights to researchers, requirement analysts, planners, and other stakeholders to help them make informed decisions when selecting the most appropriate prioritization technique for their needs.

Some potential areas of focus include:

1). A fully automated system for requirements prioritization integrating potential features of machine learning, fuzzy logic, and optimization techniques is required to address the problem of accuracy, scalability, efficiency, and redundancy.

2). Future work could focus on developing machine learning-based requirements prioritization methods that can effectively incorporate more accurate, scalable, and less redundant solutions.

3). The DRank system for Requirements prioritization needs to be developed in the future to cover more dependency types other than contribution and business dependencies.

4). Fuzzy logic-based RP systems require improvement in accuracy and the optimal set of solutions. Future work could focus on developing fuzzy logic-based RP methods for scalable solutions, and further improving the prioritization results in terms of accuracy, optimal set of solutions, consistency ratio, and self-adaptivity.

5). Optimization-based RP techniques need to improve prioritization results in terms of efficiency, scalability, and redundancy. Therefore, future work could focus on developing optimization-based RP techniques for more scalable, efficient, and less redundant solutions.

6). Most of the surveyed fuzzy logic-based RP techniques are not tested or validated for the statistical significance of results. Future work may extend to focus on the validation of results.

Overall, there are many directions that future work in AI-based Requirements prioritization problems could take, including improving accuracy, scalability, efficiency, redundancy, consistency ratio, self-adaptability, and an optimal set of solutions.

DECLARATIONS

This section includes declarations as stated below.

- Funding The authors did not receive support from any organization for the submitted work.
- Conflict of interest/Competing interests All authors certify that they have no affiliations with or involvement in any organization or entity with any financial interest or non-financial interest in the subject matter or materials discussed in this manuscript.
- Ethics approval We ensure that accepted principles of ethical and professional conduct have been followed for this research.
- Consent for publication All authors agreed with the content and all gave explicit consent to submit and they obtained consent from the responsible authorities at the institute/organization where the work was carried out before the work was submitted.
- Availability of data and materials The datasets generated during and/or analyzed during the current study are available from the corresponding author upon reasonable request.
- Code availability 'Not applicable'
- Authors' contributions All authors contributed to the study's conception and design. Material preparation, data collection, literature review, analysis, and the first draft of the manuscript was written by [Rahila Anwar] under the supervision of [Dr. Muhammad Bilal Bashir]. All authors commented on previous versions of the manuscript. All authors read and approved the final manuscript.

REFERENCES

[1] L. A. Zadeh, "Fuzzy sets," *Inf. Control*, vol. 8, pp. 338–353, Jun. 1965.

[2] A. J. Bagnall, V. J. Rayward-Smith, and I. M. Whittley, "The next release problem," *Inf. Softw. Technol.*, vol. 43, no. 14, pp. 883–890, Dec. 2001.

[3] M. S. Feather and T. Menzies, "Converging on the optimal attainment of requirements," in *Proc. IEEE Joint Int. Conf. Requirements Eng.*, Essen, Germany, Sep. 2002, pp. 263–270.

[4] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002, doi: 10.1109/4235.996017.

[5] P. Avesani, S. Ferrari, and A. Susi, "Case-based ranking for decision support systems," in *Proc. Int. Conf. Case Based Reasoning*. Berlin, Germany: Springer, 2003, pp. 35–49.

[6] P. Avesani, C. Bazzanella, A. Perini, and A. Susi, "Supporting the requirements prioritization Process. A machine learning approach," in *Proc. 16th Int. Conf. Softw. Eng. Knowl. Eng. (SEKE)*, Canada, 2004, pp. 306–311.

[7] D. Greer and G. Ruhe, "Software release planning: An evolutionary and iterative approach," *Inf. Softw. Technol.*, vol. 46, no. 4, pp. 243–253, Mar. 2004.

[8] P. Avesani, C. Bazzanella, A. Perini, and A. Susi, "Facing scalability issues in requirements prioritization with machine learning techniques," in *Proc. 13th IEEE Int. Conf. Requirements Eng. (RE)*, Aug. 2005, pp. 297–305.

[9] M. Harman, A. Skaliotis, K. Steinhöfel, and P. Baker, "Search-based approaches to the component selection and prioritization problem," in *Proc. 8th Annu. Conf. Genetic Evol. Comput.*, Jul. 2006, pp. 1951–1952.

[10] P. Baker, M. Harman, K. Steinhofel, and A. Skaliotis, "Search based approaches to component selection and prioritization for the next release problem," in *Proc. 22nd IEEE Int. Conf. Softw. Maintenance*, Sep. 2006, pp. 176–185.

[11] M. O. Saliu and G. Ruhe, "Bi-objective release planning for evolving software systems," in *Proc. 6th Joint Meeting Eur. Softw. Eng. Conf. ACM SIGSOFT Symp. Found. Softw. Eng.*, Dubrovnik, Croatia, Sep. 2007, p. 105, doi: 10.1145/1287624.1287641.

[12] L. Karlsson, T. Thelin, B. Regnell, P. Berander, and C. Wohlin, "Pair-wise comparisons versus planning game partitioning—Experiments on requirements prioritisation techniques," *Empirical Softw. Eng.*, vol. 12, no. 1, pp. 3–33, Jan. 2007, doi: 10.1007/s10664-006-7240-4.

[13] B. A. Kitchenham and S. M. Charters, "Guidelines for performing systematic literature reviews in software engineering," Keele Univ. Univ. Durham, Keele, England, Tech. Rep. EBSE-2007-01, 2007.

[14] M. Gen and L. Lin, "Genetic algorithms," in *Wiley Encyclopedia of Computer Science and Engineering*. American Cancer Society, 2007, pp. 1–15.

[15] J. C. Quiroz, S. J. Louis, A. Shankar, and S. M. Dascalu, "Interactive genetic algorithms for user interface design," in *Proc. IEEE Congr. Evol. Comput.*, Sep. 2007, pp. 1366–1373.

[16] Y. Zhang, M. Harman, and S. A. Mansouri, "The multi-objective next release problem," in *Proc. 9th Annu. Conf. Genetic Evol. Comput.*, Jul. 2007, pp. 1129–1137.

[17] 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 Eng.*, vol. 14, no. 4, pp. 231–245, Dec. 2009, doi: 10.1007/s00766-009-0075-y.

[18] J. J. Durillo, Y. Zhang, E. Alba, and A. J. Nebro, ''A study of the multi-objective next release problem,'' in *Proc. 1st Int. Symp. Search Based Softw. Eng.*, May 2009, pp. 49–58, doi: 10.1109/SSBSE.2009.21.

[19] C. Duan, P. Laurent, J. Cleland-Huang, and C. Kwiatkowski, ''Towards automated requirements prioritization and triage,'' *Requirements Eng.*, vol. 14, no. 2, pp. 73–89, Jun. 2009.

[20] P. Tonella, A. Susi, and F. Palma, ''Using interactive GA for requirements prioritization,'' in *Proc. 2nd Int. Symp. Search Based Softw. Eng.*, Sep. 2010, pp. 57–66.

[21] J. del Sagrado, I. M. del Águila, and F. J. Orellana, ''Ant colony optimization for the next release problem: A comparative study,'' in *Proc. 2nd Int. Symp. Search Based Softw. Eng.*, Sep. 2010, pp. 67–76.

[22] J. J. Durillo, Y. Zhang, E. Alba, M. Harman, and A. J. Nebro, ''A study of the bi-objective next release problem,'' *Empirical Softw. Eng.*, vol. 16, no. 1, pp. 29–60, Feb. 2011.

[23] J. T. de Souza, C. L. B. Maia, T. do N. Ferreira, R. A. F. do Carmo, and M. M. A. Brasil, ''An ant colony optimization approach to the software release planning with dependent requirements,'' in *Search Based Software Engineering*. Berlin, Germany: Springer, 2011, pp. 142–157.

[24] M. Ramzan, M. A. Jaffar, and A. A. Shahid, ''Value based intelligent requirement prioritization (VIRP): Expert driven fuzzy logic based prioritization technique,'' *Int. J. Innov. Comput. Inf. Control*, vol. 7, no. 3, pp. 1017–1038, 2011.

[25] A. Ejnioui, C. E. Otero, and A. A. Qureshi, ''Software requirement prioritization using fuzzy multi-attribute decision making,'' in *Proc. IEEE Conf. Open Syst.*, Oct. 2012, pp. 1–6.

[26] A. Perini, A. Susi, and P. Avesani, ''A machine learning approach to software requirements prioritization,'' *IEEE Trans. Softw. Eng.*, vol. 39, no. 4, pp. 445–461, Apr. 2013.

[27] P. Tonella, A. Susi, and F. Palma, ''Interactive requirements prioritization using a genetic algorithm,'' *Inf. Softw. Technol.*, vol. 55, no. 1, pp. 173–187, Jan. 2013.

[28] G. Kaur and S. Bawa, ''A survey of requirement prioritization methods,'' *Int. J. Eng. Res. Technol*, vol. 2, no. 5, pp. 958–962, 2013.

[29] M. Pergher and B. Rossi, ''Requirements prioritization in software engineering: A systematic mapping study,'' in *Proc. 3rd Int. Workshop Empirical Requirements Eng. (EmpiRE)*, Jul. 2013, pp. 40–44, doi: 10.1109/EmpiRE.2013.6615215.

[30] M. Sadiq and S. K. Jain, ''Applying fuzzy preference relation for requirements prioritization in goal oriented requirements elicitation process,'' *Int. J. Syst. Assurance Eng. Manage.*, vol. 5, no. 4, pp. 711–723, Dec. 2014.

[31] H. Momeni, H. Motameni, and M. Larimi, ''A neuro-fuzzy based approach to software quality requirements prioritization,'' *Int. J. Appl. Inf. Syst.*, vol. 7, no. 7, pp. 15–20, Aug. 2014.

[32] P. Achimugu, A. Selamat, R. Ibrahim, and M. N. Mahrin, ''A systematic literature review of software requirements prioritization research,'' *Inf. Softw. Technol.*, vol. 56, no. 6, pp. 568–585, Jun. 2014.

[33] K. Govindan, M. Kaliyan, D. Kannan, and A. N. Haq, ''Barriers analysis for green supply chain management implementation in Indian industries using analytic hierarchy process,'' *Int. J. Prod. Econ.*, vol. 147, pp. 555–568, Jan. 2014.

[34] 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*, vol. 575, A. T. Azar and S. Vaidyanathan, Eds. Cham, Switzerland: Springer, 2015, pp. 73–93, doi: 10.1007/978-3-319-11017-24.

[35] M. I. Babar, M. Ghazali, D. N. A. Jawawi, S. M. Shamsuddin, and N. Ibrahim, ''PHandler: An expert system for a scalable software requirements prioritization process,'' *Knowl.-Based Syst.*, vol. 84, pp. 179–202, Aug. 2015, doi: 10.1016/j.knosys.2015.04.010.

[36] 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, pp. 21–28, Nov. 2015, doi: 10.11113/jt.v77.6321.

[37] B. Jawale and A. T. Bhole, ''Adaptive fuzzy hierarchical cumulative voting: A novel approach toward requirement prioritization,'' *Int. J. Res. Eng. Technol.*, vol. 4, no. 5, pp. 365–370, May 2015.

[38] A. M. Pitangueira, R. S. P. Maciel, and M. Barros, ''Software requirements selection and prioritization using SBSE approaches: A systematic review and mapping of the literature,'' *J. Syst. Softw.*, vol. 103, pp. 267–280, May 2015.

[39] M. Dabbagh and S. P. Lee, ''An approach for prioritizing NFRs according to their relationship with FRs,'' *Lect. Notes Softw. Eng.*, vol. 3, no. 1, pp. 1–5, 2015.

[40] J. M. Chaves-González and M. A. Pérez-Toledano, ''Differential evolution with Pareto tournament for the multi-objective next release problem,'' *Appl. Math. Comput.*, vol. 252, pp. 1–13, Feb. 2015.

[41] S. Devulapalli, A. Khare, and O. R. S. Rao, ''Requirement prioritization-survey and analysis,'' in *Proc. Int. Congr. Inf. Commun. Technol.*, Udaipur, India, Oct. 2015, pp. 567–575.

[42] A. C. Kumari and K. Srinivas, ''Comparing the performance of quantum-inspired evolutionary algorithms for the solution of software requirements selection problem,'' *Inf. Softw. Technol.*, vol. 76, pp. 31–64, Aug. 2016, doi: 10.1016/j.infsof.2016.04.010.

[43] S. Valsala and A. R. Nair, ''Requirement prioritization and scheduling in software release planning using hybrid enriched genetic revamped integer linear programming model,'' *Res. J. Appl. Sci., Eng. Technol.*, vol. 12, no. 3, pp. 347–354, Feb. 2016.

[44] N. Mishra, M. A. Khanum, and K. Agrawal, ''Approach to prioritize the requirements using fuzzy logic,'' presented at the ACEIT Conf., 2016.

[45] F. Evbota, E. Knauss, and A. Sandberg, ''Scaling up the planning game: Collaboration challenges in large-scale agile product development,'' in *Agile Processes in Software Engineering and Extreme Programming*, vol. 251, H. Sharp and T. Hall, Eds. Cham, Switzerland: Springer, 2016, pp. 28–38.

[46] M. Yousuf, M. U. Bokhari, and M. Zeyauddin, ''An analysis of software requirements prioritization techniques: A detailed survey,'' in *Proc. 3rd Int. Conf. Comput. Sustain. Global Develop. (INDIACom)*, Mar. 2016, pp. 3966–3970.

[47] R. Qaddoura, A. Abu-Srhan, M. H. Qasem, and A. Hudaib, ''Requirements prioritization techniques review and analysis,'' in *Proc. Int. Conf. New Trends Comput. Sci. (ICTCS)*, Oct. 2017, pp. 258–263, doi: 10.1109/ICTCS.2017.55.

[48] R. V. Rao, D. P. Rai, and J. Balic, ''A multi-objective algorithm for optimization of modern machining processes,'' *Eng. Appl. Artif. Intell.*, vol. 61, pp. 103–125, May 2017, doi: 10.1016/j.engappai.2017.03.001.

[49] K. Gulzar, J. Sang, M. Ramzan, and M. Kashif, ''Fuzzy approach to prioritize usability requirements conflicts: An experimental evaluation,'' *IEEE Access*, vol. 5, pp. 13570–13577, 2017.

[50] M. H. Marghny, H. M. El-Hawary, and W. H. Dukhan, ''An effective method of systems requirement optimization based on genetic Algorithms,'' *Inf. Sci. Lett.*, vol. 6, no. 1, pp. 15–28, Jan. 2017.

[51] K. S. Ahmad, N. Ahmad, H. Tahir, and S. Khan, ''Fuzzy_MoSCoW: A fuzzy based Moscow method for the prioritization of software requirements,'' in *Proc. Int. Conf. Intell. Comput., Instrum. Control Technol. (ICICICT)*, Kannur, India, Jul. 2017, pp. 433–437.

[52] F. Shao, R. Peng, H. Lai, and B. Wang, ''DRank: A semi-automated requirements prioritization method based on preferences and dependencies,'' *J. Syst. Softw.*, vol. 126, pp. 141–156, Apr. 2017, doi: 10.1016/j.jss.2016.09.043.

[53] A. Hudaib, M. H. Qasem, and N. Obeid, ''FIPA-based semi-centralized protocol for negotiation,'' in *Cybernetics Approaches in Intelligent Systems*. Cham, Switzerland: Springer, 2018, pp. 135–149, doi: 10.1007/978-3-319-67618-0-13.

[54] A. Gupta and C. Gupta, ''CDBR: A semi-automated collaborative execute-before-after dependency-based requirement prioritization approach,'' *J. King Saud Univ. Comput. Inf. Sci.*, vol. 34, no. 2, pp. 421–432, Feb. 2022, doi: 10.1016/j.jksuci.2018.10.004.

[55] H. Ahuja, Sujata, and U. Batra, ''Performance enhancement in requirement prioritization by using least-squares-based random genetic algorithm,'' in *Innovations in Computational Intelligence*, B. Panda, S. Sharma, and U. Batra, Eds. Singapore: Springer, 2018, pp. 251–263.

[56] M. Sufian, Z. Khan, S. Rehman, and W. Haider Butt, ''A systematic literature review: Software requirements prioritization techniques,'' in *Proc. Int. Conf. Frontiers Inf. Technol. (FIT)*, Dec. 2018, pp. 35–40.

[57] F. Hujainah, R. B. A. Bakar, M. A. Abdulgabber, and K. Z. Zamli, "Software requirements prioritisation: A systematic literature review on significance, stakeholders, techniques and challenges," *IEEE Access*, vol. 6, pp. 71497–71523, 2018.

[58] N. R. Bollumpally, A. C. Evans, S. W. Gleave, A. R. Gromadzki, and G. Learmonth, "A machine learning approach to workflow prioritization," in *Proc. Syst. Inf. Eng. Design Symp. (SIEDS)*, Charlottesville, VA, USA, Apr. 2019, pp. 1–5, doi: 10.1109/SIEDS.2019.8735589.

[59] 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*. Singapore: Springer, 2019, pp. 350–364.

[60] H. Sadia, S. Q. Abbas, and M. Faisal, "Volatile requirement prioritization: A fuzzy based approach," *Int. J. Eng. Adv. Technol.*, vol. 8, no. 5, pp. 2467–2472, Jun. 2019.

[61] D. Mougouei, D. M. W. Powers, and E. Mougouei, "A fuzzy framework for prioritization and partial selection of security requirements in software projects," *J. Intell. Fuzzy Syst.*, vol. 37, no. 2, pp. 2671–2686, Sep. 2019.

[62] F. A. Bukhsh, Z. A. Bukhsh, and M. Daneva, "A systematic literature review on requirement prioritization techniques and their empirical evaluation," *Comput. Standards Interface*, vol. 69, Mar. 2020, Art. no. 103389.

[63] H. Alrezaamiri, A. Ebrahimnejad, and H. Motameni, "Parallel multi-objective artificial bee colony algorithm for software requirement optimization," *Requirements Eng.*, vol. 25, no. 3, pp. 363–380, Sep. 2020, doi: 10.1007/s00766-020-00328-y.

[64] A. Bisht and M. Kushwaha, "Parameter optimization of software requirement by using fuzzy algebra," *Int. J. Res. Develop. Appl. Sci. Eng.*, vol. 20, no. 1, 2020. Accessed: Aug. 29, 2021.

[65] F. Hujainah, R. B. A. 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," *Inf. Softw. Technol.*, vol. 131, Mar. 2021, Art. no. 106501, doi: 10.1016/j.infsof.2020.106501.

[66] J. Dąbrowski, E. Letier, A. Perini, and A. Susi, "Analysing app reviews for software engineering: A systematic literature review," *Empirical Softw. Eng.*, vol. 27, no. 2, p. 43, Mar. 2022, doi: 10.1007/s10664-021-10065-7.

RAHILA ANWAR received the M.I.T. degree in information technology from the University of Arid Agriculture Rawalpindi, Pakistan, in 2003, and the M.S. degree in computer science, major in software engineering, from Allama Iqbal Open University, Islamabad, Pakistan, in 2017. She is currently pursuing the Ph.D. degree in computer science with Iqra University, Islamabad, with a focus on AI-based software requirements prioritization.

Since 2008, she has been a Lecturer in computer science with the Higher Education Department of Azad Kashmir. Her research interests include software engineering, emerging techniques of artificial intelligence, and integration of AI techniques in requirement engineering for prioritization.

Ms. Anwar's award includes the Best Teacher Award, in 2007.

MUHAMMAD BILAL BASHIR was born in Faisalabad, in 1985. He received the B.S. degree from the University of Central Punjab, in 2006, the M.S. degree in software and system engineering from Mohammad Ali Jinnah University, in 2009, and the Ph.D. degree from the Capital University of Science and Technology, Islamabad, Pakistan, in 2018, with a focus on search-based mutation testing.

Since 2018, he has been an Assistant Professor with Iqra University, Islamabad. Before joining Iqra University, he has also been a System Administrator in an international web hosting organization, JaguarPC LLC., since 2006. His main research interests include software testing, mobile agents, and the semantic web. He has 14 research publications.